



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Computational Efficiency in Conditional Text Generation:
Exploring Lightweight Architectures in Cross-Lingual and
Multilingual Environments

End of Degree Project

Bachelor's Degree in Data Science

AUTHOR: Hurtado Beneyto, Marc

Tutor: Molina Marco, Antonio

External cotutor: Castro Bleda, María José

Experimental director: Ahuir Esteve, Vicent

ACADEMIC YEAR: 2025/2026

Resum

Aquest Treball Final de Grau aborda el repte de l'eficiència computacional en la generació de text condicionat dins del Processament del Llenguatge Natural, una tasca actualment dominada per models massius d'alt cost energètic. En la línia de la Intel·ligència Artificial Sostenible (Green AI), l'estudi desenvolupa un sistema trilingüe controlable per llengua, longitud i entitats, pensat per a l'execució local en equips de gamma de consum (per exemple, un portàtil o un ordinador de sobretaula corrent). Amb aquest objectiu, s'ha construït un corpus trilingüe adaptat específicament a l'ajustament per instruccions. S'implementa una arquitectura lleugera basada en mT5 i es contrasten dos enfocaments de control: l'ús de prompts en llenguatge natural i una modificació estructural de l'arquitectura mitjançant tokens de control. L'avaluació, amb mètriques clàssiques i LLM-as-a-Judge, mostra que el model assoleix una fidelitat semàntica competitiva i supera models decoder-only molt més grans en el compliment de restriccions. L'enfocament s'ha presentat al CLEF 2026 SimpleText Track, amb una publicació científica associada. Així, el treball consolida els models compactes com una alternativa robusta, privada i sostenible.

Paraules clau: Generació condicionada de textos, Eficiència computacional, Arquitectures lleugeres, Intel·ligència Artificial Sostenible, Green AI, Models de llenguatge, Arquitectura T5, empremta de carboni, Multilingüisme.

Resumen

Este Trabajo Fin de Grado aborda el reto de la eficiencia computacional en la generación de texto condicionado dentro del Procesamiento del Lenguaje Natural, una tarea actualmente dominada por modelos masivos de alto coste energético. En la línea de la Inteligencia Artificial Sostenible (Green AI), el estudio desarrolla un sistema trilingüe controlable por idioma, longitud y entidades, pensado para la ejecución local en equipos de gama de consumo (por ejemplo, un portátil o un ordenador de sobremesa estándar). Con este objetivo, se ha construido un corpus trilingüe adaptado específicamente al ajuste por instrucciones. Se implementa una arquitectura ligera basada en mT5 y se contrastan dos enfoques de control: el uso de prompts en lenguaje natural y una modificación estructural de la arquitectura mediante tokens de control. La evaluación, con métricas clásicas y LLM-as-a-Judge, muestra que el modelo alcanza una fidelidad semántica competitiva y supera a modelos decoder-only mucho más grandes en el cumplimiento de restricciones. El enfoque se ha presentado en el CLEF 2026 SimpleText Track, con una publicación científica asociada. Así, el trabajo consolida los modelos compactos como una alternativa robusta, privada y sostenible.

Palabras clave: Generación condicionada de textos, Eficiencia computacional, Arquitecturas ligeras, Inteligencia Artificial Sostenible, Green AI, Modelos de lenguaje, Arquitectura T5, Huella de carbono, Multilingüismo.

Abstract

This Bachelor’s Thesis addresses the challenge of computational efficiency in conditional text generation within Natural Language Processing, a task currently dominated by massive, high-energy-cost models. In line with Sustainable Artificial Intelligence (Green AI), the study develops a trilingual system controllable by language, length, and entities, designed for local execution on consumer-grade hardware (e.g., a standard laptop or desktop computer). To this end, a trilingual corpus specifically adapted for instruction tuning has been built. A lightweight architecture based on mT5 is implemented, and two control approaches are contrasted: the use of natural language prompts and a structural modification of the architecture using control tokens. The evaluation, using classical metrics and LLM-as-a-Judge, shows that the model achieves competitive semantic fidelity and outperforms much larger decoder-only models in constraint compliance. The approach has been presented at the CLEF 2026 SimpleText Track, with an associated scientific publication. Thus, the work consolidates compact models as a robust, private, and sustainable alternative.

Key words: Conditional text generation, Computational efficiency, Lightweight architectures, Sustainable Artificial Intelligence, Green AI, Language models, T5 architecture, Carbon footprint, Multilingualism.

Contents

Contents	vii
List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Report Structure	4
1.4 Relationship with the Studies Undertaken	5
2 State of the Art	7
2.1 Introduction to Natural Language Processing (NLP)	7
2.2 Historical Evolution and Classical Techniques	7
2.2.1 The Origins: The Era of Rule-Based Systems (1950s–1960s)	8
2.2.2 Statistical NLP: The Rise of Data-Driven Models (1980s–2000s)	9
2.2.3 Deep Learning: The Revolution of Neural Architectures (2010s–Present)	12
2.3 The Transformer Revolution	15
2.3.1 Attention Calculation: Query, Key, and Values	16
2.3.2 Global Architecture: Encoder and Decoder Blocks	17
2.4 The T5 Model Family: The Text-to-Text Paradigm	20
2.4.1 Text-to-Text	20
2.4.2 Architectural Modifications of T5	21
2.4.3 Pretraining and the C4 Corpus	22
2.4.4 Multilingual Version - mT5	23
2.4.5 Instruction Optimization: The FLAN-T5 Model	23
2.5 The Paradigm Shift in LLMs: The Evolution Towards Decoder-Only Networks	25
2.5.1 Efficient Scaling: Mixture of Experts (MoE)	25
2.5.2 The Shift Toward Small Language Models (SLMs)	26
2.6 Conditional Text Generation	26
2.7 State of the Art Critique	27
2.8 Proposal	27
3 Data Preparation and Understanding	29
3.1 Dataset Creation	29
3.1.1 Acquisition of Primary Datasets	29
3.1.2 Dataset Translations	30
3.2 Dataset Conditioning and Enrichment	31
3.2.1 Language Determination	33
3.2.2 Length Categorization	33
3.2.3 Named Entity Extraction	33
3.3 Dataset Cleaning	35
3.3.1 Phase 1: Filtering by Length Ratios (TATOEBA Method)	35
3.3.2 Phase 2: Named Entity Preservation	37
3.3.3 Phase 3: Semantic Similarity via Embeddings	37

3.3.4	Final Acceptance Criteria	38
3.4	Partitions	38
3.4.1	Downsampling Strategy and Evaluation Set Construction	39
3.4.2	Definition of Truncation Thresholds	39
3.4.3	Processing of the Dacsa Corpora (Catalan and Spanish) for Test-I	39
3.4.4	Processing of the Dacsa Corpora (Catalan and Spanish) for Test-NI	39
3.4.5	Processing of the Newsroom Corpus for Test-I	40
3.4.6	Processing of the Newsroom Corpus for Test-NI	41
3.4.7	Homogenization and Design of the Dacsa Validation Set	41
3.4.8	Downsampling and Balancing of the Dacsa Spanish Training Set	41
3.5	Summary and Statistics for the Final Dataset	43
4	Proposed Architectural Framework	49
4.1	Conditioning the mT5 Model via Custom Tokens	49
4.1.1	Adaptation of the Input Layer (SplitEmbedding)	51
4.1.2	Adaptation of the Output Layer (SplitLMHead)	53
4.1.3	Model Input Structure	55
4.2	Conditioning the mT5 Model via Natural Language Sentences	55
4.2.1	Creation of the Sentences	56
4.2.2	Model Input Structure	57
4.3	Generation of Conditional Prefixes	57
5	Training and Fine-Tuning Strategy	61
5.1	Environment and Hardware Setup	61
5.2	Data Preparation and Augmentation	62
5.3	Training Hyperparameters	63
5.4	Low-Rank Adaptation (LoRA)	63
5.5	Prefix Masking Strategy	64
6	Evaluation Methodology	65
6.1	Evaluation Metrics	65
6.1.1	Text Quality Metrics (Linguistic and Semantic Evaluation)	65
6.2	LLM-as-a-Judge	67
6.2.1	Open G-Eval	68
6.3	Quantitative Evaluation (Traditional Metrics)	70
6.3.1	Custom Tokens Conditioning	70
6.3.2	Natural Language Sentence Conditioning	71
6.3.3	Qwen3.6 Prompting	72
6.4	Advanced Qualitative Evaluation (LLM-as-a-Judge)	73
6.5	Comparative Discussion and Optimal Model Selection	74
7	Paper for the CLEF 2026 SimpleText Track	77
7.1	Introduction	77
7.2	Tasks and Datasets	78
7.3	System description	80
7.3.1	Architecture	80
7.3.2	Simplification Action Prediction	80
7.3.3	Action-driven Text Generation	83
7.4	Results and Discussion	85
7.5	Future Work	87
8	Conclusions and Future Work	89
8.1	Future Work	89
	Acknowledgements	91
	Declaration on Generative AI	93

Bibliography	95
Appendices	
A Sustainable Development Goals (SDGs)	103
A.1 Degree of Relationship of the Work with the Sustainable Development Goals (SDGs)	103
A.2 Reflection on the Relationship Between the Thesis and the SDGs	103
B Prepint of the SimpleText paper	105
C Prompts Used for SimpleText	123
D Prompts Used for General LLM as a Judge	129
E Prompts Used for M-Prometheus Evaluation	135
F Prompts Used for Qwen3.6 Model	141

List of Figures

1.1	Percentage distribution of global web content language usage. Own elaboration based on data from (Q-Success / W3Techs, 2026).	3
2.1	Schematic diagram of a Feedforward Neural Network with one hidden layer, illustrating forward propagation (black arrows) and the backpropagation path (dashed red arrows). Synthetic illustration generated with the assistance of Generative AI.	13
2.2	Architecture of a Many-to-One Recurrent Neural Network (RNN) processing a sequence of word embeddings. Reprinted from (Chan, 2020).	14
2.3	Detailed schematic of the self-attention mechanism, illustrating the calculation of Query (q), Key (k), and Value (v) representations, followed by the softmax operation. Reprinted from (Exxact Corporation, 2023).	17
2.4	The Transformer model architecture, featuring stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. Reprinted from (Vaswani et al., 2017)	19
2.5	Tokenization output comparison for a Catalan text sample using FLAN-T5-base versus mT5, illustrating the inefficiency of English-centric vocabularies in cross-lingual environments.	24
3.1	Distribution of how much the summaries are compressed compared to the original texts. The 0.1 limit used to sort them by length is shown.	34
3.2	Examples of translations with repetition and hallucination errors in sequence-to-sequence models.	36
3.3	Proportional contribution of OPUS and NLLB models to dataset translation, categorized by language pair directional flow in the training split.	47
4.1	Architectural pipeline of the custom SplitEmbedding layer, illustrating the conditional routing, index adjustment, and mask-based vector reconstruction for vocabulary expansion.	52
4.2	Architectural data flow within the proposed SplitLMHead layer, showcasing the bifurcated projection of hidden states and their subsequent concatenation into a unified global logits tensor.	54
4.3	Step-by-step construction of the instruction block and the final concatenated string fed into the encoder.	55
4.4	Illustrative examples of multi-attribute conditioning setups. The figure demonstrates how various structural attributes are synthesized into specific natural language prompts tailored to instruct the model.	56
4.5	Probability distribution of the language selection mechanism and its corresponding target system tokens for multilingual conditioning.	58
4.6	Probability decision tree of the NER routing mechanism.	58
4.7	Probability decision tree for the summary length routing mechanism.	59
7.1	Architecture Diagram of the Text Simplification System	80

C.1	System prompt used for the action-driven controlled generation (<i>action-based</i>). It defines the available action space and the strict input format. . .	124
C.2	User prompt used for the action-driven controlled generation (<i>action-based</i>). It provides the global context and details each sentence paired with its specific simplification action.	125
C.3	System prompt used for the autonomous, action-free generation. It provides global simplification guidelines, relying entirely on the model’s autonomous capabilities.	126
C.4	User prompt used for the autonomous, action-free generation . It introduces the instructions and solely provides the complete source text. . . .	127
D.1	Evaluation prompt used by the LLM-as-a-Judge to assess the <i>Consistency</i> dimension. It assigns the model the role of an expert editor to evaluate the logical flow, structural organization, and narrative unity of the generated summary on a discrete 1-5 scale.	130
D.2	Evaluation prompt used by the LLM-as-a-Judge to assess the <i>Fluency</i> dimension. It assigns the model the role of an expert editor to evaluate the logical flow, structural organization, and narrative unity of the generated summary on a discrete 1-5 scale.	131
D.3	Evaluation prompt used by the LLM-as-a-Judge to assess the <i>Relevance</i> dimension. It assigns the model the role of an expert editor to evaluate the logical flow, structural organization, and narrative unity of the generated summary on a discrete 1-5 scale.	132
D.4	Evaluation prompt used by the LLM-as-a-Judge to assess the <i>Coherence</i> dimension. It assigns the model the role of an expert editor to evaluate the logical flow, structural organization, and narrative unity of the generated summary on a discrete 1-5 scale.	133
E.1	Evaluation prompt specifically formatted for the <i>M-Prometheus</i> model to assess the <i>Coherence</i> dimension. It directs the architecture to evaluate the structural organization and logical flow of the summary.	136
E.2	Evaluation prompt specifically formatted for the <i>M-Prometheus</i> model to assess the <i>Consistency</i> dimension. It restricts the evaluation strictly to factual alignment and the penalization of hallucinations.	137
E.3	Evaluation prompt specifically formatted for the <i>M-Prometheus</i> model to assess the <i>Fluency</i> dimension. It focuses the model’s analytical capabilities on linguistic quality, grammar, and naturalness.	138
E.4	Evaluation prompt specifically formatted for the <i>M-Prometheus</i> model to assess the <i>Relevance</i> dimension. It forces the model to weigh the presence of core facts against redundant or trivial information.	139
F.1	Instructional prompt provided to the Qwen3.6:27b model for the news summarization task.	142

List of Tables

3.1	Global structural statistics and extraction metrics per partition for the Newsroom dataset. Values for characters, words, sentences, and unique vocabulary denote the average per article and summary. TTR is a measure of vocabulary richness. The extraction metrics included in the original dataset are Comp., the ratio of article length to summary length, Cov., the proportion of summary words extracted directly from the article, and Dens., the average length of extractive word sequences.	30
3.2	Top 10 sources (domains) of the dataset distributed by partition, alongside totals and overall percentage. The numerical values represent the number of article-summary pairs. While the complete Newsroom dataset contains a total of 741 unique sources	30
3.3	Global statistics for the Dacsa corpus (Catalan variant) at the character, word, and sentence levels, grouped by anonymized source and partition. Sources marked with an asterisk are exclusively present in the <i>Test-NI</i> partition.	31
3.4	Global statistics for the Dacsa corpus (Spanish variant) at the character, word, and sentence levels, grouped by anonymized source and partition. Sources marked with an asterisk are exclusively present in the <i>Test-NI</i> partition.	32
3.5	Word ratio statistics for the different translation directions. The analysis is calculated over a total of 10,000,000 samples for each language pair using the TATOEBa dataset.	37
3.6	Top 10 news sources and frequency long-tail summary.	40
3.7	Global statistics for the final catalan corpus. Number of samples, average words in the original text, average words in the selected summaries, average number of named entities, average compression, and average similarity are shown, grouped by source and partition. Sources marked with an asterisk (*) appear exclusively in the <i>Test_NI</i> partition.	43
3.8	Global statistics for the final Spanish corpus. Number of samples, average words in the original text, average words in the selected summaries, average number of named entities, average compression, and average similarity are shown, grouped by source and partition. Sources marked with an asterisk (*) appear exclusively in the <i>Test_NI</i> partition.	45
3.9	Global statistics for the English corpus. Number of samples, average words in the original text, average words in the selected summaries, average number of named entities, average compression, and average similarity are shown, grouped by source and partition.	46
3.10	Source distribution (top 10 + OTHER) and overall metrics for the English <i>Test-NI</i>	46
5.1	Summary of the global training hyperparameters.	63
6.1	Human alignment results for the different evaluated models regarding the <i>SummEval</i> standard. Spearman (ρ) and Kendall (τ) correlation coefficients are reported for each dimension.	70
6.2	Lexical evaluation metrics for the custom token-based conditioning approach. Performance is reported on <i>test_i</i> and <i>test_ni</i> using ROUGE, SacreBLEU, METEOR, and ChrF variants.	70

6.3	Semantic and embedding-based evaluation metrics for the custom token-based conditioning approach. Performance is evaluated on <i>test_i</i> and <i>test_ni</i> using F1-scores from BERTScore, alongside BARTScore and COMET metrics.	70
6.4	Linguistic properties, entity preservation, and constraints evaluation for the custom token-based conditioning approach. Performance is measured on <i>test_i</i> and <i>test_ni</i> via NER recall, percentiles, language accuracy, and compression correctness.	71
6.5	Lexical evaluation metrics for the natural language sentence-based conditioning approach. Performance is reported on <i>test_i</i> and <i>test_ni</i> using ROUGE, SacreBLEU, METEOR, and ChrF variants.	71
6.6	Semantic and embedding-based evaluation metrics for the natural language sentence-based conditioning approach. Performance is evaluated on <i>test_i</i> and <i>test_ni</i> using F1-scores from BERTScore, alongside BARTScore and COMET metrics.	71
6.7	Linguistic properties, entity preservation, and constraints evaluation for the natural language sentence-based conditioning approach. Performance is measured on <i>test_i</i> and <i>test_ni</i> via NER recall, percentiles, language accuracy, and compression correctness.	72
6.8	Lexical evaluation metrics for the Qwen3.6:27b prompting approach. Performance is reported on standard and non-standard test splits (<i>test_i</i> and <i>test_ni</i>) using ROUGE, SacreBLEU, METEOR, and ChrF variants.	72
6.9	Semantic and embedding-based evaluation metrics for the Qwen3.6:27b prompting approach. Performance is evaluated on in-domain and not in-domain test splits (<i>test_i</i> and <i>test_ni</i>) using F1-scores from BERTScore, alongside BARTScore and COMET metrics.	73
6.10	Linguistic properties, entity preservation, and constraints evaluation for the Qwen3.6:27b prompting approach. Performance is measured on in-domain and not in-domain test splits (<i>test_i</i> and <i>test_ni</i>) via NER recall, percentiles, language accuracy, and compression correctness.	73
6.11	Qualitative evaluation using an LLM-as-a-Judge approach. Performance is measured across Coherence, Cohesion, Fluency, and Relevance for the <i>test_i</i> and <i>test_ni</i> data splits.	74
7.1	Examples of sentence-level and document-level simplification extracted from the Cochrane-auto corpus, illustrating the transformation flow from complex to simple text.	79
7.2	Metrics for SimpleText Document Simplification Dataset	86
7.3	Metrics for SimpleText Sentence Simplification Dataset	86
7.4	Metrics for SimpleText 2026 Test Documents	87
7.5	Metrics for SimpleText 2026 Test Sentences	87
A.1	Assessment of Sustainable Development Goals	103

CHAPTER 1

Introduction

In the last few years, the amount of information available to the society has increased exponentially and shows no signs of stopping. While this phenomenon could initially be perceived as a clear technological and social advantage, as it allows citizens to be better informed, it has also led to exhaustion caused by information overload (Graf & Antoni, 2022). To put this volume into perspective, it is estimated that the average 21st-century individual processes more information in a single day than an inhabitant of the Middle Ages consumed over a lifetime. Currently, hundreds of thousands of articles and news stories are published online daily, in addition to massive volumes of content on social media platforms such as YouTube, Instagram, and TikTok. Despite this, a significant part of society suffers a profound disconnection from the geopolitical and social reality of their surroundings, feeling unable to process and contextualize nearby events, which ultimately renders this increase in information useless (Goyanes et al., 2021; Heiss et al., 2023).

Given this scenario, one might think that the rise of generative Artificial Intelligence (AI), driven by models based on the Transformer architecture, is the ideal solution to facilitate access to quality information. Because it can process large amounts of information and provide a summary or explanation of varying quality, (Nie et al., 2025).

However, many users grant absolute reliability to AI-generated responses without questioning their veracity. This is particularly dangerous due to hallucinations, an inherent characteristic of these models, where entirely fabricated or false claims are produced with a highly convincing tone of certainty. For these reasons, in a scenario where rigor is required, it is necessary to guide the model to avoid hallucinations, (Huang et al., 2023).

In addition to the challenge of avoiding hallucinations, the vast amount of resources required by these models must be considered, making it impossible for a regular person to access them without relying on the major companies that control them. Furthermore, even if the necessary hardware resources are available, many of these models are protected as trade secrets, thus privatizing their use.

1.1 Motivation

The main motivation of this Bachelor's Thesis is the creation of an automatic conditioned text generation system. Instead of relying on massive and unpredictable generative mod-

els, which incur prohibitive computational costs and promote technological centralization, the intention is to condition text generation to ensure it meets the specific needs of the reader, narrowing the information to the desired context and thus minimizing the risk of hallucinations.

The desire to avoid using these enormous models is motivated by my participation in the Càtedres de l'Estratègia Nacional d'Intel·ligència Artificial de la UPV (Càtedres ENIA), an initiative of the Ministry of Economic Affairs and Digital Transformation within the framework of the European Recovery, Transformation and Resilience Plan, funded by the European Union-Next Generation EU (Universitat Politècnica de València, 2026). Among other objectives, these chairs strive for energy efficiency and advocate for the development of AI systems that are environmentally conscious and aligned with sustainable development goals.

Therefore, it is essential to discard the use of massive models that require huge server infrastructures operating continuously, which causes the consumption of exorbitant amounts of electrical energy and millions of liters of fresh water for cooling systems. This extremely high consumption of natural resources generates a large emission of greenhouse gasses, highlighting that the current "bigger is better" paradigm is unsustainable in the long term.

Beyond the reduction of the ecological footprint, the use of smaller models offers critical advantages in privacy. By being able to run locally, we avoid sending information to external servers, a standard practice among larger, proprietary models that harvest user data for subsequent retraining cycles.

Furthermore, the project aims to address the current linguistic asymmetry of these technologies. It is undeniable that the majority of AI models exhibit a significantly superior comprehension and proficiency in English relative to any other language, a bias that is even more accentuated in small models. This disparity is a direct reflection of the distribution of content on the web, as illustrated in Figure 1.1. For this reason, another key motivation is the evaluation of the system in a trilingual environment: English, Spanish, and Catalan. The inclusion of Catalan is particularly relevant in this study, as it allows us to verify the viability and behavior of the model in a language with lower volume of data, ensuring that such useful technologies do not marginalize my language.

On a personal level, I find this problem very interesting, mainly because its broad applicability: conditioned text generation systems, such as automatic text simplification, can be integrated across a wide range of devices and platforms to directly assist individuals in comprehending complex information. Furthermore, the technical challenge of optimizing smaller models to avoid relying on massive ones serves as a significant motivation.

1.2 Objectives

The main objective of this thesis is to develop and evaluate a controllable and computationally efficient conditioned text generation system, capable of offering competitive performance in conditioned generation compared to large language models (LLMs), taking into account computational, temporal, and data limitations, with special emphasis on

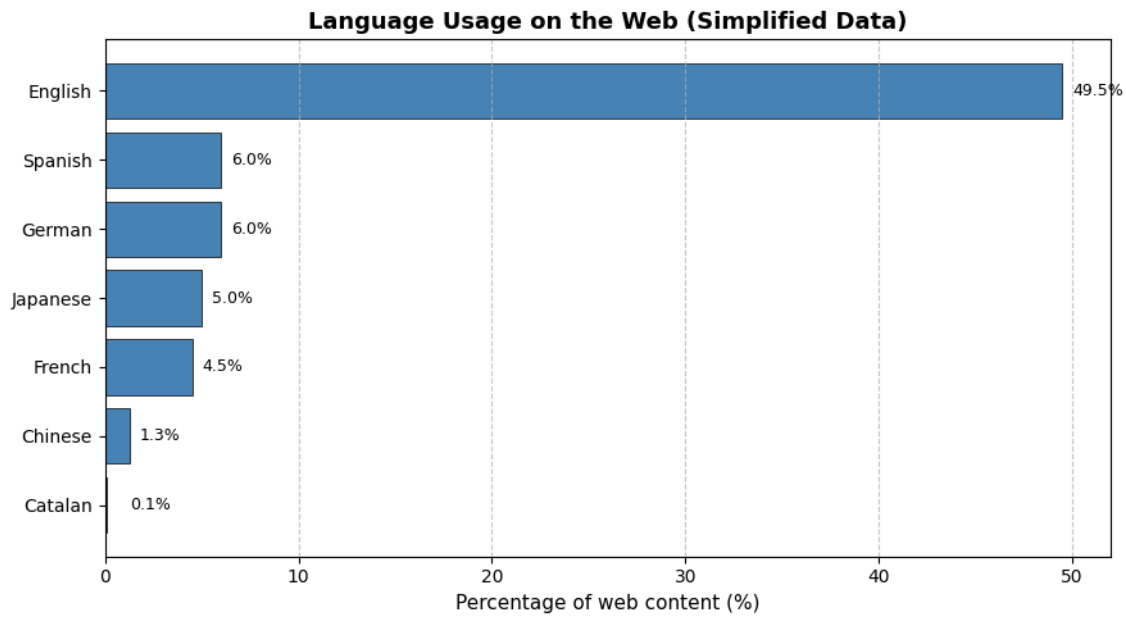


Figure 1.1: Percentage distribution of global web content language usage. Own elaboration based on data from (Q-Success / W3Techs, 2026).

its proper functioning in a trilingual environment (English, Spanish, and Catalan).

In order to achieve the general objective, the following specific objectives are proposed:

1. **Analyze the state of the art:** Study current Natural Language Processing (NLP) solutions for summary generation, contrasting massive auto-regressive models with lighter and more efficient architectures, in order to technically justify the choice of the optimal architecture for this project.
2. **Create and adapt a trilingual dataset:** Build and process a structured dataset to facilitate the instruction-tuning of conditional summarization models across English, Spanish, and Catalan, since optimal resources for this specific task do not currently exist.
3. **Train and optimize an efficient model:** Perform full fine-tuning or PEFT training on a reduced-parameter model, prioritizing local execution to ensure data privacy and a sustainable ecological footprint, aligning with the core principles of the ENIA Chairs.
4. **Evaluate hallucination reduction and conditioning:** Check the system's capacity and quality to strictly adhere to user instructions, establishing quantitative metrics for contextual accuracy and the suppression of irrelevant outputs.
5. **Evaluate linguistic asymmetry:** Analyze the performance of the solution in the three target languages, to determine its practical efficacy in lower-resourced languages (e.g., Catalan) against the established English baseline.
6. **Benchmark against state-of-the-art models:** Conduct a comprehensive qualitative and quantitative comparison (using standard metrics such as ROUGE, BLEU, BERTScore...) between the outputs of the proposed efficient system and those generated by contemporary, proprietary massive models.

7. **Participate in the CLEF 2026 SimpleText competition:** Validate the performance of the developed system and other small-sized models in an international and standardized evaluation environment, contrasting their efficacy, efficiency, and generation quality against the capabilities of LLMs.

1.3 Report Structure

The present Bachelor's Thesis is organized into eight main chapters and six appendices.

Chapter 2 reviews the State of the Art in Natural Language Processing (NLP). It traces the historical evolution of classical techniques up to the Transformer revolution, specifically focusing on the T5 model family and the paradigm shift toward Small Language Models (SLMs) for conditional text generation, concluding with a critical analysis and the formal proposal of this work.

Chapter 3 details the Data Preparation and Understanding phase. It covers the acquisition, translation, and enrichment of the datasets, followed by a rigorous cleaning process, the partitioning strategy, and the final structural statistics of the corpus.

Chapter 4 presents the Proposed Architectural Framework. This section explains the methodologies developed to condition the mT5 model, exploring both the adaptation of input and output layers via custom tokens (SplitEmbedding and SplitLMHead) and the use of natural language sentences.

Chapter 5 outlines the Training and Fine-Tuning Strategy, documenting the hardware environment, data augmentation techniques, general hyperparameters, and the application of Low-Rank Adaptation (LoRA).

Chapter 6 establishes the Evaluation Methodology. It contrasts quantitative evaluations utilizing traditional text quality metrics against an advanced qualitative LLM-as-a-Judge approach, ultimately leading to a comparative discussion for optimal model selection.

Chapter 7 summarizes the practical application of this system in the CLEF 2026 SimpleText Track, describing the system architecture, action-driven generation, and the results obtained in the competition.

Finally, Chapter 8 presents the final conclusions derived from the project and outlines potential avenues for future work.

Additionally, the document includes several appendices. Appendix A assesses the relationship of the thesis with the Sustainable Development Goals (SDGs). Appendix B contains the preprint of the SimpleText paper, while Appendices C, D E, and F compile the specific prompts utilized for the SimpleText track, the general LLM-as-a-Judge, the M-Prometheus evaluation, and the Qwen3.6 model, respectively.

1.4 Relationship with the Studies Undertaken

Without the studies and knowledge acquired throughout the degree, it would have been impossible to carry out a project of these dimensions. It is undeniable that this work is more deeply interconnected with a specific subset of courses.

First of all, the Python programming knowledge obtained from courses such as Programming Fundamentals (14002), Programming (14003), Data Structures (14008), and Algorithmics (14007) has proven indispensable. Furthermore, the broader field of Artificial Intelligence demands a rigorous statistical baseline, which was established through: Exploratory Data Analysis (14004), Statistical Models for Decision Making I (14005), and Statistical Models for Decision Making II (14006).

Moreover, the acquisition of Machine Learning knowledge with subjects such as: Scalable Techniques in Machine Learning (14009), Descriptive and Predictive Models I (14010), and Descriptive and Predictive Models II (14011). Also, the block of project-based subjects, such as Project I: Data Understanding (14019), Project II: Data Integration and Preparation (14020), and Project III: Data Analysis (14021), has provided a foundation of critical thinking and good practices for carrying out complex projects.

Finally, the subject of Visualization (14022) stands out. And without any doubt, the main subject for the realization of this Bachelor's Thesis is Natural Language and Information Retrieval (14029), as it provided the foundational techniques and concepts integral to Natural Language Processing.

CHAPTER 2

State of the Art

2.1 Introduction to Natural Language Processing (NLP)

Natural Language Processing (NLP) constitutes a fundamental interdisciplinary research area at the intersection of artificial intelligence, computer science, and computational linguistics. The main goal of NLP is to provide computer systems with the ability to process, understand, interpret, and generate human language in a way similar to how a person would. This discipline seeks to close the gap between human communication, which is inherently ambiguous and unstructured, and computational understanding based on formal representations and data. (Jurafsky & Martin, [2026](#))

Fundamentally, NLP is divided into two primary functional areas that operate complementarily, each presenting distinct computational challenges:

- **Natural Language Understanding (NLU):** This branch is responsible for interpreting and extracting meaning from text or speech. The primary objective of NLU is to resolve the inherent ambiguity of human communications, such as polysemy, irony, and double meanings enabling the system to infer user intent and structure input data. Applications like sentiment analysis, Named Entity Recognition (NER) and text classification belong to this category.
- **Natural Language Generation (NLG):** Conversely, NLG focuses on the ability to produce structured, coherent, and grammatically correct text from knowledge representations or non-linguistic data. This branch is responsible for generating intelligible responses, creating automated summaries (El-Kassas et al., [2021](#)), and translating complex data into human-readable language, just as current conversational models do.

2.2 Historical Evolution and Classical Techniques

While the recent technological boom and massive adoption of Natural Language Processing in recent years may give the impression of being a strictly current phenomenon, the scientific underpinnings of this discipline were established several decades ago. Understanding this evolution is essential to comprehend the steps that have allowed the technology to leave the laboratory and consolidate itself in the current mainstream paradigm (GeeksforGeeks, [2025](#)).

2.2.1. The Origins: The Era of Rule-Based Systems (1950s–1960s)

The first efforts to grant machines the ability to process human language date back to the 1950s, a period marked by post-World War II technological optimism. This optimism was initially engineered for ballistic calculations and cryptographic decryption, for example the Enigma machine. It was thought that these emerging "thinking machines," which had helped win a war, possessed the potential to resolve any intellectual challenge.

The beginning of this discipline can be situated with the publication of the article *Computing Machinery and Intelligence* by Alan Turing (1950). In this work, Turing proposed the "Turing Test," which is based on distinguishing between a conversation carried out by a human being and a machine. This test was subsequently regarded as the definitive benchmark for assessing artificial intelligence. This theoretical approach planted the seed for Natural Language Generation and Understanding, turning language into the main testing ground for intelligent computing.

Nevertheless, the main motivation during the early years of NLP had a geopolitical and military nature. In the context of the Cold War, governments (especially that of the United States) allocated substantial investments toward the automation of Machine Translation, specifically targeting the translation of Russian documents into English. This initial success generated disproportionate expectations about the possibility of solving the translation problem within a few years.

Els models desenvolupats durant aquestes dues primeres dècades es caracteritzaven per una aproximació purament simbòlica. Marcat per la falta de grans capacitats de còmput i dels conjunts de dades massius actuals. Aquests sistemes basats en regles (rule-based systems) no aprenien per si mateixos, sinó que eren construïts completament a mà. A partir de diccionaris bilingües i de complexes regles gramaticals codificades manualment per lingüistes i informàtics. Un exemple dels "models" basats en regles es ELIZA, un primerenc sistema conversacional que utilitzava tècniques de reconeixement de patrons i regles de substitució de paraules, per intentar superar en Test de Turing, (Weizenbaum, 1966)

Models developed throughout these initial two decades were defined by a strictly symbolic approach, marked by the lack of large computing capabilities and the massive datasets available today. Rather than possessing autonomous learning capabilities, these models were based on rules meticulously constructed by hand. They relied on bilingual dictionaries and intricate grammatical rules manually encoded by linguists and computer scientists. An example of rule-based "models" is ELIZA, an early conversational system that used pattern recognition methodologies and word-substitution rules to attempt to pass the Turing Test (Weizenbaum, 1966).

Despite these early developments, the rule-based approach presented a series of limitations due to the nature of human language, such as its extreme complexity, high dependence on context, grammatical exceptions, and, above all, semantic ambiguity. This methodological shortfall culminated in the release of the ALPAC (Automatic Language Processing Advisory Committee) report (Automatic Language Processing Advisory Committee (ALPAC), 1966). The findings empirically demonstrated that machine translation was more costly, considerably slower, and significantly less accurate than human translation. Generating a prolonged period of academic and financial stagnation.

2.2.2. Statistical NLP: The Rise of Data-Driven Models (1980s–2000s)

After a period of stagnation, the 1980s marked a resurgence in the field, motivated by significant advancements in computer processing power and hardware improvements, as well as the appearance of the first training corpora. A corpus constitutes a massive, structured collection of texts used to teach a machine learning algorithm to understand, interpret, analyze, and generate human language. Fundamentally, corpora are mainly divided into two types:

- **Unannotated Corpus (Raw):** This is pure, raw text (millions of web pages, books, Wikipedia articles).
- **Annotated Corpus:** This is text that has been categorized or marked, generally by humans, for a specific task. Typically by human annotators, to facilitate specific targeted tasks.

Nevertheless, the simple availability of these data is not enough, since machine learning algorithms require numerical inputs and are incapable of processing raw text directly. This necessity generated the development of techniques to transform words into discrete mathematical representations.

The most foundational representation is One-Hot encoding. It stands out because it does not evaluate the entire sentence at once, but rather translates it word by word (Bishop, 2006). This approach generates a vector whose length equals the size of the entire vocabulary, where all values are 0 except for a 1 in the exact position of the word. This method is extremely memory-inefficient (sparse vectors) and captures absolutely no semantic relationship between words.

The natural evolution of one-hot encoding is the Bag of Words (BoW). It was created with the intention of overcoming the inefficiency of representing text word by word to obtain a global representation of the document (Salton et al., 1975). A document is processed to compute the frequency of occurrence for each term present in the global vocabulary. It is fast and simple, but this method exhibits a critical shortcoming: it gives too much weight to very common words that do not provide real meaning, known as stop-words (such as articles or prepositions).

N-gram representation is an evolution of BoW designed to avoid completely losing the context of the sentence. Rather than tracking isolated words (unigrams), the vocabulary is built based on sequences of N contiguous words. This advancement enables the extraction of compound entities like "New York" or capturing negations like "not good." The primary drawback is that it causes the vocabulary size to grow exponentially due to the large number of possible combinations.

The main problem with both Bag of Words and the use of N-grams is that they give more importance to the most frequent words or combinations. In Catalan, words such as "el", "de", "la", or "i" will dominate any text, despite contributing negligible semantic information regarding the document's underlying topic. Term Frequency-Inverse Document Frequency was presented as a solution to this problem. First of all, two calculations are performed (Spärck Jones, 1972)

:

- **TF (Term Frequency):** The frequency of a word in a specific document (as in BoW).

$$\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

- **IDF (Inverse Document Frequency):** A measure that penalizes words appearing in many documents across the corpus, rewarding rare and specific terms. Logarithms are used to smooth the scale.

$$\text{IDF}(t, D) = \log \left(\frac{N}{|\{d \in D : t \in d\}|} \right)$$

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D)$$

Where t is the word we are evaluating and d is any document in the corpus. D . $f_{t,d}$ is the number of times the term t appears in document d . N : is the total number of documents in the corpus. Finally, $|\{d \in D : t \in d\}|$ is the number of documents containing term t . Often, a value of 1 is added to this term to avoid division by zero.

This is an excellent technique for keyword extraction and information retrieval (search engines). A word gets a high TF-IDF score if it appears often in one document but is rare in others. Despite the robustness of TF-IDF for keyword extraction, but it cannot understand synonyms. If two sentences mean the same thing but use different words, TF-IDF will think they are completely different. To fix this, we use Latent Semantic Analysis (LSA).

LSA uses a math technique called Singular Value Decomposition (SVD) on the TF-IDF matrix (Deerwester et al., 1990). Mathematically, it reduces the original dimension $|V|$ to a smaller, truncated k number of dense dimensions. This process projects documents into a space where axes no longer represent specific words, but underlying concepts. Due to this geometric transformation, document $d1$ changes from a long, mostly empty list of numbers into a short, compact dense vector (e.g., two-dimensional like $[0.85, -0.21]$), capable of being mathematically related to other documents with similar topics, creating the basis for modern models like neural embeddings.

All these techniques share the same basic goal: feature extraction. They are exclusively representation methods that translate human language into an algebraic format that a computer can process. Once these vectors are obtained (whether they are sparse like TF-IDF or dense like LSA), we use them as inputs (X) for traditional Machine Learning algorithms. The choice of the appropriate model depends directly on the linguistic task to be solved.

For simpler representations like Bag of Words, the Naïve Bayes classifier initially stood out (McCallum & Nigam, 1998). Thanks to its probabilistic nature, it adapted perfectly to frequency counts, rapidly calculating the category of a text based on the individual probabilities of its words.

As TF-IDF representation was adopted to improve feature extraction, a new challenge emerged: highly dimensional and extremely sparse vector spaces. In this scenario, Support Vector Machines (SVM) established themselves as the reference algorithm (Joachims, 1998). SVMs mathematically address this problem by finding the optimal hyperplane

that separates different classes, maximizing the margin between them, which makes them highly effective in sentiment analysis and complex text classification.

However, regardless of whether Naïve Bayes or SVM is used, all these classical models share a structural limitation: they treat text as a "bag" of elements, completely ignoring word order. While excellent for categorizing an entire document (e.g., spam detection), they are ineffective for sequence labeling tasks.

In Natural Language Processing, sequence labeling means giving a category to every single word in a sentence based on its context. Classic examples of this include Part-of-Speech (POS) tagging, where labels such as "Noun", "Verb", or "Determiner" are assigned to each word, and NER, where words are tagged as "Person", "Organization", or "Location". In these problems, language exhibits a profoundly sequential nature: the tag of a word (e.g., a noun) depends heavily on the tags of neighboring words (e.g., whether the preceding word was a determiner).

Classic examples are POS tagging, which labels words as "Noun," "Verb," or "Determiner," and Named Entity Recognition (NER), which identifies things like a "Person," "Organization," or "Location." In these tasks, language is highly sequential. The correct tag for one word (like a noun) often depends on the words right next to it (like if the word before it was a determiner).

At first, Hidden Markov Models (HMM) gained prominence (Rabiner, 1989). These generative models figure out a sentence's structure using two probabilities, the transition probability between tags (what is the probability that a Verb follows a Noun) and the emission probability (what is the probability of generating the word "bank" knowing that the current tag is "Noun").

Although pioneering, HMMs presented a fundamental limitation: due to their generative nature and the Markov property, they assumed that each word depended solely on its current tag, and that each tag depended solely on the immediately preceding tag. This strict rule ignored the overall context of the sentence (like words further ahead or behind) and drastically limited the inclusion of complex morphological features (such as whether the word started with a capital letter or what suffix it had).

To fix these limits, researchers created Conditional Random Fields (CRF) (Lafferty et al., 2001). CRFs are discriminative probabilistic models that, instead of modeling the joint probability, directly evaluate the conditional probability of the entire sequence of tags given the sequence of global observations at the sentence level, formulated as $P(Y|X)$. This approach allows for the incorporation of a practically unlimited set of arbitrary contextual features without assuming independence between them. CRFs use fast methods like the Viterbi algorithm to calculate the optimal tag sequence that maximizes this global probability. However, even though CRFs were great at learning language rules, finding the right features (called feature engineering) took a lot of hard manual work and expert knowledge. This problem motivated researchers to move towards architectures capable of learning representations automatically.

2.2.3. Deep Learning: The Revolution of Neural Architectures (2010s–Present)

As just mentioned, the biggest problem with traditional probabilistic models was that people had to manually extract features (feature engineering). This process took a lot of time and effort.

The Multilayer Perceptron and Feature Learning

The beginning of Deep Learning changed everything by introducing representation learning (Goodfellow et al., 2016). Instead of researchers doing it by hand, neural networks could now automatically learn how to find representations, patterns, and underlying dependencies in textual sequences.

This change happened mainly because of the Multilayer Perceptron (MLP). From a historical perspective, it should be noted that the mathematical foundations of neural networks and the backpropagation itself actually started in the 1950s and 1980s (Rumelhart et al., 1986). However, their practical application was severely limited for years due to computers were not powerful enough to handle the heavy calculations, and there was not enough data to train the models. The boom of these architectures occurred at the beginning of the 21st century, driven by the convergence of three critical factors:

- **Parallel computing capacity:** The adoption of GPUs allowed complex matrix calculations to be accelerated exponentially.
- **Availability of massive data:** The creation of huge digital corpora provided the volume of information necessary for deep models to generalize patterns without falling into overfitting.
- **Algorithmic improvements:** The use of new non-linear activation functions such as ReLU (Glorot et al., 2011) and efficient regularization techniques optimized gradient convergence.

A modern MLP is a feedforward network made up of an input layer, one or more hidden layers, and an output layer. Each neuron in a layer receives the outputs from the previous layer, linearly combined with a weight matrix (W) and a bias vector (b). The most important part of this architecture is applying a non-linear activation function (σ) to that combined result:

$$y = \sigma(W \cdot x + b)$$

These non-linear functions are essential for the proper functioning of the model. This is because the combination of multiple consecutive linear operations make the whole network act like one simple linear regression, preventing the resolution of problems that are not linearly separable, such as the XOR logical function. Thanks to these hidden layers and their non-linear functions, the MLP acts as a universal function approximator, marking the beginning of representation learning. An exemplified version of this explanations is shown in Figure 2.1

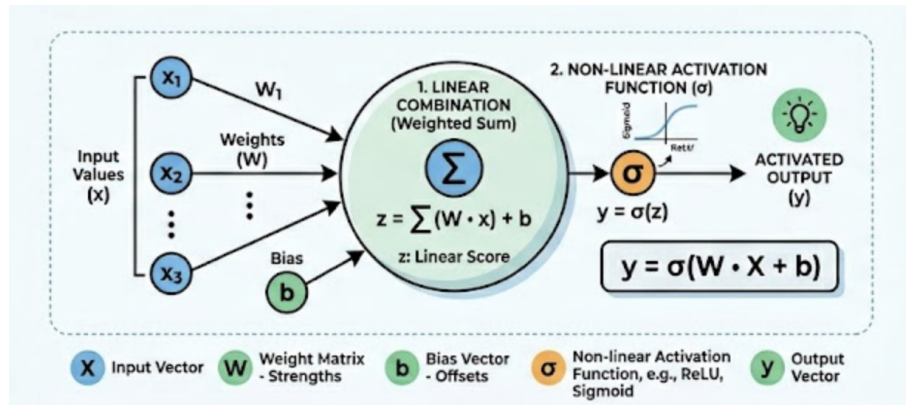


Figure 2.1: Schematic diagram of a Feedforward Neural Network with one hidden layer, illustrating forward propagation (black arrows) and the backpropagation path (dashed red arrows). Synthetic illustration generated with the assistance of Generative AI.

Word Embeddings: Dense Vector Representations

To enable this automatic learning on discrete data such as language, it is necessary to develop techniques to project the vocabulary into continuous mathematical spaces. This representation is called as "Word embeddings", which represent text as dense vectors where similar words are placed close to each other. For instance, if we take the vector representing "king", subtract the vector for "man", and add the vector for "woman", the resulting vector will be the representation of "queen":

$$\mathbf{v}_{\text{king}} - \mathbf{v}_{\text{man}} + \mathbf{v}_{\text{woman}} \approx \mathbf{v}_{\text{queen}}$$

The first major model for these vectors is known as Word2Vec (Mikolov et al., 2013). This model consists of a shallow neural network trained to generate embeddings from large text corpora. Before training, every single word in the vocabulary is turned into a basic code (usually using One-Hot Encoding) before being fed into the network so the network can read it. From this architecture, the model is optimized by predicting local contexts, resulting in two distinct training methods:

Continuous Bag-of-Words (CBOW): This architecture aims to predict a central word based on its context. The context words start as One-Hot vectors and go into a hidden layer, where the network calculates their average. From this continuous intermediate representation, the output layer evaluates and predicts the probability of the target word. By the prediction error, the model iteratively adjusts the network weights until it understands the language patterns properly.

Skip-gram: This is the exact opposite of CBOW, this strategy exclusively uses the central word as input to predict its context words. Because the architecture must deduce multiple context elements from a single discrete input, the optimization of the hidden layer weights is considerably more demanding. However, because of this characteristic, Skip-gram architecture is generally superior at understanding subtle meanings and modeling representations of infrequent words or highly technical terms.

Embedding extraction: The primary goal of this training is not the prediction task itself, but the optimization of the hidden layer's weight matrix. Once the network completes training, everything is discarded except the resulting intermediate weight matrix.

This matrix becomes the final embedding dictionary, transforming isolated discrete representations into a collection of dense mathematical vectors.

Another option besides Word2Vec is GloVe, created by researchers at Stanford University (Pennington et al., 2014). This model directly analyzes the statistics of the entire text, counting how many times each pair of words appears together across the whole corpus (global co-occurrence matrix). From these counts, the algorithm synthesizes this statistical information to generate the embeddings, efficiently capturing semantic relationships.

Modeling Sequentiality: Recurrent Neural Networks and LSTMs

The classical architecture of the Multilayer Perceptron (MLP) presented a severe structural limitation: it processes inputs independently and requires a fixed input length. This clashes with human language, which is inherently sequential and consists of sentences of variable lengths. Because of this, neural networks needed a way to have "memory."

This necessity drove the adoption of Recurrent Neural Networks (RNN) (Elman, 1990). Unlike feedforward networks, RNNs process the text sequentially one step (or word) at a time. At each time step t , the network takes as input the vector of the current word (x_t) and the hidden state from the previous step (h_{t-1}), generating a new hidden state (h_t). You can see how this step-by-step process works in Figure 2.2.

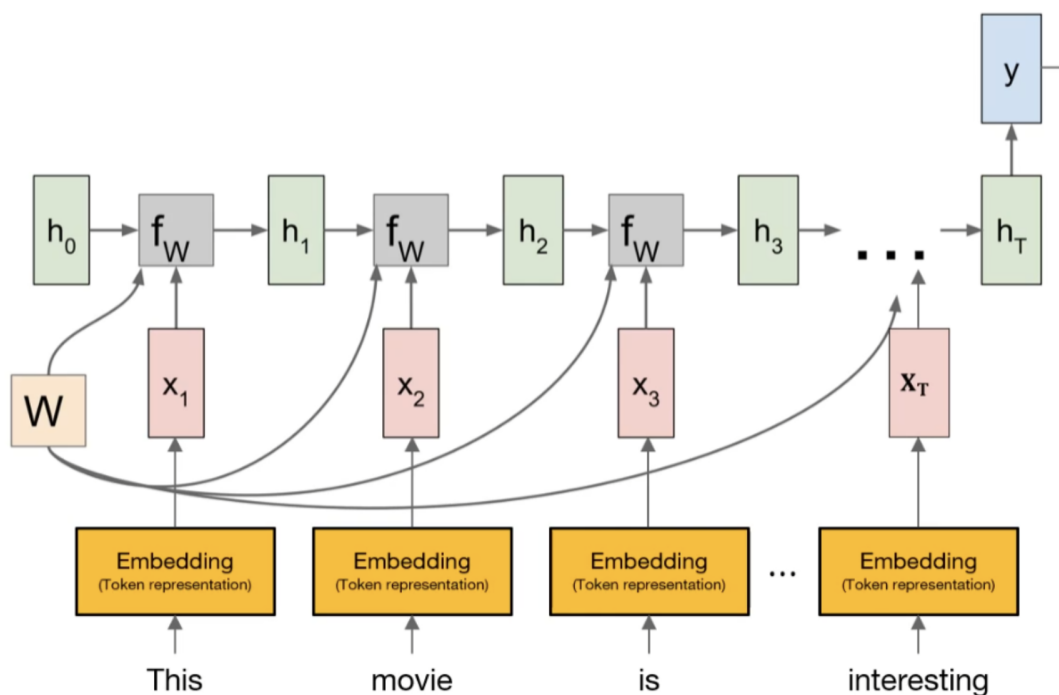


Figure 2.2: Architecture of a Many-to-One Recurrent Neural Network (RNN) processing a sequence of word embeddings. Reprinted from (Chan, 2020).

This looping setup lets the network carry information from past words forward, helping it understand the context. However, training standard RNNs using the backpropagation algorithm suffers from the vanishing gradient problem (Bengio et al., 1994). When sentences get too long, the gradients calculated to update the network weights tend to

shrink exponentially towards zero, which makes it impossible for the model to learn long-term dependencies.

To fix this math problem, researchers invented Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997). This architecture extends the RNN concept by incorporating a "cell state" (C_t) that carries long-term information. The flow of data is strictly regulated through non-linear gate mechanisms:

- **Forget gate:** Decides which old information is no longer needed and throws it away.
- **Input gate:** Decides which new facts from the current word are worth saving.
- **Output gate:** Uses the updated cell state to decide exactly what the network should output for the current step.

Thanks to this capacity to selectively retain or forget information, LSTMs consolidated themselves as the standard for years.

2.3 The Transformer Revolution

Even though combining Word Embeddings and LSTM networks substantially improved the performance in NLP tasks, the overall architecture still had a critical semantic bottleneck at the input layer. Representation models like Word2Vec and GloVe generate what are known as static or non-contextual embeddings.

In these models, every word gets just one fixed set of numbers regardless of the context in which it is used during the inference phase. This is a big problem for dealing with polysemy. For instance, the term "bank" will receive exactly the same numerical mathematical representation in the sentence "he deposited the money in the bank" as in "he sat on the river bank," even though it has a completely different meaning in each context.

Because the LSTM architecture can only process the sequence based on the initial geometric representation, assigning a static vector forces the model to use a messy representation that mixes all the possible meanings of the word. This showed that the network not only needed "memory" but also that the input embeddings themselves had to be dynamic and calculated in real time based on the full context of the sentence.

These limitations were overcome in 2017 with the introduction of the Transformer architecture (Vaswani et al., 2017). This model completely dispenses with recurrence and relies exclusively on attention mechanisms. This big change allowed two important things: total parallelization of data processing (taking full advantage of modern GPUs) and the creation of highly dynamic, contextual embeddings calculated in real time.

The mathematical core that makes this contextual understanding possible is the Self-Attention mechanism. Unlike a recurrent network that reads the sentence from left to right, self-attention allows each word in the sequence to simultaneously "observe" all the other words to figure out which ones help explain its own meaning.

2.3.1. Attention Calculation: Query, Key, and Values

Given an input text sequence formed by n words, the starting word embeddings form a matrix called X . The first critical step of the architecture consists of linearly projecting this input matrix into three different vector subspaces. It does this by multiplying X with three sets of weights (W_Q , W_K , and W_V) that will be adjusted and learned during the network's training phase:

$$Q = X \cdot W_Q$$

$$K = X \cdot W_K$$

$$V = X \cdot W_V$$

- **Query Matrix (Q):** This represents what a word is "looking for" in the rest of the sentence to understand its own context. It is the "question" asked by the evaluated element.
- **Key Matrix (K):** This holds the necessary information to answer the "questions" formulated by the queries of other words.
- **Value Matrix (V):** Contains the actual semantic content of the word. This meaning is only added to the final result if there is a strong match between a query and a key.

Once these three representations are obtained for the entire sequence, the attention operation is executed by sequentially applying four steps:

The first step is to calculate the dot product between the Query matrix and the transposed Key matrix ($Q \cdot K^T$). This operation evaluates the geometric similarity between each pair of words. If one word's Query and another word's Key point in the exact same direction (meaning the angle between them is very small), and maximizing their cosine similarity, their dot product will be high, indicating a strong syntactic or semantic dependency.

The result of this dot product produces a matrix of attention scores. Next, these scores are divided by the square root of the key dimension to ensure mathematical stability ($\sqrt{d_k}$). This step keeps the numbers mathematically stable. In high dimensions, dot products tend to grow dramatically in magnitude, which could lead to a vanishing gradient problem.

After that, the softmax function is applied independently over each row of the scaled matrix. This function converts the raw scores into strictly positive normalized weights, the sum of which is exactly 1. These weights determine the exact percentage of importance each context word deserves.

Finally, the attention weight matrix is multiplied by the Values matrix (V). The result is a weighted combination: words that have demonstrated greater semantic affinity (higher weight) will contribute a large portion of their information to the output vector, while the information from unrelated words will practically disappear. This iterative process can be visually summarized in Figure 2.3.

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

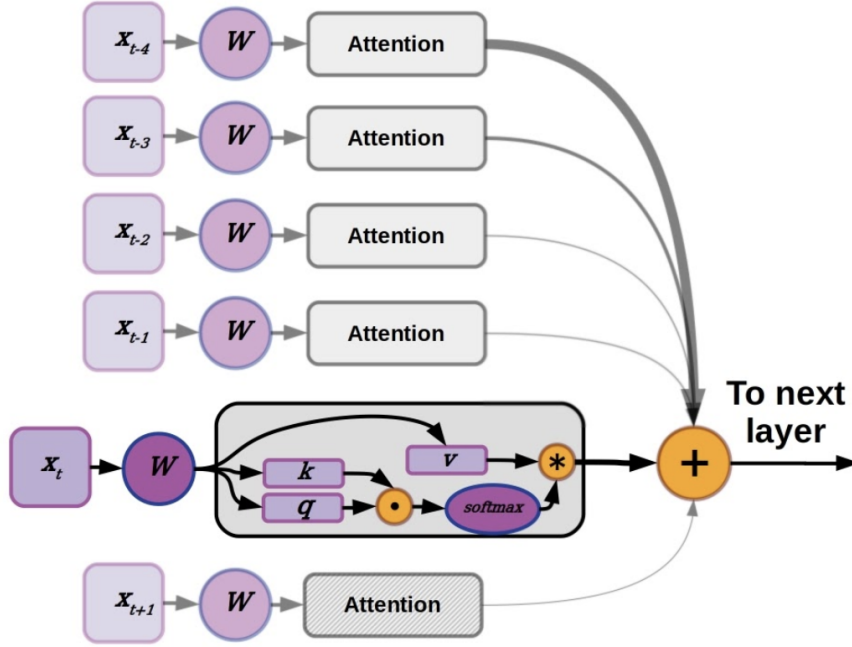


Figure 2.3: Detailed schematic of the self-attention mechanism, illustrating the calculation of Query (q), Key (k), and Value (v) representations, followed by the softmax operation. Reprinted from (Exxact Corporation, 2023).

To make the model even stronger, the Transformer architecture does not execute this calculation just once per layer. Instead, it applies what is known as Multi-Head Attention. Rather than applying attention over the vectors in their original dimensionality (d_{model}), the network linearly and independently projects the Queries, Keys, and Values into h reduced-dimension vector subspaces (typically $d_k = d_{model}/h$).

These parallel projections ensure that the network can focus on different linguistic phenomena simultaneously. For example, one "head" might focus on matching pronouns to their subjects, while another head specializes in the syntactic structure of verbs or overall meaning. Finally, the resulting vectors from all the heads are concatenated and projected again through a linear transformation to create the final output for the multi-attention layer.

2.3.2. Global Architecture: Encoder and Decoder Blocks

The original Transformer architecture is structured into two main specialized blocks: the Encoder and the Decoder. Although they were initially designed to work together. Later research showed that each block possesses unique properties that make them optimal for tasks of different natures.

The Encoder Block: Bidirectional Understanding

The Encoder is a module designed exclusively to process an input text sequence and extract a deep, highly contextualized mathematical representation from it. It works "bidirectionally," which means it looks at the whole sentence all at once. Allowing each word to be analyzed based on its full context (both preceding and succeeding words). Each layer inside the Encoder has two main parts:

- **Multi-Head Self-Attention:** Allows each word in the input sequence to analyze the rest of the words to extract context.
- **Position-wise Feed-Forward Network (FFN):** After attention, the resulting representation goes through a standard fully connected feed-forward network. It consists of two linear transformations separated by a non-linear activation function (usually ReLU or GELU). This step is vital since the attention mechanism alone is a fundamentally linear operation (based on weighted sums) so the FFN adds the "non-linearity" the model needs.

In addition, both parts are wrapped in "residual connections" (Add) followed by a layer normalization. Mathematically, the output of each sublayer is $\text{LayerNorm}(x + \text{Sublayer}(x))$. These residual connections allow gradients to flow directly through the deepest layers without vanishing, which is incredibly important when training huge networks.

Models that only use an Encoder are the standard choice for reading comprehension tasks. They are perfect for when the computer needs to read a whole text to make a decision, like sorting documents or figuring out the emotion (sentiment) behind a sentence.

The Decoder Block: Autoregressive Generation

On the other hand, the Decoder is a module designed specifically for sequential text generation. It works "autoregressively", meaning it predicts the next token based exclusively on the tokens that have already been generated previously.

To operate without being connected to an Encoder, the internal structure of a Decoder layer looks very similar to an Encoder, but with a critical difference in its first sub-module:

- **Masked Multi-Head Self-Attention:** Because the model has to learn to predict the future, a mathematical mask is applied (assigning negative infinite values prior to the softmax function). This mask strictly prevents any current position from "looking" at future positions in the sequence, avoiding information leakage during training.
- **Feed-Forward Network (FFN):** This is exactly the same as in the Encoder, also accompanied by residual connections and layer normalization.

The Complete Architecture (Encoder-Decoder)

Even though we just looked at the Encoder and Decoder separately, the original Transformer architecture was designed for sequence-to-sequence (Seq2Seq) tasks, such as machine translation. This requires a synergistic combination of both modules, where the

Encoder builds a deep understanding of the input text, and the Decoder uses that information to write the output one step at a time.

The architectural key to this union lies in a third sublayer introduced within the Decoder block, strategically located between the Masked Multi-Head Self-Attention and the Position-wise Feed-Forward network: the Cross-Attention mechanism. The full encoder-decoder architecture is shown in Figure 2.4

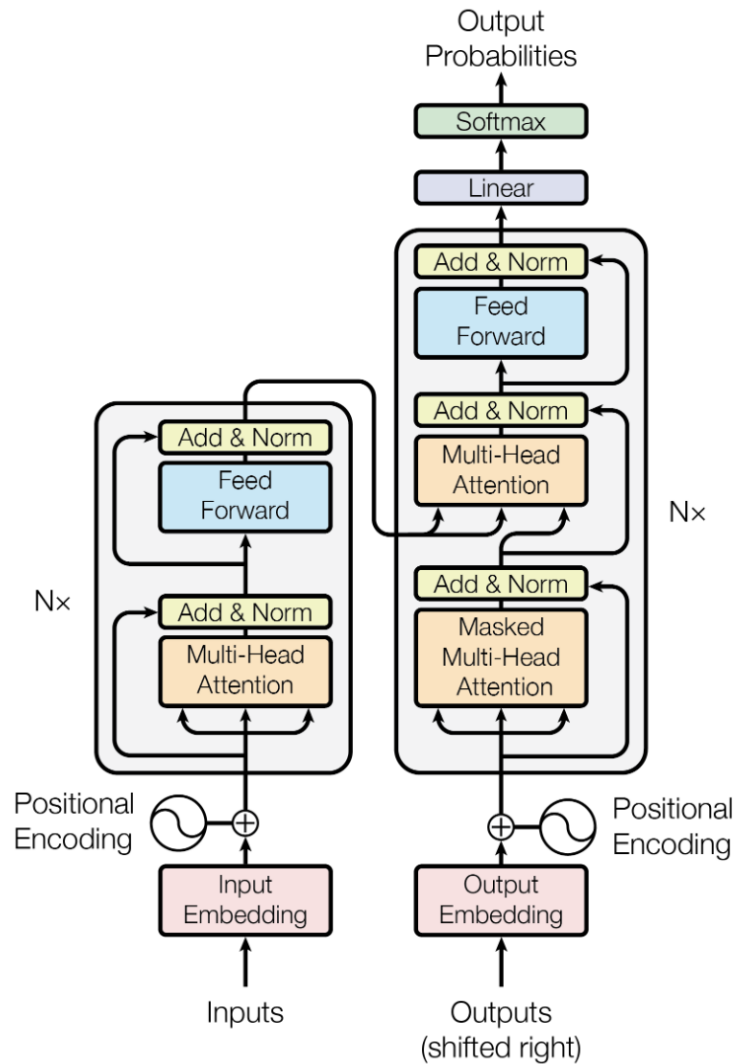


Figure 2.4: The Transformer model architecture, featuring stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. Reprinted from (Vaswani et al., 2017)

Unlike Self-Attention, where all matrices originate from the same sequence, Cross-Attention acts as an information bridge between the two modules.

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

Queries - Q: These originate from the previous sublayer of the Decoder itself (the output of the Masked Self-Attention). They represent the current state of generation; asking what information it "needs" to guess the next word.

Keys - K and Values - V : These come straight from the very end of the Encoder. The K and V matrices contain the highly contextualized representation of the entire original input sequence.

When we multiply the Decoder's Query matrix by the Key matrix, we get a scores. These scores, after being scaled and normalized, produce attention weights.

Finally, these weights are multiplied by the Encoder's Value matrix. This allows the Decoder to carefully "look back" at the most important parts of the original input sentence at the moment of generating each new token. Depending on the word being generated, the mechanism will assign more weight to certain input positions than to others (soft alignment).

It is important to note difference of the model when it is being used (inference) compared to when it is learning (training):

- **Inference:** The model operates completely autoregressively. The Decoder guesses a token, adds it to the output sentence and then, the entire set passes through the Decoder again to predict the next token. Using the prediction at time t as input to predict time $t + 1$, while continuously consulting the static representation from the Encoder.
- **Training:** If the premise that the output at $t - 1$ becomes the input at t were applied during learning, one early mistake would mess up the rest of the sentence, creating a snowball effect of errors. To mitigate this phenomenon, the architecture employs a technique known as Teacher Forcing. Instead of making the model use its own messy guesses, the input is directly replaced by the complete and correct target sequence. To make sure the model still learns to guess and doesn't just cheat by reading ahead, the target sequence is introduced in a single block but shifted one position to the right, preceded by a beginning-of-sequence token (<BOS>). This shift, working in conjunction with the causal mask of the Masked Self-Attention, guarantees that the calculation of the token at position i only has access to the correct preceding tokens ($< i$). In this way, learning is not hindered by previous bad predictions, achieving a parallelizable, faster, and much more stable training process.

2.4 The T5 Model Family: The Text-to-Text Paradigm

The T5 (Text-to-Text Transfer Transformer) model brought back the standard Encoder-Decoder architecture and revolutionized the field through the introduction of a paradigm called Text-to-Text (Raffel et al., 2020).

2.4.1. Text-to-Text

Unlike classical approaches that require modifying the network architecture and adding specific output layers depending on the nature of the problem, T5 unifies all NLP tasks under a single loss function and optimization method. Every single problem is simply rewritten as a "sequence-to-sequence" task (turning an input text into an output text).

To make this unification possible, T5 uses something called a "task prefix", which consists of adding a constant text string indicating the desired operation before feeding the text into the Encoder. For example, to translate a sentence, the input might be: "translate English to German: The cat sits on the mat". This instruction triggers the necessary attentional patterns for the Decoder to generate the text "Die Katze sitzt auf der Matte".

This design perfectly uses the Encoder-Decoder synergy, where the Encoder bidirectionally maps the input text into a continuous space of rich representations, and the Decoder uses Cross-Attention to generate the output textual sequence autoregressively.

2.4.2. Architectural Modifications of T5

Although T5 maintains an Encoder-Decoder structure, it introduces three optimizations to improve computational efficiency and large-scale stability.

Relative Positional Encoding

In the original architecture, the absolute position of each token is stored, assigning a fixed vector to each word that indicates its exact location. The problem with this approach is that the meaning of a sentence depends fundamentally on the proximity between words, rather than their absolute position. To fix this, T5 calculates the relative distance between the Query and Key tokens. In standard attention, the operation $q_i k_j^T$ evaluates the semantic similarity between two words. By adding the term $b_{i,j}$, T5 incorporates an additional weight based exclusively on their spatial distance $i - j$.

$$\text{Attention}(Q, K, V)_{i,j} = \text{softmax} \left(\frac{q_i k_j^T}{\sqrt{d_k}} + b_{i,j} \right) v_j$$

This implies that the network learns which distances are semantically relevant. For example, it might identify that an adjective is usually located at a distance of 1 or 2 positions from a noun, assigning this bias $b_{i,j}$ a high value and forcing the network to pay close attention to neighboring words.

In a perfect world with infinite computational capacity, the T5 attention mechanism would calculate and learn an independent scalar bias parameter ($b_{i,j}$) for every possible distance between any pair of tokens. However, given that in long sequences the number of unique distances grows exponentially, scaling this approach in real environments is computationally unfeasible and inefficient. To solve this, parameter and memory explosion, the model groups distances into *buckets*, which is structured into three levels of resolution:

- **Exact phase:** Words that are very close together each get their own *bucket*. This level of detail is fundamental for the network to correctly capture local syntactic and grammatical structure.
- **Grouped phase:** When words are more than 8 spaces apart, knowing the exact distance no longer matters as much. Therefore, distances are grouped; the further apart the words are, the larger the *bucket* containing them (e.g., distances between 12 and 15 might share the same positional weight).

- **Limit phase:** All words separated by a distance greater than a maximum threshold are placed in a single final *bucket*, treated generically as "very distant" words.

Thanks to this mechanism, from needing millions of numbers to calculate positions to using only 32 for each layer. This saving allows the architecture to efficiently process extremely long texts during the inference phase.

RMSNorm Normalization and Pre-LN Topology

With the aim of reducing the computational load, the standard Layer Normalization is replaced by a variant called Root Mean Square Normalization (RMSNorm) (B. Zhang & Sennrich, 2019). This method is based exclusively on the root mean square of the elements of the input vector $\mathbf{x} \in \mathbb{R}^d$:

$$\text{RMSNorm}(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2 + \epsilon}} \odot \gamma$$

This simplification eliminates sequential mathematical operations and decreases computation time. Additionally, a Pre-LN configuration is adopted, where normalization is executed immediately at the input of each attention and Feed-Forward block. This ensures a direct flow of gradients through the residual connections, preventing the vanishing gradient problem in deep architectures.

Bias-Free Architecture

The model removes extra additive numbers (biases) from all its linear layers, relying strictly on multiplying the weight matrices. This bias-free setup prevents the accumulation of constant numerical derivatives in large-scale models, which stabilizes optimization and reduces rounding errors when working in reduced floating-point precision formats (such as bfloat16 or float16).

2.4.3. Pretraining and the C4 Corpus

The authors developed the Colossal Clean Crawled Corpus (C4), a dataset of approximately 750 GB of clean text extracted directly from the web. To make sure the text was high quality, a purification process was carried out, including the removal of duplicated lines, the filtering of offensive text, and the elimination of source code.

To pretrain T5, instead of using BERT's classic Masked Language Modeling (MLM) (where individual tokens are replaced by a fixed mask element), T5 introduces a more efficient variant known as span-corruption. During this process:

1. The system randomly selects 15% of the words in the input text.
2. Consecutive tokens or entire fragments (*spans*) that have been selected are collapsed and replaced by a single special token (e.g., $\langle X \rangle$, $\langle Y \rangle$, $\langle Z \rangle$).

If the original sequence is "The black cat jumps over the large table", the corruption might transform the input to the Encoder into "The black cat $\langle X \rangle$ jumps over $\langle Y \rangle$ ". The Decoder's job is to output exactly the missing pieces: " $\langle X \rangle$ black $\langle Y \rangle$ the large table $\langle Z \rangle$ ". This trick drastically cuts down computational costs, as the target sequence length is

limited to the removed portions rather than the entire text, thereby decreasing training time.

2.4.4. Multilingual Version - mT5

While the original T5 was trained to understand English, French, Romanian, and German, Multilingual T5 (mT5) covers 101 languages, Catalan among them. This model almost exactly replicates the Encoder-Decoder architecture of T5, but it is trained on the mC4 corpus, which provides its extensive linguistic capabilities. The adaptation to multilingualism required two primary modifications compared to the original model (Xue et al., 2020):

1. Vocabulary increase: To efficiently represent such diverse languages without introducing bottlenecks, the model's vocabulary was increased from the 32,000 tokens of the original T5 to approximately 250,000 tokens.
2. Data sampling: Since the internet has exponentially more text in languages like English than in minority languages, training on the raw data would have hurt the performance of less exposed languages. To mitigate this issue, a sampling function is applied. This procedure artificially increases the sampling frequency of lower-resource languages during the pre-training phase, which prevents overfitting on high-resource languages and facilitates cross-lingual transfer.

2.4.5. Instruction Optimization: The FLAN-T5 Model

While the T5 architecture unifies NLP tasks, achieving optimal performance traditionally requires task-specific fine-tuning with thousands of examples for each new application. To overcome this limitation, researchers introduced the FLAN-T5 variant, or Fine-tuned Language Net, enhancing the base architecture through a technique known as instruction fine-tuning (Chung et al., 2024).

Instruction tuning involves an additional training phase on a corpus of academic tasks once the model has been pre-trained. Rather than merely providing isolated inputs and outputs to optimize the weights, the model is supplied with an explicit command (for example, "Read the following text and answer the question concisely: [text] [question]").

Being exposed to such a wide variety of commands during training changes how FLAN-T5 operates at inference time. This optimization gives the network a strong ability to generalize, enabling it to comprehend and execute direct demands without requiring additional task-specific training.

As a result, the model gains *zero-shot* (direct inference without any prior examples) and *few-shot* capabilities (inference provided with only a small set of reference examples within the input itself). This adaptability drastically cuts down the need for large annotated datasets for training and significantly boosts performance compared to the base T5 model.

Practical Limitations and Vocabulary in Public Implementations

Although the instruction-tuning methodology works for any language in theory, the official FLAN-T5 models (found in repositories like Hugging Face) present a critical architectural limitation: they are fundamentally restricted to the English language. This happens because these models reuse the same tokenizer and vocabulary from the original T5 model, which is limited to just 32,000 tokens.

Because of this, using FLAN-T5 directly to datasets in other languages (such as Catalan or Spanish) results in highly inefficient tokenization and a drastic degradation in performance. The examples in the Figure ?? illustrate the inability to obtain high-quality text in Catalan using FLAN-T5:

Example 1: Tokenization inefficiency due to vocabulary reduction

Original sentence: "Aquesta es una frase d'exemple en català"

FLAN-T5-base (Vocabulary: 32,128 tokens) Number of tokens: 20 Generated tokens: ['_A', 'quest', 'a', '_', 'e', 's', '_un', 'a', '_fra', 's', 'e', '_', 'd', '"', 'exemple', '_', 'en', '_cat', 'al', 'à']

mT5 (Vocabulary: 250,112 tokens) Number of tokens: 11 Generated tokens: ['_A', 'questa', '_es', '_una', '_frase', '_d', '"', 'exemple', '_en', '_c', 'atalà']

Figure 2.5: Tokenization output comparison for a Catalan text sample using FLAN-T5-base versus mT5, illustrating the inefficiency of English-centric vocabularies in cross-lingual environments.

This excessive fragmentation prematurely saturates the model's context window and dilutes local semantic meaning.

Secondly, and with more severe consequences, the small vocabulary lacks certain characters specific to Romance languages (such as open and closed accents, or the consonant *ñ*). When the tokenizer encounters one of these characters, it replaces it with the special <unk> (unknown) token. This ruins the text completely before the neural network even starts working with it. This design problem for non-English-speaking environments¹ can be reproduced through a simple encoding and decoding operation:

```
1 from transformers import AutoTokenizer
2
3 tokenizer = AutoTokenizer.from_pretrained("google/flan-t5-base")
4 output = tokenizer.decode(tokenizer.encode("Hola, bon matí."))
5
6 print(output)
7 #Hola, bon mat<unk>.</s>
```

Demonstration of information loss due to OOV characters

¹This architectural limitation is known and discussed by the developer community: <https://huggingface.co/google/flan-t5-large/discussions/8>

2.5 The Paradigm Shift in LLMs: The Evolution Towards Decoder-Only Networks

Despite the initial success of full Encoder-Decoder architectures, such as T5 and BART, across a wide variety of Natural Language Processing tasks, recent years have witnessed a paradigm shift. The field has transitioned from mid-sized models fine-tuned for specific tasks to massive, continuously scaling models with billions of parameters. This change has mostly relied on using architectures that only contain a decoder block. This paradigm shift has given rise to the most advanced model families currently available, such as OpenAI's GPT-5.2 series (OpenAI, 2026), Google's Gemini 3 (Gemini Team, 2026), and DeepSeek v4 (DeepSeek-AI, 2026).

The main reason for discarding the encoder block lies in the unification of NLP tasks under the next-token prediction framework. In Decoder-only models, both the input prompt and the generated text share the same space. The model processes the input through a causal attention mechanism that, as previously explained, applies a future-masking strategy. This simplifies the architecture by eliminating the need for separate encoding and decoding steps, as well as the cross-attention operations between them.

The decisive factor driving the Decoder-only architecture is its synergy with Scaling Laws (Kaplan et al., 2020). Tests show that the model's error rate, via loss, drops predictably as we increase the parameter count, the amount of training data, and the computing power. In terms of computational efficiency, the structural homogeneity of a Decoder-only network is significantly easier to parallelize and optimize on modern hardware (GPUs and TPUs), enabling massive training runs at scales of hundreds of billions of parameters with optimal resource utilization.

As a direct consequence of this massive scaling, these models have developed what researchers call "emergent abilities" (Zoph et al., 2022). One key example is in-context learning. This means the model can follow instructions and solve complex problems, like using Chain-of-Thought (Wei et al., 2022), just by reading the prompt. It can do this in zero-shot or few-shot scenarios without needing to be fine-tuned for every specific task.

2.5.1. Efficient Scaling: Mixture of Experts (MoE)

Even though Scaling Laws suggest that larger models perform better, making dense models encounters practical and economic limits due to the computational cost of inference. To overcome this barrier, some state of the art models have adopted sparse architectures based on the Mixture of Experts (MoE) paradigm.

In Sparse Expert Models, the standard feed-forward layers of the Transformer are replaced by multiple independent subnetworks known as experts. During inference, a routing mechanism picks a small subset of these experts to handle each token. This separates the total size of the model from the actual computing power it needs to run. Architectures such as Mixtral (Jiang et al., 2024) and DeepSeek leverage the MoE paradigm to deliver performance equivalent to that of massive dense models, yet requiring significantly less computational power and execution time.

2.5.2. The Shift Toward Small Language Models (SLMs)

In parallel to the race for massive models, there is a strong research focused on Small Language Models (SLMs). The prohibitive training costs and the reliance on massive data centers for inferencing hundred-billion-parameter models have pushed developers to create reduced alternatives.

These SLMs achieve highly competitive performance through techniques such as knowledge distillation (Hinton et al., 2015). More importantly, through the rigorous curation and filtering of high-quality training data, as demonstrated by (Gunasekar et al., 2023). The main goal of these models is to be highly efficient, enabling them to be deployed and executed locally or on mobile edge devices.

2.6 Conditional Text Generation

Conditional text generation is a subfield of NLP that covers a wide array of subtasks and domains. Unlike free-form generation, where the only goal is to maximize sequence coherence, conditional generation requires the model to incorporate specific information, adhere to certain semantic constraints, or modify attributes of the input text.

In practice, any interaction with a LLM that specifies rewriting rules, formatting, or style, what we are doing is a form of conditional generation. Within the scientific literature, several methodologies exist to compel or guide the model to comply with these constraints:

- **Prompting:** Because the recent evolution toward massive models based exclusively on Decoder-only architectures, the most widespread approach involves describing the constraint directly in natural language within the input context or prompt (P. Liu et al., 2023).
- **PEFT (Parameter-Efficient Fine-Tuning):** Instead of just using text instructions, a small number of additional parameters are fine-tuned to guide the model's behavior toward a specific task without massively altering the original network weights (Hu et al., 2021).
- **Constrained Decoding:** This technique intervenes directly in the search algorithm during the inference phase (e.g., through variations of Beam Search) to mathematically guarantee the inclusion or exclusion of certain tokens or entities (Post & Vilar, 2018).
- **Prefix-based Fine-Tuning:** A strategy historically employed by Encoder-Decoder models such as T5, where a task label (prefix) is embedded during full supervised training, thereby robustly associating the prefix with the desired transformation pattern (Ji et al., 2022).

It is important to point out that some rules are much harder to follow than others. A baseline condition, like asking for a summary of a text, establishes a global semantic transformation that current models resolve efficiently using simple instructions.

However, a significantly more demanding constraint, such as mandating a specific grammatical operation for each sentence, or imposing the sequential inclusion of particular entities or structures, creates major technical challenges. In these strict situations,

methods relying exclusively on prompting tend to exhibit instability. It is common for the model to disregard rules as generation length increases, or to generate hallucinations when attempting to reconcile multiple contradictory instructions.

2.7 State of the Art Critique

Even though the undeniable advancements of massive models, the current trend toward bigger and bigger networks presents structural, economic, and ethical limitations. In the research domain and in NLP competitions, such as the CLEF² and IberLEF³ forums, most developers use Large Language Models (LLMs) through private APIs. While it is true that these solutions achieve highly competitive performance, relying on them everywhere causes major problems.

Infinite scaling requires huge data center infrastructures and industrial-level computing resources. Making it impossible to run these models without APIs for most researchers, with the exception of the large corporations and organizations.

On the one hand, the reliance on external APIs demands the continuous transmission of data corpora to third-party servers. This is a major issue for private data and makes it impossible to guarantee that experiments can be reproduced. Due to the black-box scenario and access restrictions, the original training data and internal filtering mechanisms remain entirely hidden (Siddiq et al., 2025). On the other hand, using models with billions of parameters for small, specific tasks is massive computational overkill. This approach requires a huge consumption of electricity and water resources in data centers. Beyond the high costs, this paradigm directly conflicts with the principles of technological sustainability, drastically diverging from the objectives of green and environmentally responsible artificial intelligence (Jegham et al., 2025; Strubell et al., 2019).

From a strictly technical perspective, using these large models with basic prompting is not reliable for highly controlled text generation tasks. In scenarios demanding strict formal constraints, large LLMs tend to exhibit instability. It is common for the model to forget initial instructions or hallucinate when attempting to resolve complex or seemingly contradictory directives. This proves that basic natural language prompting is not strong enough for tasks that need very high precision (Junyan Wang & Xiaoqi Wang, 2025).

2.8 Proposal

This Bachelor Thesis have the goal to fill this technological gap by demonstrating the viability of efficient, locally executed solutions for complex conditional generation tasks. This project acts as a proof of concept. It proves that we do not need to rely on external APIs and black boxes to get accurate results in text simplification.

What makes this work different from the most common solutions today is its focus on sustainability and efficiency. While the state of the art shares the ultimate goal of achieving semantic coherence through prompting instructions, this thesis improves upon the current paradigm by proposing the strategic application of optimized, classic Encoder-

²<https://clef2026.clef-initiative.eu/>

³<https://sites.google.com/view/iberlef-2026/home>

Decoder architectures (such as the T5 family). Keeping the encoder block and using smaller models allows them to run completely locally. This approach not only ensures strict control over conditioning rules and absolute privacy of experimental data, but also drastically reduces the ecological footprint of the process, prioritizing a responsible and efficient use of artificial intelligence.

By carefully applying and fine-tuning efficient open-source models on local hardware, we can achieve high accuracy, reproducibility, and computational efficiency. Current general-purpose models simply cannot guarantee these features in such a controlled way.

CHAPTER 3

Data Preparation and Understanding

3.1 Dataset Creation

This section describes the methodology followed to construct the multilingual corpus. First, it details the acquisition and structural characteristics of the primary datasets (Dacsa and Newsroom). Finally, it explains the data augmentation process using machine translation models and the measures adopted to mitigate their potential biases.

3.1.1. Acquisition of Primary Datasets

The construction of the main dataset corpus is based on the Dacsa (Segarra Soriano et al., 2022) and Newsroom (Grusky et al., 2018) corpora.

Table 3.1 shows the overall structural statistics of the Newsroom dataset. The extreme compression ratio is particularly noteworthy: texts of approximately 680 words are reduced to summaries of about 28 words, yielding a ratio of 42 across all 3 partitions. Also, Table 3.2 shows the distribution of the 10 most representative sources in this corpus. Unlike the Dacsa corpus, which concentrates information within a limited number of sources, Newsroom uses 741 unique sources; despite this, a highly elevated percentage of the corpus originates from a small subset of these. For example, the top 3 sources account for 28.83% of the total dataset.

For the Dacsa dataset, these same structural statistics have been broken down at the source level in Table 3.3 for the Catalan variant and in Table 3.4 for the Spanish variant. Firstly, the Catalan subset includes 9 anonymized sources, 3 of which have been reserved exclusively for the Test_NI (Out-of-Domain) partition. This source distinction is crucial, as the writing style of each source can influence the final outcome. For instance, the CA08* source, used solely for Test_NI, features the longest texts but presents a remarkably low Type-Token Ratio (TTR) (Richards, 1987) of 0.43 for the articles, significantly lower than the approximate mean of 0.52, indicating structural vocabulary repetition. Additionally, the length stability of the summaries for the Train sources is notable, as they consistently average around 22 words. On the other hand, the Spanish partition exhibits a higher volume of samples and domains, comprising a total of 21 sources, 8 of which are allocated to Test_NI.

This segregation between Test_I (In-Domain) and Test_NI (Not In-Domain) is a fundamental aspect in the experimental design, as it allows validating the system’s generalization capabilities when evaluated on linguistic styles and domains not observed during the training phase. These substantial differences between sources can be observed, for example, in the Dacsa Spanish partition, which exhibits high heterogeneity, with texts ranging from highly synthetic articles of 375 words (ES11*) to massive documents of 1,641 words (ES17*).

Split	Article					Summary					Metrics		
	Chars	Words	Sents	Vocab	TTR	Chars	Words	Sents	Vocab	TTR	Comp.	Cov.	Dens.
Train	3960.6	681.7	38.9	310.5	0.535	161.8	27.6	1.5	24.2	0.920	43.900	0.825	9.483
Validation	3928.1	676.3	38.6	309.3	0.536	163.7	28.0	1.5	24.4	0.920	42.828	0.825	9.613
Test	3920.1	674.7	38.5	308.9	0.536	163.3	27.9	1.5	24.3	0.920	42.135	0.824	9.501

Table 3.1: Global structural statistics and extraction metrics per partition for the Newsroom dataset. Values for characters, words, sentences, and unique vocabulary denote the average per article and summary. TTR is a measure of vocabulary richness. The extraction metrics included in the original dataset are Comp., the ratio of article length to summary length, Cov., the proportion of summary words extracted directly from the article, and Dens., the average length of extractive word sequences.

Source Domain	Train	Validation	Test	Total	% of Total
nytimes	122,633	13,338	13,508	149,479	12.33
washingtonpost	92,503	9,631	9,799	111,933	9.23
foxnews	72,612	7,776	7,746	88,134	7.27
theguardian	58,030	6,374	6,272	70,676	5.83
nydailynews	55,443	6,040	5,884	67,367	5.55
cnn	33,846	3,618	3,658	41,122	3.39
time	32,323	3,670	3,608	39,601	3.27
mashable	31,893	3,414	3,315	38,622	3.18
people	27,466	3,347	3,229	34,042	2.81
wsj	27,601	3,118	3,075	33,794	2.79

Table 3.2: Top 10 sources (domains) of the dataset distributed by partition, alongside totals and overall percentage. The numerical values represent the number of article-summary pairs. While the complete Newsroom dataset contains a total of 741 unique sources

3.1.2. Dataset Translations

Once the data was acquired, we translated the summaries to get summaries of the articles in the three languages. This procedure consisted of translating each sample into the two remaining languages. For the Dacsa subsets, in both Catalan and Spanish, the multilingual version available in the official repository of the ELiRF research group¹ was utilized. Conversely, for the Newsroom corpus, a machine translation process was conducted using Deep Learning models. These translations were performed employing Helsinki-NLP models, specifically from the OPUS-MT family (Tiedemann & Thottingal, 2020). These are the same models used by the ELiRF team to translate the dacsa dataset.

Machine translation can introduce biases derived from the source model’s architecture and training data. It could lower the quality of our corpus and affect our final summarizer model. To mitigate this effect, we decided to incorporate a second translation.

¹<https://huggingface.co/datasets/ELiRF/dacsa-multilingual-summaries>

Source	Split	Samples	Article					Summary				
			Chars	Words	Sents	Vocab	TTR	Chars	Words	Sents	Vocab	TTR
CA01	train	226,441	2505.7	444.1	18.8	213.1	0.522	122.8	21.8	1.2	19.6	0.917
	validation	5,896	2506.3	445.0	18.8	213.4	0.523	122.4	21.8	1.2	19.6	0.918
	test_i	5,896	2511.7	445.5	19.0	213.7	0.522	123.6	22.0	1.2	19.8	0.915
CA02	train	182,905	2808.0	495.3	21.6	244.6	0.524	132.1	23.2	1.3	20.8	0.911
	validation	5,896	2804.9	494.9	21.7	243.7	0.523	131.9	23.1	1.3	20.8	0.912
	test_i	5,896	2793.4	492.6	21.4	242.2	0.524	131.6	23.1	1.3	20.7	0.911
CA03	train	125,655	2428.0	432.3	16.6	206.6	0.515	128.3	22.8	1.2	20.5	0.912
	validation	5,896	2457.8	437.5	16.8	208.5	0.514	128.9	22.9	1.2	20.5	0.912
	test_i	5,896	2442.1	434.5	16.7	207.1	0.515	127.6	22.7	1.2	20.4	0.914
CA04	train	45,035	2273.4	407.5	16.4	194.3	0.523	112.8	20.2	1.3	18.3	0.920
	validation	5,896	2276.1	407.8	16.2	194.5	0.522	112.6	20.2	1.3	18.3	0.919
	test_i	5,896	2298.3	412.0	16.5	195.3	0.521	113.6	20.3	1.3	18.4	0.920
CA05	train	32,589	3201.4	560.1	25.5	257.6	0.512	145.7	25.5	1.2	22.1	0.902
	validation	5,896	3230.0	565.3	25.7	259.2	0.510	146.0	25.5	1.2	22.1	0.902
	test_i	5,896	3180.6	557.0	25.5	255.6	0.514	147.0	25.7	1.3	22.2	0.901
CA06	train	23,971	2604.4	449.0	13.5	213.2	0.508	132.4	22.8	1.1	20.3	0.904
	validation	5,896	2611.2	450.0	13.4	213.5	0.508	134.1	23.1	1.1	20.5	0.903
	test_i	5,896	2562.1	441.3	13.1	210.2	0.508	132.2	22.7	1.1	20.2	0.904
CA07*	test_ni	7,104	2836.9	499.9	20.4	207.7	0.485	148.7	26.3	1.1	23.2	0.899
CA08*	test_ni	5,882	8471.5	1476.2	71.5	573.1	0.423	348.2	59.2	2.6	46.6	0.820
CA09*	test_ni	4,850	2887.6	509.1	25.1	228.5	0.534	178.0	30.9	1.2	26.5	0.882

Table 3.3: Global statistics for the Dacsa corpus (Catalan variant) at the character, word, and sentence levels, grouped by anonymized source and partition. Sources marked with an asterisk are exclusively present in the *Test-NI* partition.

Consequently, two synthetic variations are obtained for each original summary, resulting in a total of 5 summary per sample (the summary in the original languages plus two different translations for the other two languages). This secondary translation was executed using Meta’s No Language Left Behind (NLLB) model (NLLB Team et al., 2022), and it was applied to both the Dacsa and Newsroom partitions.

3.2 Dataset Conditioning and Enrichment

The primary objective of this work is to control the characteristics of the generated text by conditioning the model on three specific factors: the target language of the summary (regardless of the source language), the desired length, and the inclusion of specific named entities. To achieve this objective, a conditional generation strategy was implemented. This paradigm requires that, during the training phase, the model processes not only the source document but also a set of constraints or control attributes that guide its response.

Consequently, to enable this level of control, a preliminary enrichment of the dataset is essential. The original dataset has been enriched through the extraction of these three fundamental metadata features for each sample: the target language, the length, and the relevant Named Entities. This information has been stored as additional attributes associated with each document-summary pair. The procedure for obtaining these values is explained in the following sections:

Source	Split	Samples	Article					Summary				
			Chars	Words	Sents	Vocab	TTR	Chars	Words	Sents	Vocab	TTR
ES01	train	534,140	4023.0	674.5	35.9	314.6	0.501	144.1	24.3	1.5	20.8	0.895
	validation	8,004	4078.6	684.7	35.9	318.4	0.499	144.2	24.3	1.5	20.8	0.894
	test_i	8,004	4019.1	673.6	35.9	314.6	0.501	144.7	24.4	1.5	20.9	0.896
ES02	train	326,037	2700.3	455.4	19.7	224.0	0.517	127.2	21.6	1.1	19.2	0.898
	validation	8,004	2689.7	453.9	19.7	223.4	0.516	127.0	21.6	1.1	19.2	0.898
	test_i	8,004	2717.5	458.4	19.9	225.2	0.515	127.5	21.6	1.1	19.2	0.898
ES03	train	180,402	2508.3	423.8	17.1	208.6	0.524	112.8	19.2	1.1	17.3	0.914
	validation	8,004	2533.7	428.2	17.3	209.5	0.523	112.7	19.2	1.1	17.3	0.914
	test_i	8,004	2508.2	424.1	17.3	208.5	0.524	112.1	19.1	1.1	17.2	0.914
ES04	train	152,057	3289.9	552.8	25.5	255.8	0.512	130.1	22.0	1.2	19.6	0.903
	validation	8,004	3289.3	552.0	25.2	256.7	0.511	131.4	22.1	1.2	19.7	0.902
	test_i	8,004	3278.0	550.4	25.4	255.5	0.514	130.4	22.0	1.2	19.6	0.903
ES05	train	132,045	3710.8	630.3	30.2	289.3	0.496	159.0	27.1	1.6	23.5	0.882
	validation	8,004	3655.2	620.7	29.6	285.6	0.497	157.9	26.9	1.6	23.3	0.883
	test_i	8,004	3732.7	634.1	30.5	290.5	0.496	158.5	27.0	1.6	23.3	0.880
ES06	train	100,553	4303.9	717.0	28.4	316.8	0.473	303.7	50.0	1.4	39.1	0.809
	validation	8,004	4333.5	721.6	28.7	318.7	0.473	303.6	50.0	1.4	39.2	0.809
	test_i	8,004	4324.6	720.4	28.4	317.8	0.472	305.6	50.3	1.4	39.3	0.808
ES07	train	91,154	3541.7	588.7	22.5	269.5	0.491	178.0	29.6	1.4	25.1	0.871
	validation	8,004	3574.3	593.8	22.8	271.5	0.491	177.4	29.5	1.4	25.1	0.871
	test_i	8,004	3527.5	586.2	22.4	268.0	0.493	177.7	29.5	1.4	25.1	0.871
ES08	train	83,090	3468.1	586.0	27.0	273.2	0.497	123.0	20.8	1.3	18.5	0.907
	validation	8,004	3490.7	589.7	27.2	274.0	0.496	123.3	20.8	1.3	18.4	0.906
	test_i	8,004	3457.1	584.5	26.6	271.5	0.496	123.8	20.9	1.3	18.6	0.905
ES09	train	65,939	2760.6	464.7	17.6	221.4	0.506	124.6	21.2	1.1	18.8	0.903
	validation	8,004	2759.9	464.3	17.6	221.1	0.507	125.3	21.3	1.1	18.9	0.902
	test_i	8,004	2753.4	463.3	17.5	221.1	0.507	124.9	21.2	1.1	18.8	0.903
ES10	train	58,016	4027.7	677.0	32.4	296.9	0.516	142.0	23.9	1.4	21.0	0.893
	validation	8,004	4301.1	722.4	34.2	304.5	0.515	142.2	23.9	1.4	21.0	0.894
	test_i	8,004	4152.7	697.4	32.9	299.9	0.515	141.8	23.9	1.4	21.0	0.893
ES11*	test_ni	70,193	2275.4	375.6	12.6	185.9	0.533	297.0	49.1	1.4	38.1	0.791
ES12	train	41,227	2602.7	438.0	17.8	216.0	0.523	125.5	21.1	1.3	18.7	0.908
	validation	8,004	2592.8	436.4	17.8	215.8	0.523	125.1	21.1	1.3	18.7	0.908
	test_i	8,004	2598.4	437.8	17.8	215.8	0.523	124.8	21.0	1.3	18.6	0.907
ES13	train	19,155	3048.3	511.6	21.1	237.7	0.492	128.9	21.8	1.2	19.3	0.903
	validation	8,004	2991.8	502.5	20.5	234.6	0.493	128.5	21.8	1.2	19.3	0.903
	test_i	8,004	3017.0	506.8	20.8	235.7	0.492	128.2	21.8	1.2	19.2	0.902
ES14	train	19,104	4220.5	712.7	32.9	329.1	0.492	181.6	30.2	1.3	25.6	0.864
	validation	8,004	4256.4	718.8	33.0	331.1	0.491	182.4	30.3	1.3	25.6	0.863
	test_i	8,004	4256.2	719.4	33.3	331.5	0.491	180.7	30.1	1.3	25.4	0.864
ES15*	test_ni	17,379	3076.5	512.9	19.1	236.5	0.497	263.5	44.5	2.3	31.7	0.715
ES16*	test_ni	16,965	4361.0	723.9	31.5	313.6	0.462	145.9	24.5	1.2	21.4	0.887
ES17*	test_ni	2,450	10014.7	1641.0	82.8	627.2	0.440	457.5	75.1	3.4	54.3	0.763
ES18*	test_ni	1,374	2471.6	415.5	19.1	204.0	0.529	310.4	52.3	2.1	40.1	0.818
ES19*	test_ni	643	3280.7	548.3	19.5	248.3	0.486	85.6	14.7	1.1	13.6	0.935
ES20*	test_ni	467	2646.8	446.5	20.8	220.1	0.523	158.9	26.9	1.3	22.5	0.880
ES21*	test_ni	155	6848.6	1147.0	48.4	420.8	0.493	219.8	38.4	2.1	31.0	0.814

Table 3.4: Global statistics for the Dacsa corpus (Spanish variant) at the character, word, and sentence levels, grouped by anonymized source and partition. Sources marked with an asterisk are exclusively present in the *Test-NI* partition.

3.2.1. Language Determination

Language constitutes the primary control signal for the model. Given that the dataset contains samples in multiple languages (both originally and via translation), a categorical attribute has been generated for each instance. During the inference phase, the initial conditioning will be determined by the target language. For this reason, the original language must first be established:

- For data originating from known sources (Dacsa, Newsroom), the tag corresponding to the subset (CA, ES, EN) has been assigned.
- For machine-translated data, the target language tag of the utilized translation model (NLLB or Helsinki-NLP) has been propagated.

3.2.2. Length Categorization

To provide the model with the ability to generate summaries of varying lengths according to user needs, we calculated the compression percentage for each text. Given the complexity of establishing an absolute threshold to define what constitutes a short or long summary (for instance, a 100-word summary may be considered extensive for a 200-word article but extremely brief for a 10,000-word document), a relative metric was adopted.

Therefore, we set a threshold of 0.1. If the ratio is equal to or less than this value, the instance is categorized as a short summary; if it is greater, it is classified as a long summary. Thus, for an original document of 400 words, a 50-word summary (a ratio of 0.125) would be considered long.

The determination of this threshold comes from the goal of using length as a control feature. Initially, the intention was to segment the data into three categories: short, medium, and long summaries. However, the analysis of the compression distribution shown in Figure 3.1 reveals a really long tail. The first approach was to use percentiles as thresholds (with limits at 0.04 and 0.17); nevertheless, this method generated excessively brief short summaries and insufficient distinction between categories, so we dropped this idea. The mean compression stands at 0.08, but the final threshold was deliberately shifted to 0.1. This decision establishes a clearer cutoff value and allows for a slight increase in the length of summaries deemed short, ensuring adequate variability and a reasonable balance of samples between categories.

3.2.3. Named Entity Extraction

The third and hardest conditioning factor is the inclusion or not of named entities. The objective is to supply the model with an entity (be it a person's name, an organization, or a location) so that the model is capable of integrating it into the generated text. For the identification and extraction of relevant entities, we used a NLP library called *spaCy* (Honnibal et al., 2020), which contains specific pre-trained models for this task.

We chose the most appropriate model for each language based on availability and the underlying architecture. Whenever feasible, models based on *transformers* (TRF architecture) were prioritized in order to maximize precision in entity detection:

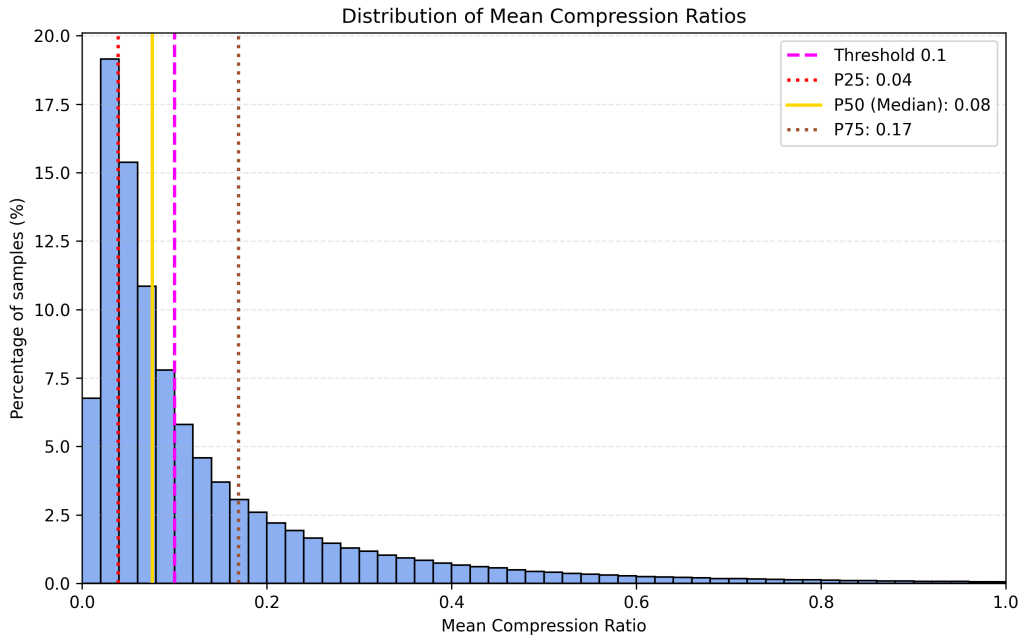


Figure 3.1: Distribution of how much the summaries are compressed compared to the original texts. The 0.1 limit used to sort them by length is shown.

- **Catalan:** The `ca_core_news_trf`² model was employed, which is based on the *RoBERTa-base-ca* architecture (Y. Liu et al., 2019a). According to the authors, this model achieves an F_1 score of 0.92 on the NER task.
- **Spanish:** The `es_core_news_lg`³ model was utilized, yielding an F_1 score of 0.90 on the NER task, as reported by the authors.
- **English:** The `en_core_web_trf`⁴ model was used, which is based on the *RoBERTa-base* architecture and, according to the authors, obtains an F_1 score of 0.90 on the NER task.

It is worth noting that for the Spanish variant, the *transformer*-based version was not utilized. This happened because of software bugs and compatibility issues inside the tool that caused execution errors. Conversely, the Catalan and English models functioned correctly without the need for any additional modifications.

The identification and extraction of entities are performed on both the source text and the summaries across all three languages. Because of this, for each text-summary pair, six lists are generated (one for the source article, one for the summary in the original language, and two for each language different from the original one) following the format: [(Entity1, Type1), (Entity2, Type2), ...]. To ensure that the model correctly learns the control relationship, it is strictly necessary that the specified entities appear in both the source text and the generated summaries. To guarantee this, a filter is applied by computing the intersection of the entity sets from the source text and each of the summaries.

²Official documentation and download: https://spacy.io/models/ca#ca_core_news_trf

³Official documentation and download: https://spacy.io/models/es#es_core_news_lg

⁴Official documentation and download: https://spacy.io/models/en#en_core_web_trf

Finally, this process gives us five lists, one for the original summary and the , containing the valid entities. The technical implementation detailing how these attributes are integrated into the architecture to guide the generation process will be comprehensively explained in Chapter 4.

3.3 Dataset Cleaning

In the training and fine-tuning of any machine learning model, especially in natural language processing, data quality directly determines the reliability and the good results of the model. This principle is known as Garbage In, Garbage Out (GIGO) and dictates that a high-quality dataset acts as a clear helps the architecture learn precise linguistic patterns, generalize correctly, and avoid hallucinations. Conversely, utilizing a deficient dataset with noise, asymmetries, or alignment errors causes the model to internalize and reproduce these very defects. The advantage of a robust dataset is not solely faster convergence during training, but the guarantee that the generated texts will be coherent, so the model does not waste time trying to understand garbage data.

Therefore, once we had all the data, we must ensure its quality. The first step consisted of conducting a preliminary manual inspection, analyzing the translated summaries and the verification of the correct identification of the entities. During this investigation, it was observed that some translations exhibited severe deficiencies; one of the primary errors was the infinite repetition of certain words or phrases.

Examples such as those shown in Figure 3.2 evidence the need to apply a multi-phase evaluation process to discard poorly translated samples. This procedure is structured by initially employing simpler filters to eliminate the biggest anomalies and, progressively, applying more complex methods to detect the errors that are harder to spot.

In order to structure this filtering process and guarantee rigorous quality control, the data schema has been expanded by incorporating validation variables (boolean indicators or flags) for each of the generated translations. For instance, for an original Catalan sample extracted from the Dacsa corpus, we create 4 new independent evaluation columns, one for every language in both translation models (NLLB and OPUS-MT).

This tabular design enables the isolated evaluation of each model's performance for every language pair. The main rule to keep a text in our final training set is that it must have at least one good translation. This means that if an OPUS-MT translation is flagged as deficient but the one generated by NLLB is valid (or vice versa), the sample is not discarded; instead, it is preserved by exclusively utilizing the optimal translation, thereby maximizing data utilization.

3.3.1. Phase 1: Filtering by Length Ratios (TATOEBA Method)

The objective of this initial phase is to identify and discard major structural anomalies. One method for evaluating translation quality, for both the OPUS-MT and NLLB models, is to consider that, depending on the language pair evaluated (e.g., English to Catalan or Spanish), the text will naturally get a bit longer or shorter. That is, when doing a cross language translation, the number of words fluctuates due to grammatical structures, plurals, or morphological variations. Consequently, a certain degree of quality can be guaranteed if the length of the translated summary stays within normal bounds. To establish

Example 1**Original summary:**

Associated Press critics Lou Kesten and Derrik J. Lang compiled their end-of-the-year lists of their favorites video games of 2014. Inquisition (BioWare, for PlayStation 3, PlayStation 4, Xbox 360, Xbox One, PC): BioWare's overstuffed role-playing epic has everything you could want in a fantasy saga: war, magic, exploration, romance and, yes, dragons. [...]

Catalan translation:

Els crítics de l'Associated Press Lou Kesten i Derrik J. Lang van compilar les seves llistes de finals de l'any de la seva sèrie de videojocs preferits de la PlayStation 3, PlayStation 4, PlayStation PlayStation 4, PlayStation 4, PlayStation Xbox One, Xbox One, PC, iOS i Android. [...] PlayStation 4, PlayStation 4, PlayStation 4, PlayStation 4 [...]

Example 2**Original summary:**

When Linda Vida sold her house in the Oakland hills this summer, she was hoping for a buyer who would live there, put kids in the local schools and "give back or participate in the community," she says. [...]

Spanish translation:

Cuando Linda Vida vendió su casa en las colinas de Oakland este verano, esperaba que muchos de estos millonarios vivieran allí, pusieran a los niños en las escuelas locales y dijeran "dar de vuelta o participar en la comunidad", dice ella. [...] los Estados Unidos, los Estados Unidos, los Estados Unidos, los Estados Unidos [...]

Figure 3.2: Examples of translations with repetition and hallucination errors in sequence-to-sequence models.

an objective frame of reference, the Gold Standard TATOEB A dataset (Tiedemann, 2020), a high-quality bilingual corpus, was utilized to study the statistical distribution of valid translation lengths. Based on this data, the ratio is computed at the word level by dividing the target text length by the source text length

As mentioned before of the primary errors observed in the dataset consisted of the model generating infinite word repetitions, resulting in extremely long sequences, we used a math trick called the Interquartile Range (IQR) method. This method enables the identification of outliers by establishing normality thresholds. Based on the first (Q_1) and third (Q_3) quartiles, the lower and upper acceptance limits are defined as follows:

$$IQR = Q_3 - Q_1$$

$$L_{lower} = Q_1 - 1.5 \times IQR$$

$$L_{upper} = Q_3 + 1.5 \times IQR$$

Consequently, any translation whose compression ratio falls outside the interval defined by $[L_{lower}, L_{upper}]$ is flagged as invalid and discarded as a candidate for final training.

Pair	Mean	Std. Dev.	P05	Q1	Median	Q3	P95	IQR	Lower Lim.	Upper Lim.
CA → EN	0.9306	0.1827	0.6842	0.8235	0.9167	1.0000	1.2222	0.1765	0.5588	1.2647
CA → ES	0.9803	0.1273	0.8000	0.9259	0.9828	1.0000	1.1702	0.0741	0.8148	1.1111
EN → CA	1.1114	0.2131	0.8182	1.0000	1.0909	1.2143	1.4615	0.2143	0.6786	1.5357
EN → ES	1.1031	0.2166	0.8286	1.0000	1.0833	1.2000	1.4359	0.2000	0.7000	1.5000
ES → CA	1.0362	0.1364	0.8545	1.0000	1.0175	1.0800	1.2500	0.0800	0.8800	1.2000
ES → EN	0.9338	0.1669	0.6964	0.8333	0.9231	1.0000	1.2069	0.1667	0.5833	1.2500

Table 3.5: Word ratio statistics for the different translation directions. The analysis is calculated over a total of 10,000,000 samples for each language pair using the TATOEBa dataset.

The results of this statistical analysis are shown in Table 3.5. This analysis allow us to identify significant patterns that validate the applied methodology. First, when English is utilized as the target language, the text gets shorter, yielding means around 0.93; whereas inverse translation from English into Catalan or Spanish induces an expansion of approximately 11% alongside greater variability ($IQR \geq 0.20$). Second, the strong morphological proximity between Catalan and Spanish is confirmed. Translations between them stay almost the exact same length (a ratio of 1) and have very little variation ($IQR \leq 0.08$). This tight correlation enables the Interquartile Range method to generate significantly narrower and more restrictive acceptance intervals $[L_{lower}, L_{upper}]$, thereby ensuring highly precise anomaly filtering.

3.3.2. Phase 2: Named Entity Preservation

Besides length considerations, it is critical that the translation preserves the key conditioning elements, the Named Entities. This second phase ensures that proper nouns have not been altered, mistranslated, or omitted. For this purpose, the previously extracted entities were used. Specifically, an initial filter was applied to exclusively select entities classified as person (PERSON), organization (ORG), and location (LOC) in both the source and translated texts, given that these are the most significant and useful for conditioning.

To evaluate this preservation, a minimum recall threshold of 0.70 was established. This means at least 70% of the entities present in the original summary appears in the translation. However, in machine translation tasks, valid orthographic variations are frequently encountered (such as "Catalunya" instead of "Cataluña", or "London" for "Londres"). To avoid throwing away good data because of this, a tolerance mechanism based on the Levenshtein distance was implemented, a metric that quantifies the minimum number of operations (insertions, deletions, or substitutions) required to transform one text string into another. Therefore, if a translation's recall falls below 0.70, the sample is exceptionally accepted when the average Levenshtein distance between the non-matching entities is less than or equal to 3 characters.

3.3.3. Phase 3: Semantic Similarity via Embeddings

The final and most rigorous test of the cleaning *pipeline* is designed to ensure meaning equivalence. A translation might possess the correct length and preserve proper nouns,

but completely alter the sentence's meaning.

For this reason, we decided to evaluate semantic coherence using the Sentence Transformers architecture (Reimers & Gurevych, 2019), specifically employing the pre-trained *paraphrase-multilingual-mpnet-base-v2*⁵ model. This model projects both the original text and the translation into the same high-dimensional vector space, generating dense embeddings. We calculated the cosine similarity between these two vectors of numbers, which measures the angle between them to see how close their meanings are.

A highly restrictive filtering threshold was applied, retaining exclusively those translations that achieve a cosine similarity greater than or equal to 0.85. This requirement guarantees that the resulting translations possess a meaning practically identical to the original, allowing the system to eliminate any trace of hallucination or contextual deviation.

3.3.4. Final Acceptance Criteria

In the end, the final high-quality dataset (*clean_dataset*) is the result of a strict intersection of these evaluation metrics. To wrap it up, a generated sample is only incorporated into the corpus if it simultaneously satisfies the following three conditions:

- **Mathematically proportional length:** The compression factor falls within the reference Interquartile Range (IQR) extracted from the TATOEBa corpus.
- **Named Entity Preservation:** The text achieves a recall ≥ 0.70 or otherwise surpasses the morphological tolerance margin with a Levenshtein distance ≤ 3 .
- **Semantic identity:** The cosine similarity between the *embeddings* of the original and translated texts is ≥ 0.85

3.4 Partitions

In order to guarantee a rigorous evaluation of the models and prevent biases in error estimation, the data partitioning strategy has been redefined. Instead of traditional approaches, where all samples are treated homogeneously, this work implements a separation into four specific subsets: Training, Validation, Test-I (In-Domain), and Test-NI (Out-of-Domain). The critical distinction lies in the nature of the testing sets:

- **Test-I (Internal):** Contains samples from the same sources as the training data, despite not being seen by the model during fine-tuning. This set allows for the evaluation of the model's capacity to learn patterns within the same operational context.
- **Test-NI (Non-Internal):** Consists exclusively of data from entirely unseen sources that did not participate in any other partition set. This subset is essential for validating the system's generalization capability across new scenarios, simulating a real production environment and ensuring that the model has not memorized specific characteristics of the training sources.

⁵Available in the official repository: <https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2>

3.4.1. Downsampling Strategy and Evaluation Set Construction

Given the huge original number of samples in the validation and test sets in the Dacsa (particularly the Spanish variant) and Newsroom corpora, a full evaluation over all the available samples would generate a prohibitive computational cost.

To address this, a truncation strategy was designed based on the language with the least number of samples in this case Catalan.

3.4.2. Definition of Truncation Thresholds

The criterion adopted by us is to use the Catalan dataset (following the cleaning and quality filtering process) as a baseline to establish the maximum number of samples per source, thereby mitigating the sample excess that would hinder results evaluation. Two thresholds (τ) have been defined:

- **Test-NI Limit (τ_{NI}):** Once the dataset cleaning is done, it is observed that the Catalan Dacsa sources used in Test-NI have the following counts: $\{CA08 : 3467, CA09 : 3198, CA07 : 4313\}$. In order to reduce the number of samples and homogenize subset sizes, the threshold is set to the minimum observed value:

$$\tau_{NI} = \min(3467, 3198, 4313) = 3198 \text{ samples per source.}$$

- **Test-I Limit (τ_I):** We find this number the exact same way, by using the least amount of samples in the Test-I Catalan sources. Since the source with the fewest examples after filtering has 3,303 texts, we set the rule as:

$$\tau_I = 3303 \text{ examples per source.}$$

3.4.3. Processing of the Dacsa Corpora (Catalan and Spanish) for Test-I

For the Dacsa datasets, since they already had an original Test-I section, a simple truncation was performed to ensure that all sources maintained an identical number of samples. Since the minimum value (the τ_I threshold) is 3,303 samples per source, all sources were truncated to this exact value. As a result, we get 19,818 Catalan examples from the 6 available sources, and 42,939 Spanish examples from 13 sources.

3.4.4. Processing of the Dacsa Corpora (Catalan and Spanish) for Test-NI

For the corpora subsets originating from Dacsa, which already featured an original partition with segregated sources for generalization purposes (Test-NI), the truncation strategy was directly applied with certain variations:

- **Catalan:** As previously mentioned, the minimum reference value was established as the threshold (τ_{NI}). Therefore, we simply cut down the data to this number. Specifically, the 3 sources are limited to 3,198 samples each, yielding a Test-NI partition of 9,594 samples from 3 unseen sources.

- **Spanish:** When looking at the original number of samples per source, it is observed that only 3 sources exceed the established threshold, while the remaining ones are highly limited and underrepresented. Because of this, the sources with higher volume were truncated to the τ_{NI} value, whereas all available samples were retained for the remaining sources. Since the Test-NI part checks how well are the model generalization capabilities, incorporating more sources with diverse styles allows for a more accurate assessment of text generation. The final set has been considerably reduced to 14,108 samples.

3.4.5. Processing of the Newsroom Corpus for Test-I

Up to this point, for the Dacs datasets, we only made smaller groups from the parts that were already there. Conversely, in the Newsroom dataset, there were no predefined Test-I or Test-NI partitions; rather, only a generic test partition was available.

The first step to make these two new parts was finding where each text comes from, in other words find the source for every sample. To achieve this, because the URL was already included in the original dataset, we were able to extract the source via the URL domain. For example, from the URL http://www.nydailynews.com/archives/news/1995/10/14/1995-10-14_selena_s_last_cries___shot_s.html, the source would be "nydailynews". An exploratory analysis reveals that within the Test partition, following the cleaning process, there are 363 unique sources. Furthermore, these sources are also spread across the Train and Validation partitions. Given that there are no exclusive sources for the Test set, it is essential to first reserve certain sources to be used for training, validation, and Test-I, while allocating the remaining ones to Test-NI.

Within the Test partition, we can count the sources as shown in Table 3.6. By applying the Test-I threshold to sources with more than 3,303 samples, 6 sources are selected and limited to this value, yielding exactly 19,818 total samples, which is the same size as the Catalan subset.

Position	Source (Brand)	Samples
1	nytimes	12,345
2	washingtonpost	8,578
3	foxnews	7,083
4	theguardian	5,514
5	nydailynews	5,221
6	cnn	3,412
7	mashable	3,010
8	time	3,009
9	people	2,939
10	wsj	2,832
... 353 remaining sources (Long-tail distribution) ...		
-	Minor sources (e.g., yahoo, thechoice)	4
-	Occasional sources (e.g., poker, fish)	2
-	67 unique sources (e.g., china, steelerswire)	1

Table 3.6: Top 10 news sources and frequency long-tail summary.

3.4.6. Processing of the Newsroom Corpus for Test-NI

For the construction of the Test-NI subset, as previously mentioned, we need to pick unique sources. In this case, it was decided to utilize the remaining sources from the long tail of the test set that lacked sufficient samples to be included in Test-I. The fundamental objective of this partition is to guarantee the evaluation of the model’s generalization capability when faced with entirely new environments, this set-up seems appropriate. Given the nature of this long-tail distribution, with a multitude of domains providing few samples, the Test-NI task in English is considered considerably more complex, as it contains a total of 357 different sources and 53,972 total samples.

Filtering of the Training and Validation Sets in Newsroom

A significant methodological problem with the original Newsroom dataset is that its default partitioning (Training, Validation, and Test) was performed randomly across the global set of samples. This approach resulted in the simultaneous presence of the same information sources across all partitions. This source overlap invalidates any true generalization testing.

To guarantee the integrity of the Test-NI set and prevent data leakage, we had to filter the data strictly. Specifically, we set an exclusion protocol: all sources designated for the Test-NI set were completely removed from both the training, validation and test_i sets. Leaving us with a final count of 437,034 samples for the training subset and 20,205 for the validation one.

3.4.7. Homogenization and Design of the Dacs Validation Set

In the validation partition, we made sure that all sources have the same number of examples and that every language uses the same number of sources. This balance is imperative to avoid biases in the evaluation metrics and to guarantee an identical validation procedure for each language. This way, it is possible to objectively assess performance and select the model checkpoint that proves to be the most equitable on a global scale. We want to avoid picking a model that looks amazing during validation just because it is optimized for one language, but then performs poorly when we move to the final test.

Therefore, just in the same manner as the threshold determination for *Test-I* and *Test-NI*, Catalan operates as the limiting factor. To make the validation fair for all languages, and given that the original Catalan validation set comprises 6 sources, these have been truncated to their minimum value to obtain a fully homogeneous validation set.

Also, to maintain this equity in the evaluation of Spanish and English, we chose to pick 6 validation sources for those languages too, limiting them to a total of 3,370 examples per source.

3.4.8. Downsampling and Balancing of the Dacs Spanish Training Set

The construction of the training set for the Spanish variant of Dacs required a down-sampling and balancing process in order to optimize model performance and guarantee the fairly representation of all data languages. The process initiates from an original set

containing a total of $N_{total} = 1.574.048$ samples across 13 different sources. Our goal is to create a final smaller group of $N_{target} = 615.919$ samples. We did not choose this number at random, it is derived from the objective of establishing a global corpus of 1,5 million samples, ensuring an equitable distribution among the three languages. Consequently, after the data cleaning, the final dataset contains a total of 884,081 Catalan and English samples the remaining margin required to reach 1,5 million is allocated to Spanish. If we used all the Spanish data available, it would create a big imbalanced dataset because Spanish would have twice as many examples as the other languages.

To shrink the data without losing the smaller sources, we designed a hybrid sampling strategy with weighted redistribution. This method prioritizes a minimum quota for all sources and subsequently reintroduces the original representativeness bias exclusively to those with higher data volumes.

The procedure begins by establishing a base quota (Q_b), which represents the perfect even split if all sources had enough data in Formula 3.1:

$$Q_b = \left\lfloor \frac{N_{target}}{|S|} \right\rfloor = \frac{615.919}{13} \approx 47.378 \text{ samples} \quad (3.1)$$

where N_{target} denotes the target number of samples allocated to the Spanish subset, and $|S|$ represents the total number of available data sources.

Based on this threshold, the sources are classified into two subsets: low-volume sources (S_{low}), whose original count satisfies $C(s_i) < Q_b$, and high-volume sources (S_{high}), on $C(s_i) \geq Q_b$.

The algorithm initially allocates all available samples for the sources in S_{low} and Q_b for the sources in S_{high} . This first step leaves some extra slots unfilled (R) due to the inability of small sources to fulfill their quota of unused samples. This surplus is calculated as shown in Formula 3.2:

$$R = N_{target} - \left(\sum_{s \in S_{low}} C(s) + (|S_{high}| \times Q_b) \right) \quad (3.2)$$

In order to distribute this remainder R while respecting the original statistical structure of the bigger sources, we use a weighting factor w_i . This weight is computed exclusively over the S_{high} subset, ensuring that the originally more representative sources receive a higher proportion of the surplus, the Formula 3.3 shows how this weighting factor is calculated:

$$w_i = \frac{C(s_i)}{\sum_{j \in S_{high}} C(s_j)}, \quad \forall s_i \in S_{high} \quad (3.3)$$

Finally, the assigned number of samples (N_i) for each source in the S_{high} group is defined by the sum of the guaranteed base quota and the proportional share of the remainder, as the Formula 3.4 explains:

$$N_i = Q_b + \lfloor R \times w_i \rfloor \quad (3.4)$$

This method allows large sources like **ES01** to recover a significant portion of their original representativeness through the allocation of approximately 77,636 samples, At

the same time, medium sources like **ES08** stay close to their real size with around 54,641 examples. This creates a perfect balance: 100% of the information from low-volume sources is preserved, and the natural bias of high-volume sources is mitigated without being completely eliminated.

Full Preservation of the Dacsa Catalan Training Set

Unlike the Spanish variant, which we had to shrink because it was too big, the Dacsa Catalan training set has been fully preserved. After cleaning the text and removing the mistakes, we included every single available sample from the sources that were not saved for the *Test_NI* test.

We made this choice because Catalan is the smallest language group in this project. Consequently, maximizing the number of training samples is a critical priority to enable the model to robustly learn the language. Also, since the final size is small enough for the computer to handle easily, keeping all the data will not cause any balancing problems or ruin the training process.

3.5 Summary and Statistics for the Final Dataset

Once we explained how the final dataset was created and cleaned, this section presents a descriptive analysis of the data. The main objective of this section is to provide a clear, numbers-based view of the main features of the collected dataset information.

Source	Partition	Samples	Avg Orig. Words	Avg Sum. Words	Avg NERs	Avg Compr.	Avg Sim.
CA01	train	149,389	412.506	21.607	1.954	0.077	0.944
	validation	3,370	414.002	21.451	1.932	0.077	0.944
	test_i	3,303	416.776	21.714	1.967	0.078	0.943
CA02	train	118,503	472.482	22.336	1.870	0.060	0.943
	validation	3,370	470.905	22.407	1.917	0.060	0.944
	test_i	3,303	469.626	22.168	1.876	0.061	0.943
CA03	train	77,246	416.917	22.584	1.837	0.068	0.941
	validation	3,370	423.311	22.544	1.822	0.067	0.941
	test_i	3,303	422.874	22.490	1.803	0.067	0.941
CA04	train	25,774	379.961	19.285	1.569	0.070	0.939
	validation	3,370	374.625	19.283	1.569	0.070	0.940
	test_i	3,303	388.545	19.254	1.580	0.069	0.940
CA05	train	21,308	545.714	24.983	2.039	0.074	0.940
	validation	3,370	550.995	25.386	1.993	0.074	0.941
	test_i	3,303	548.452	25.143	2.039	0.075	0.942
CA06	train	14,713	431.995	22.663	1.902	0.068	0.939
	validation	3,370	429.996	22.966	1.930	0.070	0.939
	test_i	3,303	425.036	22.571	1.917	0.068	0.939
CA07*	test_ni	3,198	481.851	25.183	1.718	0.074	0.943
CA08*	test_ni	3,198	1,414.632	54.956	5.270	0.053	0.939
CA09*	test_ni	3,198	482.539	29.401	3.100	0.118	0.948

Table 3.7: Global statistics for the final catalan corpus. Number of samples, average words in the original text, average words in the selected summaries, average number of named entities, average compression, and average similarity are shown, grouped by source and partition. Sources marked with an asterisk (*) appear exclusively in the *Test_NI* partition.

Looking at the Catalan part of the final dataset, shown in Table 3.7, the application of the *Test-I* and *Test-NI* thresholds is observed, limiting the number of samples to 3,303 for *Test-I* and 3,198 for *Test-NI*. Also, it is clear that the first filters worked very well. If we check the Avg Sim column, all sources have very high scores, close to 0.940. This means it is perfectly safe to use the translated texts because they mean almost exactly the same thing as the original ones.

When looking at the structure of the texts, a high degree of stability is appreciated across the majority of the training sources. The average length of the original texts is mostly between 370 and 550 words, and the summaries are very short, usually around 20 to 25 words. However, the CA08* source stands out as a clear outlier within the *Test-NI* set: its original texts triple the average length of the remaining sources (1,414 words), and the summaries are consequently longer (nearly 55 words). This additional text is also reflected in the number of Name Entities in the source which reaches an average of 5.27. This is much higher than the 1.5 to 2.0 average seen in the other sources.

When looking at the overall statistics for the Spanish dataset (Table 3.8), one of the most notable aspects is the high average semantic similarity index (Avg Sim.). This metric remains extremely constant around 0.95 and 0.96 across almost all partitions, a value superior to that observed in the Catalan corpus. This small difference likely happens because models work better and understand meaning more accurately in widely spoken languages like Spanish or English, given the massive volume of data they have been pre-trained on compared to Catalan.

We must pay special attention to the sources saved only for the external test group (*Test-NI*, marked with a star). This subset exhibits a highly marked heterogeneity and reveals sources with natures radically different from the standardized training data. For example, the ES17* source has unusually long original texts, averaging over 1,620 words, with summaries around 72 words. Likewise, domains such as ES11*, ES15*, and ES18* reflect very high informational complexity, presenting a Named Entity density exceeding 4.6, and reaching up to 5.6 in the case of ES15*. These numbers are much higher than the average of 1.5 to 2.5 found in the training data. This huge variety will be a tough test to check how well the models can adapt to new situations.

Regarding the English part, as observed in Table 3.9, the training, validation, and in-domain test (*Test-I*) groups were limited to only six main sources (CNN, Fox News, NY Daily News, NY Times, The Guardian, and Washington Post).

When comparing these data with the Catalan and Spanish corpora, it is noteworthy that the original English texts are significantly longer, often having between 750 and 880 words. Given that the summaries remain concise (between 20 and 30 words), the compression ratios are consequently more severe (with values approaching 0.05 and 0.07). This means that summarizing the English texts requires a much higher level of abstraction.

As discussed in the dataset creation section, the *Test-NI* partition corresponding to the English Newsroom corpus exhibits an extremely pronounced long-tail distribution, as shown in Table 3.10. Specifically, there are over 357 different sources with about 54,000 examples, nearly 46% of the samples are concentrated exclusively within the top 10 sources, whereas the remaining half is sparsely distributed across all other domains.

Source	Partition	Samples	Avg Orig. Words	Avg Sum. Words	Avg NERs	Avg Compr.	Avg Sim.
ES01	train	77,636	669.363	23.950	2.203	0.059	0.956
	validation	3,370	681.038	23.965	2.166	0.058	0.957
	test_i	3,303	681.038	23.965	2.166	0.058	0.957
ES02	train	67,218	451.230	21.253	1.770	0.064	0.952
	validation	3,370	453.920	21.244	1.766	0.063	0.952
	test_i	3,303	453.920	21.244	1.766	0.063	0.952
ES03	train	59,541	422.522	18.948	1.503	0.061	0.954
	validation	3,370	429.473	18.915	1.521	0.061	0.954
	test_i	3,303	419.113	18.906	1.510	0.062	0.954
ES04	train	58,355	547.240	21.670	1.766	0.059	0.958
	validation	—	—	—	—	—	—
	test_i	3,303	543.977	21.821	1.766	0.060	0.958
ES05	train	57,101	629.030	26.478	2.278	0.062	0.954
	validation	—	—	—	—	—	—
	test_i	3,303	622.337	26.461	2.290	0.062	0.953
ES06	train	56,060	716.095	49.114	3.750	0.092	0.955
	validation	3,370	725.746	49.430	3.745	0.092	0.955
	test_i	3,303	730.131	49.634	3.784	0.092	0.955
ES07	train	55,356	582.829	28.946	2.366	0.068	0.959
	validation	—	—	—	—	—	—
	test_i	3,303	565.983	28.795	2.321	0.069	0.959
ES08	train	54,641	588.709	20.316	1.766	0.050	0.957
	validation	3,370	593.994	20.378	1.757	0.049	0.956
	test_i	3,303	582.737	20.568	1.817	0.051	0.957
ES09	train	53,787	464.234	21.203	1.748	0.062	0.951
	validation	—	—	—	—	—	—
	test_i	3,303	464.731	21.252	1.757	0.062	0.951
ES10	train	50,327	675.115	23.537	1.950	0.054	0.956
	validation	—	—	—	—	—	—
	test_i	3,303	679.545	23.340	1.966	0.053	0.956
ES11*	test_ni	3,198	376.333	47.849	4.647	0.178	0.953
ES12	train	32,419	433.536	21.663	1.918	0.065	0.951
	validation	3,370	430.778	21.451	1.901	0.065	0.951
	test_i	3,303	432.198	21.646	1.893	0.065	0.952
ES13	train	16,628	511.585	21.625	1.738	0.058	0.957
	validation	3,370	504.778	21.564	1.720	0.058	0.957
	test_i	3,303	503.900	21.323	1.732	0.059	0.957
ES14	train	16,964	716.514	29.957	3.137	0.059	0.953
	validation	3,370	718.061	29.953	3.138	0.058	0.953
	test_i	3,303	712.987	29.829	3.174	0.057	0.953
ES15*	test_ni	3,198	517.024	42.807	5.652	0.113	0.961
ES16*	test_ni	3,198	721.482	24.240	2.041	0.047	0.955
ES17*	test_ni	2,215	1,621.596	71.761	4.598	0.080	0.960
ES18*	test_ni	1,270	406.261	50.445	5.416	0.165	0.959
ES19*	test_ni	500	558.642	14.824	1.966	0.038	0.956
ES20*	test_ni	404	455.748	26.314	2.288	0.080	0.954
ES21*	test_ni	125	1,174.344	37.258	3.848	0.086	0.952

Table 3.8: Global statistics for the final Spanish corpus. Number of samples, average words in the original text, average words in the selected summaries, average number of named entities, average compression, and average similarity are shown, grouped by source and partition. Sources marked with an asterisk (*) appear exclusively in the *Test_NI* partition.

Source	Partition	Samples	Avg Orig. Words	Avg Sum. Words	Avg NERs	Avg Compr.	Avg Sim.
cnn	train	109	769.963	31.638	2.320	0.051	0.945
	validation	3,370	797.931	28.025	2.242	0.059	0.948
	test_i	3,303	799.572	27.499	2.209	0.059	0.947
foxnews	train	7,527	529.292	30.683	2.408	0.092	0.946
	validation	3,370	594.225	29.321	2.395	0.080	0.945
	test_i	3,303	593.002	29.674	2.448	0.081	0.945
nydailynews	train	3,901	474.595	37.926	3.352	0.114	0.940
	validation	3,370	495.114	29.662	2.699	0.085	0.940
	test_i	3,303	491.196	29.768	2.734	0.086	0.940
nytimes	train	17,886	857.969	24.677	1.721	0.041	0.946
	validation	3,370	867.706	58.475	4.408	0.096	0.946
	test_i	3,303	866.664	58.649	4.430	0.094	0.946
theguardian	train	4,467	752.005	24.731	2.237	0.055	0.939
	validation	3,370	867.220	25.990	2.082	0.052	0.939
	test_i	3,303	864.950	25.709	2.041	0.053	0.940
washingtonpost	train	10,349	773.105	24.850	2.125	0.072	0.944
	validation	3,370	880.320	18.297	1.270	0.029	0.943
	test_i	3,303	874.666	18.151	1.228	0.028	0.943

Table 3.9: Global statistics for the English corpus. Number of samples, average words in the original text, average words in the selected summaries, average number of named entities, average compression, and average similarity are shown, grouped by source and partition.

Source	Samples	Percentage
mashable	3,010	5.6%
time	3,009	5.6%
people	2,939	5.4%
wsj	2,832	5.2%
fortune	2,371	4.4%
cnbc	2,344	4.3%
forbes	2,305	4.3%
9news	2,226	4.1%
content	1,932	3.6%
tmz	1,869	3.5%
OTHER	29,135	54.0%
Overall statistics (53,972 samples, 357 distinct sources)		
Average original words: 572.024		
Average summary words: 27.696		
Average NERs: 2.410		
Average compression: 0.101		
Average similarity: 0.943		

Table 3.10: Source distribution (top 10 + OTHER) and overall metrics for the English *Test-NI*.

This phenomenon illustrates the partition complexity: the task associated with the English *Test-NI* constitutes a highly demanding challenge for the summarization model’s generalization capability, owing to the presence of a vast quantity of highly diverse sources.

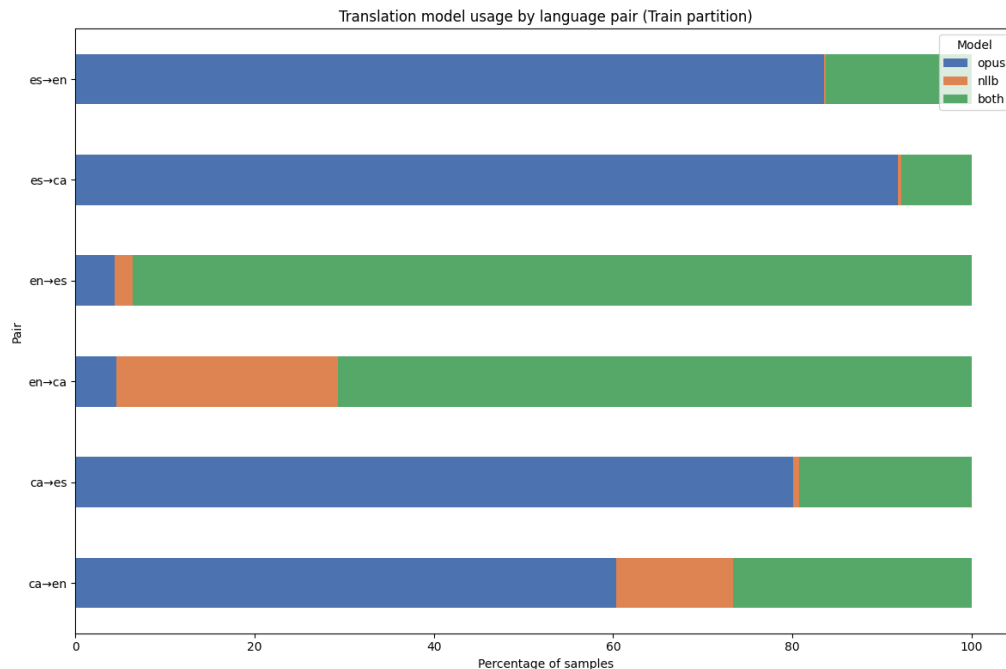


Figure 3.3: Proportional contribution of OPUS and NLLB models to dataset translation, categorized by language pair directional flow in the training split.

Finally, Figure 3.3 shows the percentage of final examples translated by each model. Even though only the training partition is represented, the exploratory analysis confirms that this behavior remains almost the same across the remaining partitions. The main takeaway is that specialized models work better than general ones. In translations involving languages with lower relative representation in massive models, such as Catalan or Spanish, the NLLB model exhibits notable difficulties, resulting in an absolute predominance of samples translated via OPUS.

When working with English, the NLLB model works very well, and both models handle most of the texts correctly. Even when translating from English to Catalan, the Meta model individually achieves a higher number of correct translations compared to OPUS, despite the immense majority of instances being optimally translated by both systems.

CHAPTER 4

Proposed Architectural Framework

Once the dataset creation process has been concluded, this chapter details the proposed architectures for text generation models and the learning process.

4.1 Conditioning the mT5 Model via Custom Tokens

Our first proposal involves the use of special tokens that will instruct the model on how it should behave. As previously discussed, the conditioning will be based on three fundamental aspects: language, length, and named entities.

In order to understand the mechanics of this conditioning, we first need to explain what a token is and what it does in the model. A token is the smallest unit of information that an artificial intelligence model can read. For example:

- A complete word (ex: "hello", "cat")
- A part of a word or a syllable (ex: "un", "pre")
- A single letter or symbol (ex: "a", "?", " ").

For each token in the model's vocabulary a unique identification number (ID) is assigned. Because of this, the phrase "Hello world" is not processed directly as text, but is rather converted into a numerical sequence, for example: [1354, 8820]. This is done by a tool called a tokenizer. In the case of the mT5 architecture, the subword tokenization algorithm known as SentencePiece is employed (Kudo & Richardson, 2018).

In our setup, we add a set of special tokens for each control feature:

- **Language:** <LANG-SAME-AS-INPUT>, <SUMMARIZE-IN-CATALAN>, <SUMMARIZE-IN-SPANISH>, <SUMMARIZE-IN-ENGLISH>
- **Length:** <DECIDE-SUMMARY-LENGTH>, <LARGE-LENGTH>, <SHORT-LENGTH>
- **Named Entities (NER):** <SELECT-RELEVANT-NERS>, <NERS>, </NERS>, <NER>, </NER>, <NO-NER>

The creation of these tokens means we have to change the tokenizer so it can read these custom text tags and turn them into numbers. But also modifying the Transformer

architecture.

The first step is to determine which encoder-decoder model will be used. We originally thought about using Flan-T5. However, as explained in section 2.4.5, the public versions of Flan-T5 only work with English, completely discarding its use given the trilingual environment of this project. Instead, we will use mT5, which is the multilingual version of T5, because it already works with all three languages. We also cannot use BART models (Lewis et al., 2019) because they do not fully support Catalan.

In the specific case of the mT5 model, the original embedding matrix has dimensions of [250,112 x 768]. However, if you check the tokenizer tool in PyTorch (Paszke et al., 2019), there are only 250,100 tokens (numbered 0 to 250,099). That means that there are 12 rows in the embedding matrix that do not correspond to real tokens since they were added so the total number is a multiple of 128, which allows their TPUs (Tensor Processing Units) to perform matrix multiplications much more efficiently.

Initially, when we add our new custom tokens they would adopt the IDs of the last 12 available tokens, and if more are added, new rows are created in the matrix. In our case, the final matrix size would become [250,113 x 768]. This generates a hidden issue during training: our pre-trained model already knows the meaning of the first 250,100 tokens because it was trained on huge amounts of text. However, it will also have 13 new ones that the model has never encountered and whose meaning it ignores. While the original 250,100 tokens possess vectors rich in meaning and are geometrically well-positioned in this 768-dimensional space, the 13 new ones are merely meaningless noise. But, we could try to solve it via manual initialization for each token. For example, <SUMMARIZE-IN-CATALAN> can be initialized as the embedding value of "Catalan" or a phrase like "Summary in Catalan". If we use a phrase with multiple words, we would just calculate the average of their numbers to create one single starting point.

Even though this method (called semantic initialization) is a clear improvement over randomness and provides a much more stable starting point, it does not fully fix the problem. This happens for a few reasons:

First, there is a big difference between a word's meaning and control function. Even if we start the token <SUMMARIZE-IN-CATALAN> with the values of "Summary in Catalan", the model understands these values as linguistic concepts, but not as a strict rule telling it to do a specific task. To learn this control or conditioning behavior, the model must be trained and update its weights.

Faced with this problem, it seems like the easiest solution would be to freeze the 250,100 original tokens to keep them safe, and only train the 13 new ones. Nevertheless, in practice, a significant technical limitation arises from frameworks like PyTorch, upon which the HuggingFace Transformers library is based (Wolf et al., 2020): embedding matrices do not natively support partial freezing. The `requires_grad` switch applies to the whole tensor at once (you must freeze all of it or none of it).

We could try to bypass this limitation through the use of *backward hooks* (functions that intercept and modify the gradient immediately prior to updating the weights, zeroing out the gradients of the legacy tokens). Even though this works, this technique is highly inefficient. The math engine will still calculate the updates for all 250,113 tokens

during the training step (*backward pass*) unnecessarily consuming computational time and GPU memory, only to ultimately discard 99.9% of these calculations.

To fix this waste of resources and completely stop the catastrophic forgetting, this thesis proposes a structural separation combined with Parameter-Efficient Fine-Tuning (PEFT) methods:

4.1.1. Adaptation of the Input Layer (SplitEmbedding)

To overcome the PyTorch limitation that prevents partial freezing of an embedding matrix, we designed the Split Embeddings setup. Instead of appending rows to the original matrix, the 13 new tokens are extracted into a new, completely separate embedding matrix. This way, we will keep the original [250,112 x 768] matrix fully frozen (incurring zero computational cost) and only train the small [13 x 768] matrix, where gradients are computed and applied cleanly and directly. To do this, we implemented a module to act as a dynamic router during the model's forward pass, dividing the data flow based on the token's origin. The process is structured into four primary steps:

- Firstly, when a new string arrives, like "Hello <SUMMARIZE-IN-CATALAN>", the model transforms it into a tensor with dimensions [batch_size, seq_len, embedding_dim], where batch_size represents the number of independent text sequences processed simultaneously, seq_len the total number of tokens in the input sequence, and embedding_dim specifies the size of the dense vector used to encode the semantic representation of each token.
- Secondly, the text is translated into a list of ID numbers (for example, [10, 23, 250109]). Once the model receives this list, it performs a classification via boolean masks by applying a cutoff point (cutoff_index = 250,100). These masks identify which tokens belong to the original vocabulary and which are the newly added ones; the legacy token mask would yield [True, True, False] and naturally the mask for the new tokens would be [False, False, True].
- Thirdly, Thanks to this identification and separation, the original tokens are only handled by the old, pre-trained matrix with its gradients disabled (requires_grad = False), making sure the model does not forget what it already knows. Conversely, since the new tokens (e.g., 250,109) cannot go straight into the original matrix, we subtract the cutoff value to the new tokens in order to normalize their index (e.g., 250,109 - 250,100 = 9) and obtain their position in the new matrix. Allowing its representative vector to be retrieved from the new, trainable embedding matrix (requires_grad = True), which was initialized using the mean of the legacy tokens.
- Finally, the vectors obtained from both matrices are recombined in their original positions, generating the output tensor with dimensions [batch_size, seq_len, hidden_size]. Here, the first two tokens get their math from the original matrix, and the last one gets its math from the new matrix. This combined data will then move forward into the model's attention layers.

Due to the weight tying inherent in the mT5 architecture, once we restructure solely this initial input layer ensures the modification remains homogeneous and propagates to the output layers. Figure 4.1 illustrates the sequence of steps to process this information.

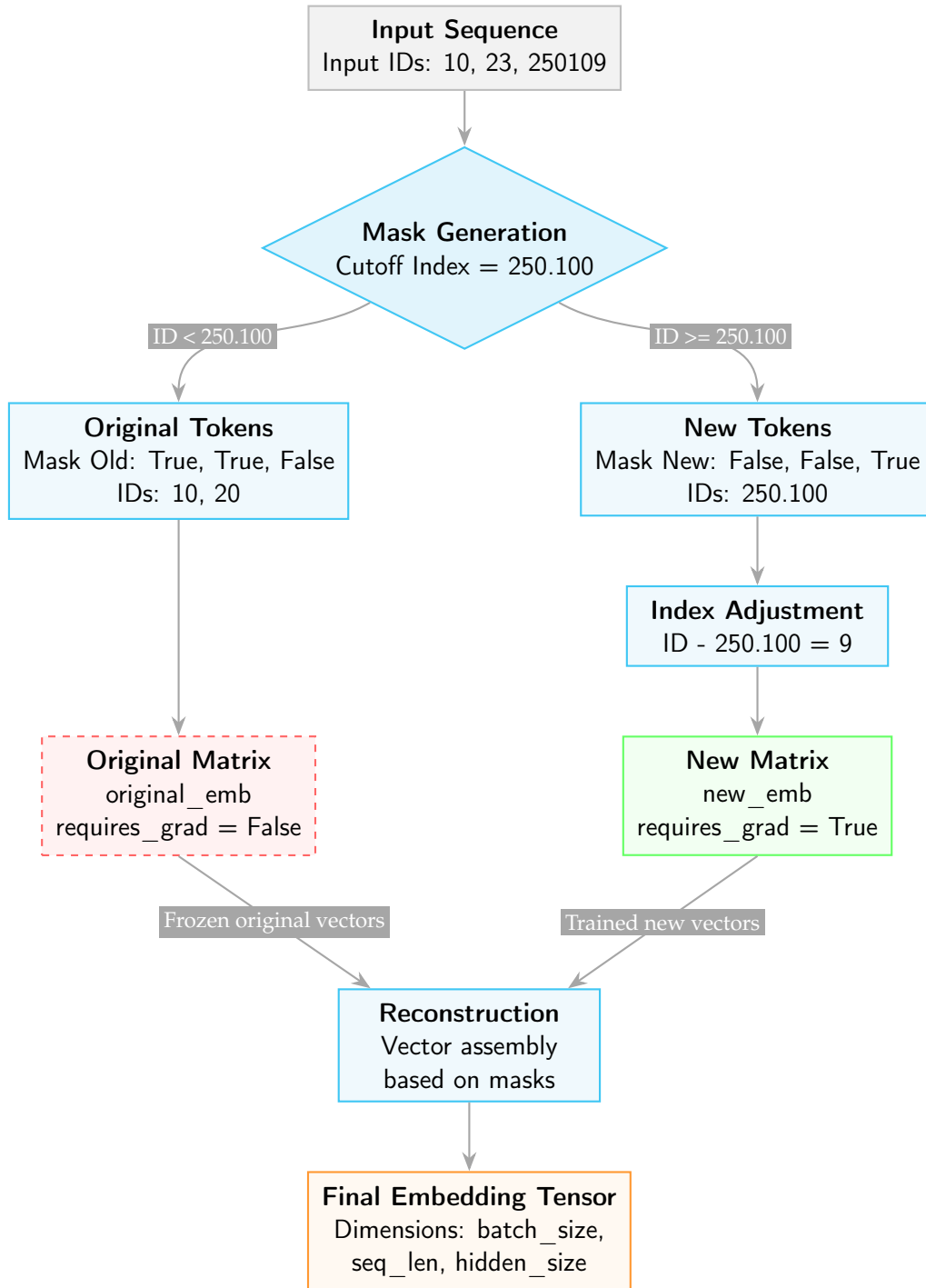


Figure 4.1: Architectural pipeline of the custom SplitEmbedding layer, illustrating the conditional routing, index adjustment, and mask-based vector reconstruction for vocabulary expansion.

4.1.2. Adaptation of the Output Layer (SplitLMHead)

Just like the input layer needs to be changed to prevent the model from forgetting and to allow it to learn new tokens, the output layer (known as the *Language Modeling Head*) presents an identical challenge. This final layer takes the data from the end of the model (hidden states) and matches it to the dictionary, calculating the chances for each possible next word.

We created a custom tool called `SplitLMHead`, which replicates the partition and isolation philosophy of the input layer, but adapted to the final projection process. Its operation can be broken down into several primary phases:

First, a restructuring and freezing of the original matrix is performed. When the model is initialized, the weight matrix is extracted from the original projection layer. Just like before, it uses the `cutoff_index` to cut out extra rows, keeping only the numbers for the valid original tokens. This truncated matrix is stored as a fixed parameter, with gradient updating completely disabled (`requires_grad = False`). Concurrently, the module is linked to the weights of the previously created `new_emb` matrix, which is left unlocked so the model can learn. This setup is almost exactly the same as the one used for the input layer.

During the final projection phase, the Transformer's last layer sends out the final data of the text, shaped as `[batch_size, seq_len, hidden_size]`. The objective of this phase is to obtain the *logits*. These are defined as the raw, unnormalized numerical values produced by the neural network immediately prior to the final prediction phase. The higher the score for a specific token, the more likely the model is to pick it as the next token.

Normally, the transformation of this hidden state tensor into the vocabulary probability space is executed via a single matrix multiplication resulting in a single global tensor. Instead of one big calculation, the process is replaced by two separate math steps happening at the same time:

- **Evaluation of the base vocabulary:** A linear projection (`F.linear`) is applied using the hidden states and the original (frozen) weight matrix. This step calculates the logits for the historical vocabulary (`logits_old`), ensuring that the original semantic relationships remain unaltered.
- **Evaluation of the new semantics:** At the same time, the same hidden states are projected exclusively against the new, trainable weight matrix. This allows the loss to update exclusively this region during training, generating a second, much lighter logit tensor (`logits_new`).

Once we got all the raw scores for the new and old tokens, we have to concatenate these two vectors into a final global logits tensor. Because in order to be able to apply the Softmax function, all the punctuations must be together to be transformed into a single probabilistic distribution.

This design completes the ecosystem initiated by the *Split Embeddings* architecture. When computing the error (*loss*) and executing backpropagation, the PyTorch architecture makes sure the learning updates only go to the new tokens. This way, the prior

linguistic knowledge of the base model is kept entirely shielded against catastrophic forgetting, while achieving a highly efficient adaptation focused only on the new tokens. The complete logic is visually illustrated in Figure 4.2.

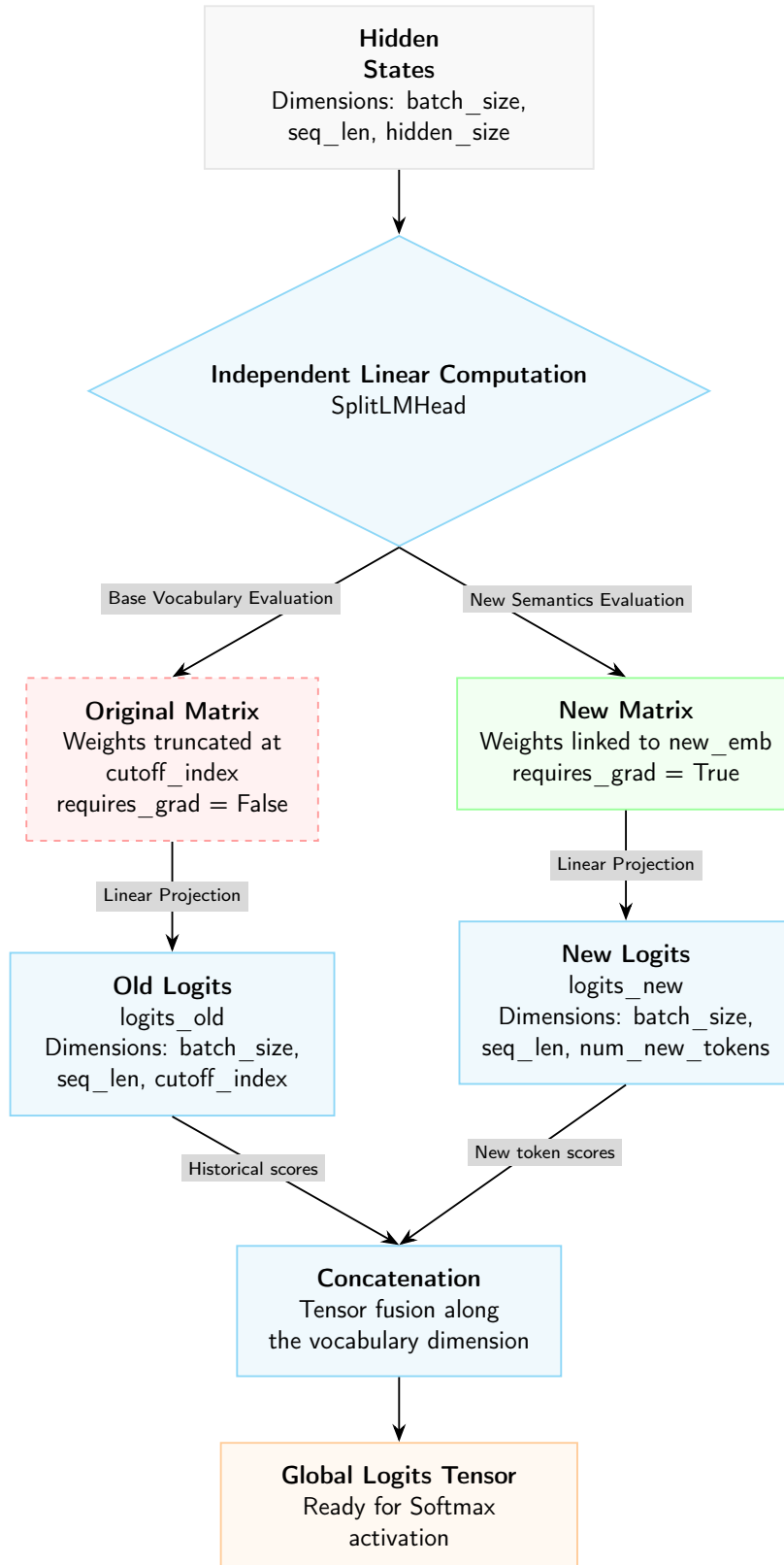


Figure 4.2: Architectural data flow within the proposed SplitLMHead layer, showcasing the bifurcated projection of hidden states and their subsequent concatenation into a unified global logits tensor.

4.1.3. Model Input Structure

We use a dual conditioning approach, which implies that we indicate the instructions in both the encoder and the decoder parts. This presents several advantages; for example, including the instructions in the encoder can help the model focus more on certain parts of the sentence over others depending on the predicted action. By passing these instructions again as `decoder_input_ids`, we reinforce the model with the instructions it must perform, so that at the time of generation it is already aware of the instructions to apply. An example of the input to the encoder is detailed in figure 4.3

Example: Instruction block construction for encoder input

1. Language command:

`<LANG-SAME-AS-INPUT>`

2. Named Entities (NER) block:

Entities: "Messi", "Barcelona"

`<NERS><NER> Messi </NER><NER> Barcelona </NER></NERS>`

3. Length instruction:

`<SHORT-LENGTH>`

Final complete input (Instructions + Text sequence):

`<LANG-SAME-AS-INPUT><NERS><NER> Messi </NER><NER> Barcelona
</NER></NERS><SHORT-LENGTH> text to summarize.`

Figure 4.3: Step-by-step construction of the instruction block and the final concatenated string fed into the encoder.

Our dual conditioning approach requires supplying these actions to the decoder too. To do this, we use the `decoder_input_ids` setting, which works by adding a prefix right before the decoder starts writing. Due to the inherent configuration of encoder-decoder architectures, generation always commences with an initial token (such as a padding or `cls` token). In the case of mT5, this is the `<pad>` token, and from this point, the model begins generating text. In our proposal, the instructions are added immediately after this initial token to remind the model of the task it must perform. This way, when the decoder starts writing the summary, it already sees the instruction tags telling it exactly how to behave.

4.2 Conditioning the mT5 Model via Natural Language Sentences

The second approach for controlling the summary is to use natural language sentences to guide the instructions the model must follow when generating the text. This approach is supported by the architecture and training methodology of the model Flan-T5. This family of models empirically demonstrated that the introduction of natural language instructions as a prefix enables the model to achieve greater generalization capabilities. And also, it is capable of efficiently executing multiple nlp tasks without the necessity of altering the embedding space. Since we do not add any new tokens, the model's struc-

ture stays exactly the same, which makes this method much simpler.

Even though it might seem too basic, this idea has a couple of benefits compared to the previous one. The biggest benefit is its simplicity. Since in this approach we don't include new tokens, there is no need to modify the architecture, allowing the use of the standard model. Furthermore, it provides greater creativity and freedom when applying the conditioning. In the best-case scenario, if the model completely understands the meaning of the words, we could theoretically use any sentence to guide the model.

4.2.1. Creation of the Sentences

For our conditioning, it is observed that there are 4 options for the target language (same as the input text, Catalan, Spanish or English), 4 for the named entities (depending on whether there are 0, 1, 2, or 3), and 2 for the length (short summary or long summary). However, when using sentences for instructions, a new variable is introduced: the input language, which determines the language in which the instruction is written. Because of this, the conditioning relies on four fundamental parameters: source language (4 options), target language (4 options), NER quantity (4 options), and length (2 options).

Given this framework, the total number of possible combinations is derived directly from the product of the options for each variable ($4 \times 4 \times 4 \times 2$), which gives a total of 128 unique base combinations or different instructions.

Due to the amount of work to introduce a degree of linguistic variability we used Large Language Models (LLMs) to create different versions of these instructions. Initially, for each of the 128 possible combinations, a base sentence was manually drafted, the Figure 4.4 illustrates several examples.

Examples of Multi-Attribute Conditioning and Prompt Structure

Configuration I: Cross-lingual Conditioning (Catalan → English)

- **Attributes:** Src Lang: ca → Tgt Lang: en | NERs: 2 | Length: Short
- **Generated Prompt:** "Extreu les idees principals breument en anglès incloent NER1 i NER2."

Configuration II: Monolingual Expansion (Spanish → Spanish)

- **Attributes:** Src Lang: es → Tgt Lang: es | NERs: 0 | Length: Long
- **Generated Prompt:** "Crea un resumen muy completo de esta noticia."

Configuration III: Cross-lingual Expansion (English → Catalan)

- **Attributes:** Src Lang: en → Tgt Lang: ca | NERs: 3 | Length: Model Choice
- **Generated Prompt:** "Translate and summarize the article into Catalan explicitly mentioning NER1, NER2, and NER3."

Figure 4.4: Illustrative examples of multi-attribute conditioning setups. The figure demonstrates how various structural attributes are synthesized into specific natural language prompts tailored to instruct the model.

After making the 128 basic sentences (one for each mix of rules) by hand, we sent a request to the Gemini 3.1 Pro model. This prompt detailed the creation logic and provided the initial sentences as examples so the model can understand this process and write 9 new sentences for each set. Consequently, a total of 10 sentences were obtained for every possible combination. To further increase variability, random manual modifications were introduced into certain sentences, such as substituting terms with synonyms or altering the syntactic order of the elements.

4.2.2. Model Input Structure

For the version that uses natural language text sentences, we use the exact same setup as the token method: dual conditioning. The instructive sentence followed by the text to be summarized is fed into the encoder; for example: "Write a brief summary of this text in Spanish including Messi and Barcelona: [Text to summarize]". In the same way, we also put this exact instruction sentence at the start of the decoder to guide it while the model writes the summary.

4.3 Generation of Conditional Prefixes

As explained in previous sections, the system controls text generation based on three main features. Even though the final dataset is ready, there is a substantial difference between possessing the dataset and having the specific training pairs that will be used to fine-tune the model. We need to decide the language, the named entities (NERs), and the target length for each sample.

The first step is to create a logical rule to clearly set these features for every article-summary pair. It is important to note that this process works no matter which setup is chosen. Whether the conditioning is executed via sentences or custom tokens, the primary difference is that, in the token approach, when a condition is met, the newly created custom token is directly selected (e.g., <SUMMARIZE-IN-CATALAN>). On the other hand, when using sentences, a small code (like "ca") is saved with the data, which is later used to find the right sentence. The charts and visualizations in this document are represented using the token architecture, as it is more visually intuitive and convenient to indicate the specific token rather than the complete sentence fragment.

Besides the three conditioning instructions, the system also needs to pick which translation tool to use (OPUS or NLLB). The first step in creating the prefix consists of applying a random sampling method; in this instance, four distinct selections are made. The first determines the text's language via a variable with four equiprobable values, the first three values are the languages in the dataset (Catalan, Spanish, and English), and the fourth option means the original text decides the final language. You can see how this choice works in Figure 4.5.

For the second conditioning, rather than a uniform random sampling, a binary decision equivalent to a coin toss is applied, to decide if we will include named entities. In this instance, there are two options, each with a 50% probability: the first option lets the model decide freely which entities to put in the summary. The second option forces the model to use a specific list of entities.

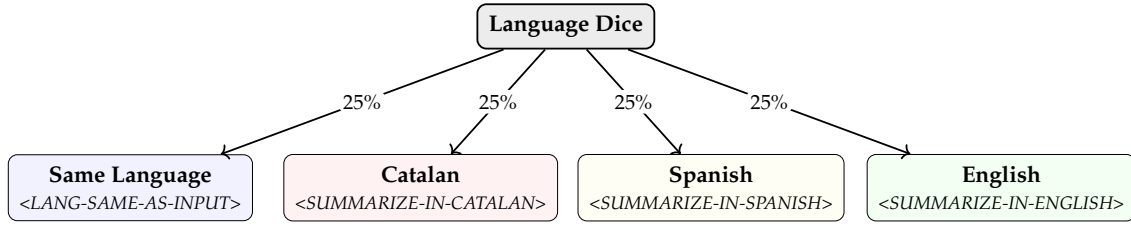


Figure 4.5: Probability distribution of the language selection mechanism and its corresponding target system tokens for multilingual conditioning.

If the random choice picks the first option, the NER setup is finished, and we move on to the next part of the instruction. However, if we get the second option, the conditioning for this block must be further developed. Initially, a check is performed to obtain the number of available entities in the reference summary. If the number is greater than zero, we make another random choice to decide exactly how many entities to put in the instruction. This choice has three options (1, 2, or 3 entities), each with about a 33% chance.

If there are no entities at all in the text, a special rule is forced to tell the model not to use any NERs. In the token-based architecture, this is achieved by introducing the `<NO-NER>` token. For the sentence-based architecture, explicit natural language restrictions are included to induce this behavior; for instance: "[...] without relying on predefined entities such as people, locations, or dates".

It should be noted that even if the random sampling determines a number bigger than the actual number of NERs in the summary, we always cut it down. The system will always pick the smaller number between the entities we actually have and the number the random choice asked for. All this NER selection logic is shown in the Figure 4.6.

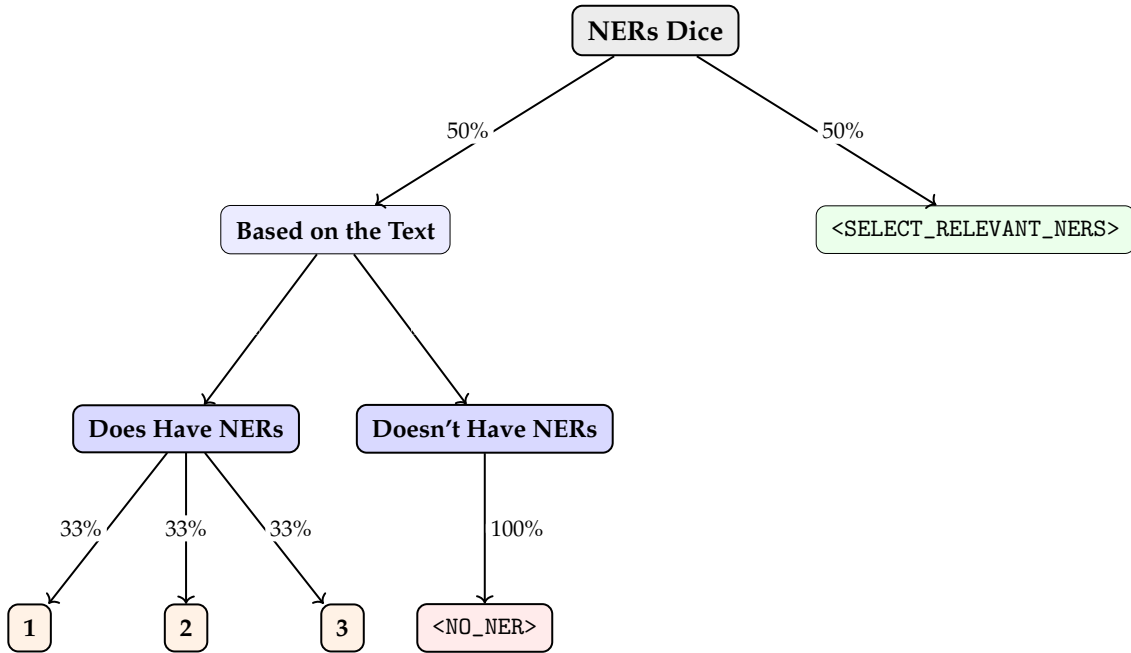


Figure 4.6: Probability decision tree of the NER routing mechanism.

The third dice determine the conditioning parameter that condition the length. We use a binary decision identical to the one used for named entities (NERs) is applied, with a 50% probability for each of the two options. The first choice lets the model pick the

summary's length on its own. If we use tokens, we add the `<DECIDE-SUMMARY-LENGTH>` token. If we use normal sentences, we just do not mention the text length at all.

When the second option is obtained, the length conditioning is not decided via a new random sampling (as occurred with NERs), but is instead empirically determined by the sample's compression ratio. If this ratio is strictly less than 0.1, the summary is classified as short; if it is equal to or greater than 0.1, it is considered long. Based on this, we insert either the `<SHORT-LENGTH>` or the `<LARGE-LENGTH>` token. When using natural language sentences, we explain out the instruction, like *"Write a long summary of this text [...]"*

The selection of the 0.1 threshold is not randomly; It is based on a study we did earlier on how much the texts in our dataset are compressed, which is documented and analyzed in Section 3.2.2. The length dice logic is visualized in Figure 4.7

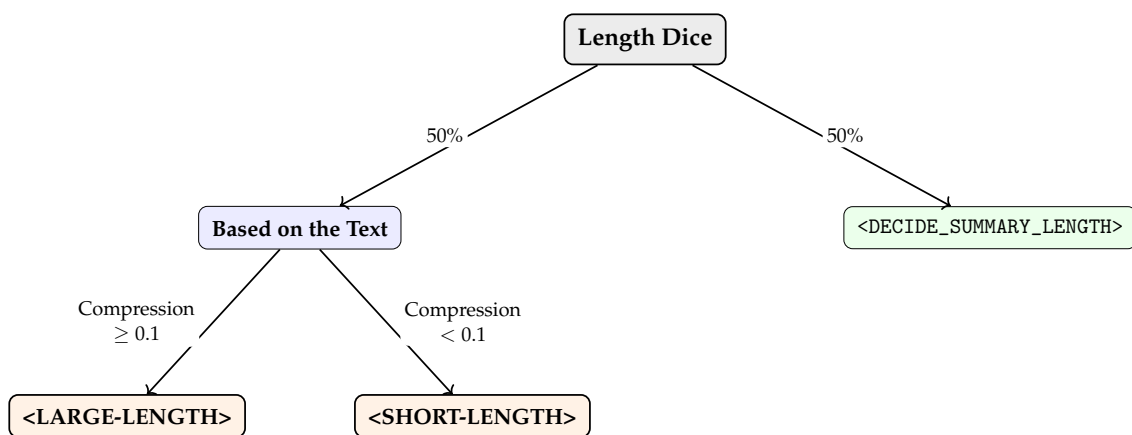


Figure 4.7: Probability decision tree for the summary length routing mechanism.

Finally, the system picks the translation model used for the summary. Even though we make a 50/50 random choice between NLLB and OPUS for every sample, this choice is not used every single time. That is because broadly speaking, 25% of the samples will default to the same language option, thus requiring no translation; in addition to this percentage we have to add the samples where the selected target language coincides with the original language.

Furthermore, during the dataset cleaning phase, certain translations were already discarded, meaning that for specific samples only one of the two models is available. When this happens, the system will always use the available model, ignoring the random choice. In summary, the stochastic selection of the translation model is only effectively applied when the selected target language differs from the original and, simultaneously, both translations (NLLB and OPUS) are available within the dataset.

CHAPTER 5

Training and Fine-Tuning Strategy

To ensure the transparency of this research and guarantee the reproducibility of the experiments, it is paramount to clearly define the framework under which this work was conducted.

5.1 Environment and Hardware Setup

For the realization of this thesis, we have full access to a dedicated workstation provided by the ELiRF (Enginyeria del Llenguatge Natural i Reconeixement de Formes) research group. This machine was equipped with an Intel Core i9-10940X processor (14 cores, 28 threads) clocked at 3.30 GHz, 128 GB of main system RAM, and with two NVIDIA GeForce RTX 3090 GPUs, each featuring 24 GB of VRAM. This local hardware setup was leveraged for the majority of the project’s general data processing pipelines and intermediate computations. Such as the Name Entity Recognition (NER) tasks for the creation of the dataset or the semantic similarities across the dataset.

However, due to the massive volume of the training dataset and the extensive memory requirements, the 24 GB VRAM limit of the RTX 3090s proved insufficient for the actual training phase. Consequently, the core fine-tuning process was offloaded to a more robust infrastructure, a machine of the Valencian Research Institute for Artificial Intelligence (VRAIN) computation cluster, featuring an NVIDIA A40 GPU with 48 GB of VRAM. The training procedure required approximately 170 hours of uninterrupted computation for each experimental approach. And in compliance with Green AI principles and to transparently document the environmental impact of this research, an analysis of the energy consumption and carbon footprint was conducted. The total training time for both architectural approaches amounted to 340 hours. While the NVIDIA A40 GPU has a maximum Thermal Design Power (TDP) of 300 W, empirical monitoring during the training phase registered an average power draw of approximately 170 W. The direct energy consumption of the GPU was approximately $E_{dir} = 0.17kW \times 340h = 57.8kWh$. In a realistic data center environment, the energy consumed by the GPU is only a fraction of the total power required. Ancillary components, such as cooling systems, memory, and CPU overhead, must be accounted for using the Power Usage Effectiveness (PUE) metric. Adopting a conservative and standard PUE value of 1.2 for modern efficient data centers, the total energy consumption is: $E_{tot} = 57.8kWh \times 1.2 = 69.36kWh$. In terms of operational expenses, this represents an approximate cost of between 8 and 15 euros, contingent upon the energy supplier.

In order to translate the energy consumption into an environmental impact metric, it is necessary to consider the average carbon intensity (I_c) of the electrical grid. Based on the Spanish electrical grid data (Comisión Nacional de los Mercados y la Competencia (CNMC), [n.d.](#)), the average emission factor is estimated at 0.258 kg CO₂ per kWh. So this means that:

$$C_{fp} = 69.36kWh \times 0.258 \frac{kgCO_2}{kWh} = 17.895kgCO_2$$

where C_{fp} represents the total carbon footprint, quantifying the greenhouse gas emissions associated with the model's energy consumption.

To contextualize the economic value of the resources provided by ELiRF, we made an estimation of the equivalent cost on a commercial Cloud. Assuming market rate (C_h) of \$1.22 per hour for an on-demand NVIDIA A40 instance (RunPod Inc., [2026](#)) the total economic cost (C_{eco}) of the training infrastructure is:

$$C_{eco} = 340h \times 1.22\$/h = \$414.80$$

It is important to emphasize that this estimate exclusively accounts for the hardware utilization during the final fine-tuning phase. If the extensive hours of computation performed on the dual RTX 3090 workstation were factored in, the cumulative economic cost of the project would be substantially higher.

Regarding the software ecosystem, the entirety of the project's was developed in Python (Van Rossum & Drake Jr, [1995](#)). The core deep learning architecture and model fine-tuning processes were implemented using PyTorch, which served as the foundational framework. To manage and interact with Large Language Models (LLMs), the project heavily relied on the Hugging Face ecosystem; specifically, the transformers library for loading state-of-the-art architectures, peft (Mangrulkar et al., [2022](#)) to implement efficient training strategies like LoRA, and datasets (Lhoest et al., [2021](#)) for optimized data handling and preprocessing.

Furthermore, the Natural Language Processing (NLP) pipelines were built upon the spaCy library. Finally, general-purpose data manipulation, scientific computing, and metric evaluations were orchestrated using industry-standard libraries such as pandas (McKinney, [2010](#)), NumPy (Harris et al., [2020](#)), scikit-learn (Pedregosa et al., [2011](#)) and evaluate (Von Werra et al., [2022](#)). This robust software stack ensured both the efficiency of the training procedures and the full reproducibility of the experimental results.

5.2 Data Preparation and Augmentation

With the objective of providing the model with a more robust approach and improving its generalization capabilities. Given the multiple combinations available for the different instructions (language, NERs, and length), applying a single, static prefix per training sample would result in a significant underutilization of the available data. To avoid, this we generate three distinct, randomly sampled prefixes were generated for each original text.

Then, the effective size of the training dataset per epoch was expanded from 1.5 million to 4.5 million different samples. Furthermore, to guarantee the strict reproducibility of these experiments, acknowledging that computational stochasticity is inherently

pseudo-random, the initialization seeds for all random number generators were explicitly fixed and recorded.

5.3 Training Hyperparameters

In order to properly compare the two architectures and evaluate which system provides superior conditioning capabilities, we decided to standardize all training hyperparameters. This ensures that any observed differences in model quality are strictly attributable to the architectural design rather than the training configurations.

The selection of these hyperparameters was configured with the objective of balancing convergence stability and computational efficiency. First, to save memory usage while maintaining good performance, we choose to use 16-bit floating-point precision (bfloat16). For the batch size, we decided to put a value of 8 samples per device to maximize GPU utilization and achieve a more time-efficient training phase. This batch size was paired with 4 gradient accumulation steps, resulting in an effective batch size of 32 samples per iteration. The entire training process spanned a total of 7 epochs over the training dataset.

In terms of learning dynamics and regularization, a learning rate of 1×10^{-4} was employed, guided by a cosine learning rate scheduler with no warmup ratio. To avoid overfitting of the language model, we applied a weight decay factor of 0.01 and a label smoothing factor of 0.1. Finally, the evaluation phase was scheduled every 30,000 steps, utilizing the ROUGE-mean metric, explained in the 6.1.1 to determine and automatically save the checkpoint with the best empirical performance. A structured view of this hyperparameters are exposed in Table 5.1

Hyperparameter	Configured Value
Base Model	google/mt5-base
Precision	bfloat16
Per-device batch size	8
Gradient accumulation steps	4
Effective batch size	32
Number of epochs	7
Learning Rate	1×10^{-4}
Scheduler	Cosine
Weight Decay	0.01
Label Smoothing	0.1
Evaluation steps	30,000

Table 5.1: Summary of the global training hyperparameters.

5.4 Low-Rank Adaptation (LoRA)

Given the enormous memory requirements to realize a Full Fine-Tuning, we applied the Low-Rank Adaptation (LoRA) technique. This strategy freezes the pre-trained weights of the original model and injects trainable low-rank decomposition matrices into specific layers. In this approach, we decided not to restrict LoRA solely to the attention matri-

ces; instead, we apply LoRA to all linear layers of the architecture. This decision provides the model with greater expressive capacity to capture complex language dependencies in conditional generation tasks.

For the LoRA hyperparameters we used a rank (r) of 32, a relatively high value that provides sufficient breadth to learn new lexical distributions without overwhelming memory resources. A scaling factor α of 64 was assigned, maintaining the ratio recommended of setting alpha to twice the rank ($2r$) to stabilize gradients. Finally, to ensure specific regularization of these new subnetworks, no additional biases were included (`bias="none"`), and a dropout probability of 5% (`lora_dropout=0.05`) was applied to the adapter weights.

5.5 Prefix Masking Strategy

There is a minor yet important implementation detail in the training process that is worth mentioning. If the conditioning prefix and the source text were passed to the model without any changes the model would attempt to learn to predict the instruction prefix. Since this prefix is selected randomly it is impossible for the model to predict it accurately, which would consequently induce an erroneous penalty in the loss function.

To address this issue, we utilized a masking technique by assigning the special value of -100 to the labels corresponding to the prefix tokens and prefix sentence. This specific value is recognized by PyTorch's `CrossEntropyLoss` implementation as an index to be ignored (The PyTorch Team, 2026). In this way, we guarantee that the optimization process focuses exclusively on generating the target text, allowing the model to effectively learn to follow the control instructions.

CHAPTER 6

Evaluation Methodology

This chapter addresses model evaluation, a fundamental phase in this project, as it allows for an objective quantification of the viability of the proposed architectures. Our evaluation framework is structured around three distinct categories of metrics: first, classical metrics focused on the lexical quality of the generated text; second, metrics assessing the fidelity and the conditioning quality; and finally, the implementation of an *LLM-as-a-Judge* approach to evaluate four fundamental dimensions of the generated output: coherence, consistency, fluency, and relevance.

6.1 Evaluation Metrics

6.1.1. Text Quality Metrics (Linguistic and Semantic Evaluation)

This section describes the metrics used to evaluate the intrinsic quality of the generated text, encompassing both lexical precision and semantic equivalence with respect to the reference text.

Lexical Overlap-Based Metrics

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** This metric measures the exact overlap of n-grams between the model-generated summaries and the reference summaries, determining the model’s capacity to retain the vocabulary and basic structure of the ground truth. For our evaluation, we employ the ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest common subsequence) variants. The scores go from 0 to 1, where 1 means the generated text is exactly the same as the reference (Lin, 2004).
- **SacreBLEU:** This metric evaluates the proportion of exact n-gram matches (ranging from unigrams to 4-grams) between the generated text and the reference. It also includes a penalty for shortness to prevent the architecture from achieving artificially high scores by generating excessively short outputs. The results go from 0 to 100, and a higher score means the text is much closer to what is expected (Post, 2018).
- **METEOR (Metric for Evaluation of Translation with Explicit ORdering):** This metric is considered a more reliable than BLEU, because it doesn’t just look for exact word matches it also incorporates alignments based on morphological roots (stemming) and synonym dictionaries. It computes the harmonic mean of unigram precision and recall. It yields a score ranging from 0 to 1, where a value of 1 denotes the best possible outcome (Banerjee & Lavie, 2005).

- **chrF / chrF++:** This metric evaluates overlap at the character n-gram level. While chrF relies solely on character n-grams, chrF++ extends this approach by incorporating word unigrams and bigrams. This makes it very reliable at handling minor spelling differences, like slight verb conjugations or plural endings. The resulting scores range from 0 to 100, where 100 represents the optimal alignment with the reference (Popović, 2015).

Meaning-Based and AI Metrics

- **BERTScore:** This metric determines semantic equivalence. It helps us check if our summary or simplified text says the same thing as the original text or the human reference, even if it uses completely different words. This is achieved through transformer models based on the BERT architecture, such as *xlm-roberta-base*¹. These are used to extract contextual embeddings for each sequence. Subsequently, the cosine similarity between tokens is computed to establish an optimal alignment. Scores are typically rescaled to fall within the [0, 1] interval (although theoretically ranging from -1 to 1), where higher values denote greater semantic similarity (T. Zhang et al., 2019).
- **BARTScore:** This metric numerically evaluates the fluency, faithfulness, and coherence of the generated text. It treats the evaluation as a text generation problem, leveraging the weights of a generative model such as *mbart-large-cc25-dacsa-ca*². Instead of measuring distance metrics in vector spaces, it computes the log-likelihood of the generated sequence being a direct continuation or transformation of the original source text. The scores are expressed as negative log-probabilities, where values closer to 0 indicate superior quality (Lewis et al., 2019).
- **COMET (Crosslingual Optimized Metric for Evaluation of Translation):** This metric utilizes a cross-lingual transformer-based architecture that simultaneously processes the original source text, the generated output, and the human reference. It extracts the best features from all of them and inputs them into a system trained on thousands of real human evaluations. The resulting scores range from 0 to 1 where higher values indicate superior perceived human quality (Rei et al., 2020).

Conditioning Metrics

Until now, the metrics we have looked at only evaluate text quality, completely ignoring the conditioning, which forms the foundation of this work. Consequently, this section assesses the model's capacity to adhere to explicit constraints and instructions. We basically focus on the three different areas: the language, the use of named entities (NERs), and the text length.

- **Language Identification:** We evaluate the model's capacity to adhere to language instructions, formulating it as a simple Boolean condition (True/False). It verifies whether the language of the generated text matches the specified target language. This detection is performed using the `langdetect` Python library³. The resulting detection is compared against the target label from the dataset.
- **NER Recall:** This metric determines the model's capacity to extract and preserve named entities (persons, organizations, locations...) within the generated text. To

¹Model available at: <https://huggingface.co/FacebookAI/xlm-roberta-large>

²Model available at: <https://huggingface.co/ELiRF/mbart-large-cc25-dacsa-ca>

³Library information available at: <https://github.com/Mimino666/langdetect>

obtain the NE we implement complex regular expressions that allow for a margin of error based on edit distance, thereby conducting fuzzy matching operations. This evaluation analyzes text accuracy across three distinct tolerance thresholds: 100%, 95%, and 90% of character alignment. This small margin ensures the model is not penalized if it translates a name into another language instead of keeping it exactly the same. The score is expressed as a ratio from 0 to 1, representing the percentage of retained entities relative to those requested in the conditioning.

- **Length and Compression Conditioning:** Finally, we check if the model respects the requested text length. This calculation is performed by extracting the exact character compression ratio between the original source and the generated text. Based on this ratio, the output is structurally categorized; for example, if the prompt requests a short text and the model generates an output with a compression rate below 0.1, the condition is satisfied. Just like the language identification, this is evaluated as a strict Boolean condition (True/False). This is an essential aspect to guarantee the model's utility in production environments with spatial or reading-time limitations.

6.2 LLM-as-a-Judge

While traditional metrics that measure lexical overlap and semantic evaluation models provide a numerical quantification of performance, it is undeniable that they exhibit structural limitations at capturing the real flow and meaning of human language. For example, a high ROUGE score does not necessarily guarantee a fluent or grammatically correct text, it only means a high vocabulary co-occurrence. Therefore, while these metrics can help to establish a quantitative baseline, relying exclusively on static metrics proves insufficient to ensure exceptional model quality.

To overcome these limitations, in this project we also use the *LLM-as-a-Judge* paradigm (Li et al., 2024). This approach uses massive decoder-only models to act as expert evaluators, judging text quality based on specific rubrics provided via prompts. We give this model a specific grading rubric in its prompt, and it autonomously scores the generated text from 1 (very bad) to 5 (excellent) across four key quality dimensions:

- **Coherence:** This metric evaluates the structural quality and logical flow of the generated text. It evaluate the model's capacity to produce a unified, well-organized text with smooth transitions, thereby preventing the output from becoming a mere compilation of isolated facts. **Evaluation Methodology:** The judge model acts like a professional editor, analyzes the generated text independently of the source document. It identifies the central thesis and verifies whether the arguments follow a coherent logical sequence. It strictly assesses the utilization of logical connectors, pronouns, and transition words to ensure that the narrative progresses without abrupt discursive leaps.
- **Consistency:** This metric measures the strict factual alignment between the generated text and the original source document. The main goal is to catch and strongly penalize any hallucinations or made-up information. **Evaluation Methodology:** The transformer acts like an *fact-checker*, extracting all the claims from the source document and breaks down the generated summary sentence by sentence. If fabricated elements, incorrect names or added data are not present in the source, the score is drastically reduced in direct proportion to the severity of the hallucination.

- **Fluency:** This metric evaluates the linguistic quality of the generated output, looking specifically at grammar, spelling, and whether the sentences flow naturally. **Evaluation Methodology:** Adopting the role of a linguist and copyeditor, the judge model reviews the concordance issues and sentences that are cognitively difficult to process. The highest score (5) is exclusively reserved for entirely error-free texts that read naturally and elegantly, virtually indistinguishable from human-authored prose.
- **Relevance:** This metric measures the generative model’s capacity to perform an effective content distillation. A good summary should only contain the main ideas, without any extra fluff, repetition, or useless details. **Evaluation Methodology:** The judge model preliminarily identifies between 2 and 4 essential aspects of the original document. Then, it inspects the generated summary to confirm the presence of these key points. Finally, it calculates the ratio between vital information and superfluous text: a summary that contains the key facts but is overwhelmed by unnecessary details will have its score significantly penalized.

6.2.1. Open G-Eval

Reviewing the recent literature on LLM-as-a-Judge models, the G-Eval paper stands out (Y. Liu et al., 2023). This paper utilizes massive models, specifically GPT-3.5 and GPT-4, to generate detailed evaluation steps and form-based rubrics, achieving a high correlation with human judgments. However, while they are accurate, using these closed, paid models goes entirely against the goals of our work. Relying on external, paid APIs compromises the privacy and confidentiality of the evaluated data and entirely precludes scientific reproducibility.

For these reasons, the use of commercial solutions has been discarded in favor of open-source model technologies. We set up an experiment comparing several state-of-the-art open models. The goal of this experimentation is to determine which of these alternatives can match the baseline performance and correlations of G-Eval across the four evaluated dimensions (coherence, consistency, fluency, and relevance).

The following subsection details the experimental methodology, assessing which model is capable of achieving correlations comparable to those obtained in the original G-Eval study.

Comparative Study of the Judge Model

To find the best open-source model to act as our judge, an experimental framework was designed based on the *SummEval* dataset (Fabbri et al., 2021). This dataset contains news-summaries pairs from *CNN/DailyMail* that have already been manually graded by three human experts on coherence, consistency, fluency, and relevance. The selection process was structured into two consecutive methodological phases: the generation of probabilistic scores and the computation of statistical alignment.

Phase 1: Implementation of the Weighted G-Eval Calculation

Using the *SummEval* dataset, for each news-summary pair, a prompt was applied to compute a score ranging from 1 to 5 for each of the four evaluated dimensions. In

order to get a more reliable and accurate representation of the judge model’s internal assessment, rather than executing multiple inferences on the same text and averaging the results, the model was constrained to generate a single token. The specific token generated is inherently irrelevant, as we directly extract the log-probabilities from the prediction distribution. We filter out everything except the valid score tokens ('1', '2', '3', '4', '5') and extract the probabilities of generating each specific value. These probabilities are then normalized so that their sum equals 1. Finally, the expected value is calculated by applying Equation 6.1, where i represents the score value and p_i denotes its normalized probability:

$$Score = \sum_{i=1}^5 (i \times p_i) \quad (6.1)$$

This procedure transforms a discrete decision into a continuous and highly precise score. Not only does it allow for capturing the uncertainty and subtle nuances of the evaluator model, but it also yields significant savings in computational resources, time, and energy by eliminating the need to execute multiple iterations over the same text.

Phase 2: Human Alignment Computation

Once the continuous scores are obtained, we need to check how well they match the human ground truth. The model’s predictions are paired with the annotations provided by each of the three human experts across the 1,600 samples comprising the dataset. Finally, two non-parametric correlation metrics are employed: Spearman’s rank correlation coefficient (ρ) and Kendall’s rank correlation coefficient (τ). These metrics evaluate exactly how closely our AI judge’s criteria align with real human reasoning.

Justification of the Selected Model

We tested this procedure on three open-weights models of varying families and capacities: *gemma4-31b*⁴, *qwen3.6-27b*⁵, and *gpt-oss-20b*⁶. To ensure scientific rigor and a fair comparison, these three models were evaluated using the exact same prompt. These prompts are available in Appendix D.

On top of that, it was decided to incorporate the *M-Prometheus* model⁷ (Pombal et al., 2025) into the experimentation. Given that this architecture is specifically designed to perform multilingual evaluation tasks, the project’s standard prompts were not utilized. Instead, customized prompts based on the model’s specific requirements were employed. These custom prompts are also available in Appendix E.

Table 6.1 shows the correlation scores (Spearman’s ρ and Kendall’s τ) for each of the four analyzed dimensions. The analysis of these results reveals that the *gemma4-31b* model not only approximates the baseline correlations of the proprietary GPT-3.5 architecture documented in the original G-Eval study, but it also outperforms it in key areas like coherence and consistency. Consequently, it is empirically demonstrated that Gemma4 fulfills the necessary requirements to be utilized as an *LLM-as-a-Judge*.

⁴ Available at: <https://ollama.com/library/gemma4:31b>

⁵ Available at: <https://ollama.com/library/qwen3.6:27b>

⁶ Available at: <https://ollama.com/library/gpt-oss:20b>

⁷ Available at: <https://huggingface.co/Unbabel/M-Prometheus-14B>

Model	Coherence		Consistency		Fluency		Relevance	
	ρ	τ	ρ	τ	ρ	τ	ρ	τ
qwen3.6-27b	0.2634	0.2222	0.2421	0.1940	0.2308	0.1847	0.1997	0.1206
gpt-oss-20b	-0.0074	-0.0060	0.0745	0.0594	0.0397	0.0314	0.0618	0.0467
M-Prometheus-14B	0.3451	0.3032	0.1895	0.1851	0.2277	0.2201	0.2365	0.2166
gemma4-31b	0.4714	0.3605	0.4924	0.3985	0.3527	0.2844	0.3821	0.2954
G-Eval-3.5	0.440	0.335	0.386	0.318	0.424	0.347	0.385	0.293

Table 6.1: Human alignment results for the different evaluated models regarding the *SummEval* standard. Spearman (ρ) and Kendall (τ) correlation coefficients are reported for each dimension.

6.3 Quantitative Evaluation (Traditional Metrics)

An evaluation of the three proposed models is conducted by analyzing two distinct sets of metrics. Initially, overlap-based metrics are discussed, subsequently followed by an analysis of semantic-level metrics.

6.3.1. Custom Tokens Conditioning

Table 6.2 presents the lexical evaluation metrics for the custom tokens conditioning approach. Overall, the token-based model establishes a baseline of modest scores in exact word matches (ROUGE, SacreBLEU). For instance, in the *newsroom* subset, the model obtains a ROUGE-1 score of 0.2968. Furthermore, performance is slightly lower on *test_ni* subset compared to the *test_i* data.

Partition	rouge1		rouge2		rougeL		sacrebleu		meteor		chrf		chrf++	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.2968	0.2578	0.1866	0.1532	0.2695	0.2317	15.1753	11.3963	0.2903	0.2406	36.7897	31.6564	33.9959	28.9585
dacsa/catalan	0.2545	0.2574	0.1090	0.1175	0.2141	0.2138	7.6028	6.9878	0.2150	0.1902	28.9113	26.1541	26.2726	23.8637
dacsa/spanish	0.2872	0.2843	0.1408	0.1449	0.2449	0.2394	10.5909	8.6603	0.2485	0.2067	32.2258	28.3165	29.6400	26.1055
all	0.2816	0.2624	0.1442	0.1474	0.2434	0.2309	10.9758	10.3756	0.2505	0.2285	32.5273	30.3956	29.8786	27.8332

Table 6.2: Lexical evaluation metrics for the custom token-based conditioning approach. Performance is reported on *test_i* and *test_ni* using ROUGE, SacreBLEU, METEOR, and ChrF variants.

Partition	bertscore_fidelity_f1		bertscore_similarity_f1		bartscore_fidelity		bartscore_similarity		comet	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.8446	0.8463	0.8783	0.8660	-3.4324	-3.3127	-5.7303	-5.7996	0.6021	0.5665
dacsa/catalan	0.8483	0.8453	0.8721	0.8687	-2.3129	-2.5752	-3.8233	-3.8096	0.6124	0.5895
dacsa/spanish	0.8503	0.8514	0.8777	0.8714	-2.7737	-2.7903	-5.6451	-5.4086	0.6194	0.5900
all	0.8484	0.8471	0.8765	0.8673	-2.8214	-3.1306	-5.2273	-5.4871	0.6135	0.5734

Table 6.3: Semantic and embedding-based evaluation metrics for the custom token-based conditioning approach. Performance is evaluated on *test_i* and *test_ni* using F1-scores from BERTScore, alongside BARTScore and COMET metrics.

However, these relatively low n-gram overlap scores do not necessarily correlate with a degradation in overall generative quality. As demonstrated in Table 6.3, the semantic and embedding-based metrics—such as the BERTScore F1 values for fidelity and similarity—remain consistently high (around 0.84 and 0.87, respectively). This indicates that while the token-based model may select a different vocabulary or rephrase sentences to meet the imposed constraints, it successfully preserves the underlying meaning and deep semantic structure of the original text.

Partition	ner_recall		ner_perc_95		ner_perc_90		language_acc		compression_acc	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.9532	0.9411	0.9594	0.9498	0.9639	0.9556	0.9228	0.9057	1.0000	1.0000
dacsa/catalan	0.9629	0.9643	0.9655	0.9675	0.9683	0.9722	0.9762	0.9120	1.0000	1.0000
dacsa/spanish	0.9566	0.9524	0.9607	0.9567	0.9662	0.9611	0.9972	0.9238	1.0000	1.0000
all	0.9572	0.9460	0.9615	0.9531	0.9661	0.9586	0.9742	0.9395	1.0000	1.0000

Table 6.4: Linguistic properties, entity preservation, and constraints evaluation for the custom token-based conditioning approach. Performance is measured on *test_i* and *test_ni* via NER recall, percentiles, language accuracy, and compression correctness.

A close look at Table 6.4, which assesses the level of rule adherence, shows some inconsistencies in the token-based modeling performance. Firstly, it should be noted that the compression accuracy is absolutely consistent (1.0000), while entities are preserved quite effectively, with NER Recall exceeding 0.94. However, language recognition appears to be more challenging. Language accuracy sits at 0.9742 for *test_i* cases and drops to 0.9395 for the not in-domain test partition. In other words, the usage of customized tokens seems to negatively affect the language choice made by the model.

6.3.2. Natural Language Sentence Conditioning

Compared to the custom tokens approach discussed previously, the natural language sentence-based conditioning exhibits a substantial improvement in exact n-gram overlap. As presented in Table 6.5, traditional lexical metrics such as ROUGE, SacreBLEU, and METEOR show significantly higher scores. For instance, the ROUGE-1 score in the *test_i* *newsroom* subset rises from 0.2968 (in the token-based model) to 0.4280. However, the model still indicates a performance discrepancy between the in-domain (*test_i*) and out-of-domain (*test_ni*) data splits, dropping to 0.3738 in the latter. While the *dacsa* partitions (both Catalan and Spanish) have lower overall scores than *newsroom*, their performance remains much more stable across both splits. ChrF and ChrF++ corroborate this trend, showing a notable drop in the *newsroom* split while maintaining strong morphological stability across the out-of-domain *dacsa* subsets.

Partition	rouge1		rouge2		rougeL		sacrebleu		meteor		chrf		chrf++	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.4280	0.3738	0.2782	0.2309	0.3822	0.3294	21.8546	16.5436	0.4082	0.3423	44.7632	38.6734	42.6463	36.4738
dacsa/catalan	0.3217	0.3183	0.1409	0.1468	0.2672	0.2594	9.3908	8.4351	0.2685	0.2327	32.0299	28.8517	29.7601	26.8452
dacsa/spanish	0.3584	0.3574	0.1832	0.1889	0.3024	0.2971	13.4265	11.3841	0.3080	0.2627	35.7543	32.3313	33.6044	30.4414
all	0.3663	0.3641	0.1959	0.2132	0.3131	0.3152	14.4848	14.6439	0.3226	0.3149	37.0274	36.3560	34.8565	34.2345

Table 6.5: Lexical evaluation metrics for the natural language sentence-based conditioning approach. Performance is reported on *test_i* and *test_ni* using ROUGE, SacreBLEU, METEOR, and ChrF variants.

Partition	bertscore_fidelity_f1		bertscore_similarity_f1		bartscore_fidelity		bartscore_similarity		comet	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.8370	0.8395	0.8925	0.8794	-3.6639	-3.5749	-5.0331	-5.1722	0.6448	0.6072
dacsa/catalan	0.8421	0.8403	0.8786	0.8748	-3.5675	-3.6230	-4.5952	-4.4531	0.6282	0.6060
dacsa/spanish	0.8428	0.8449	0.8832	0.8783	-3.3548	-3.3704	-5.1448	-4.9363	0.6361	0.6090
all	0.8412	0.8405	0.8843	0.8787	-3.4804	-3.5449	-4.9857	-5.0427	0.6363	0.6074

Table 6.6: Semantic and embedding-based evaluation metrics for the natural language sentence-based conditioning approach. Performance is evaluated on *test_i* and *test_ni* using F1-scores from BERTScore, alongside BARTScore and COMET metrics.

Partition	ner_recall		ner_perc_95		ner_perc_90		idioma_acc		compression_acc	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.9511	0.9383	0.9575	0.9481	0.9622	0.9554	0.9930	0.9880	1.0000	1.0000
dacsa/catalan	0.9706	0.9692	0.9737	0.9738	0.9768	0.9778	0.9946	0.9926	1.0000	1.0000
dacsa/spanish	0.9619	0.9579	0.9665	0.9617	0.9716	0.9660	0.9946	0.9926	1.0000	1.0000
all	0.9611	0.9456	0.9658	0.9536	0.9703	0.9600	0.9942	0.9894	1.0000	1.0000

Table 6.7: Linguistic properties, entity preservation, and constraints evaluation for the natural language sentence-based conditioning approach. Performance is measured on *test_i* and *test_ni* via NER recall, percentiles, language accuracy, and compression correctness.

In addition to the improved lexical overlap, the semantic and embedding-based metrics detailed in Table 6.6 highlight that the natural language approach maintains the robust semantic integrity observed in the previous model. BERTScore F1 values for fidelity and similarity remain consistent, averaging approximately 0.84 and 0.88 respectively across all subsets. Furthermore, the COMET metric, which strongly correlates with human judgment, yields global scores exceeding 0.60 for both splits. Finally, BARTScore fidelity and similarity metrics show minimal fluctuation, confirming this semantic robustness beyond exact token matching.

Finally, Table 6.7 illustrates the model’s adherence to structural and linguistic constraints, revealing a crucial advantage over the token-based approach. The model achieves an average Named Entity Recognition (NER) Recall above 0.94 and maintains a perfect compression accuracy (1.0000). Most notably, the natural language conditioning resolves the language identification issues observed in the previous model, achieving a nearly flawless language selection accuracy (≥ 0.98) across all partitions. This confirms that the model can strictly follow structural rules and language constraints simultaneously while generating high-quality text.

6.3.3. Qwen3.6 Prompting

Before proceeding with the analysis of the metrics in this subsection, a methodological clarification is required. Even though the Qwen3.6:27b model was not discussed during the preliminary architectural design and training phases, it was considered relevant to run it and include it in this evaluation phase. The primary objective of this addition is to establish a rigorous baseline, thereby allowing for the validation and benchmarking of the custom architecture proposed in this study against a state-of-the-art Large Language Model (LLM). The prompts used to execute the model are available in Appendix F.

Partition	rouge1		rouge2		rougeL		sacrebleu		meteor		chrF		chrF++	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.2482	0.2411	0.0869	0.0856	0.1727	0.1704	4.2588	4.2703	0.2722	0.2677	32.4212	31.5536	29.5758	28.7904
dacsa/catalan	0.2252	0.2644	0.0717	0.0880	0.1608	0.1791	3.2315	4.1088	0.2570	0.2613	30.6012	33.9476	27.8634	30.8030
dacsa/spanish	0.2393	0.2915	0.0775	0.1044	0.1673	0.1979	3.5246	5.2966	0.2617	0.2698	31.5014	35.3235	28.7502	32.3367
all	0.2381	0.2528	0.0784	0.0892	0.1671	0.1763	3.6341	4.4308	0.2631	0.2673	31.5119	32.5091	28.7409	29.6598

Table 6.8: Lexical evaluation metrics for the Qwen3.6:27b prompting approach. Performance is reported on standard and non-standard test splits (*test_i* and *test_ni*) using ROUGE, SacreBLEU, METEOR, and ChrF variants.

An analysis of the results obtained with the prompting strategy of the Qwen3.6:27b model highlights the unique nature of its performance compared to previous iterations. According to Table 6.8, the lexical metrics (ROUGE, SacreBLEU, METEOR, ChrF) demon-

Partition	bertscore_fidelity_f1		bertscore_similarity_f1		bartscore_fidelity		bartscore_similarity		comet	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.8314	0.8368	0.8565	0.8543	-5.3111	-5.2175	-6.5419	-6.5531	0.6576	0.6448
dacsa/catalan	0.8548	0.8520	0.8589	0.8618	-3.5708	-3.6925	-4.8952	-4.7501	0.6619	0.6809
dacsa/spanish	0.8479	0.8513	0.8598	0.8639	-5.1278	-5.2497	-6.6069	-6.5086	0.6561	0.6677
all	0.8455	0.8412	0.8588	0.8569	-4.8027	-5.0363	-6.1846	-6.3244	0.6578	0.6532

Table 6.9: Semantic and embedding-based evaluation metrics for the Qwen3.6:27b prompting approach. Performance is evaluated on in-domain and not in-domain test splits (*test_i* and *test_ni*) using F1-scores from BERTScore, alongside BARTScore and COMET metrics.

Partition	ner_recall		ner_perc_95		ner_perc_90		language_acc		compression_acc	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
newsroom	0.9032	0.8788	0.9156	0.8935	0.9272	0.9076	0.9453	0.9437	0.9213	0.8901
dacsa/catalan	0.8574	0.8494	0.8791	0.8706	0.9052	0.8960	0.7655	0.7649	0.9712	0.9512
dacsa/spanish	0.8861	0.8768	0.9016	0.8909	0.9200	0.9074	0.8279	0.8240	0.8871	0.9133
all	0.8841	0.8749	0.9002	0.8903	0.9185	0.9061	0.8417	0.9007	0.9265	0.9182

Table 6.10: Linguistic properties, entity preservation, and constraints evaluation for the Qwen3.6:27b prompting approach. Performance is measured on in-domain and not in-domain test splits (*test_i* and *test_ni*) via NER recall, percentiles, language accuracy, and compression correctness.

strate more consistent performance when comparing the test in-domain set and the not in-domain one, with fewer performance drops compared to the custom token-based model. However, the absolute number of n-grams preserved is less than that obtained using the sentence-based prompting method, thus revealing that LLMs used with prompting often prioritize generative diversity over n-gram repetition.

Furthermore, the results regarding the semantic quality (Table 6.9) are impressive. Specifically, the BERTScore F1 measures obtain highly stable values, with their maximum being 0.8618 on the dacsa/catalan split. It should be noted that even though the BARTScore values are somewhat lower (more negative) compared to the previous models, COMET metric demonstrates excellent performance, scoring above 0.68 in the dacsa Catalan subset. This means that the answers provided by the model are not only fluent but also semantically relevant, thus being considered valuable for the evaluation system.

Last but not least, Table 6.10 examines how well the answers respect the given constraints. Even though this approach allows for an effective entity preservation (NER Recall is approximately 0.87), it demonstrates poor performance when it comes to strict constraint following, in particular, when it comes to compression quality (from 0.88 to 0.97). In comparison with the previous models, whose compression was perfect, the proposed method depends on instructiveness of the model, which, even though it is quite fluent, lacks mathematical consistency.

6.4 Advanced Qualitative Evaluation (LLM-as-a-Judge)

In addition to using the existing automated measures, which fail to completely address the semantic value of the text and its naturalness, the following section provides the advanced qualitative analysis through the LLM as Judge paradigm. Gemma4:31b was used as the impartial judge of the results.

Model	Coherence		Cohesion		Fluency		Relevance	
	test_i	test_ni	test_i	test_ni	test_i	test_ni	test_i	test_ni
Sentence-Based	4.4906	4.3845	4.6830	4.7191	2.9844	3.6904	4.1514	4.0899
Token-Based	4.5657	4.4832	4.7528	4.7206	3.3212	4.0157	4.2219	4.1933
Qwen3.6:27b	4.9172	4.9087	4.8329	4.9079	4.1857	4.3421	4.7566	4.8457

Table 6.11: Qualitative evaluation using an LLM-as-a-Judge approach. Performance is measured across Coherence, Cohesion, Fluency, and Relevance for the *test_i* and *test_ni* data splits.

Due to limitations of time and computation, it was impossible to perform a qualitative analysis on the whole test dataset. Therefore, only a sample portion is used. This is through a method called stratified random sampling where the sample selected is evenly distributed according to the target languages and source domains without deviation from the actual data distribution.

It is evident from the analysis of the scores in Table 6.11 that the vast majority of parameters have very high ratings, especially Coherence, Cohesion, and Relevance, which often exceed 4.5 out of 5. Such a systematization suggests a possible "generous" attitude of the evaluator model towards its assessments.

However, a consistent trend appears; Fluency invariably gets the lowest ratings, especially for the encoder-decoder architectures. For instance, the natural language conditioning model is estimated at only 2.9844 in the *test_i* split. It reveals a difficulty of these models to generate absolutely fluent texts under strict instructions.

It is important that, despite the potential exaggeration of the actual qualities of the texts by the evaluator model, the bias is applied to all of them equally. Therefore, the comparative assessment remains reliable. Indeed, according to the results in Table 6.11, the differences between Qwen3.6:27b (a relatively large and general-purpose language model) and the two fine-tuned models are surprisingly minor in some categories such as Coherence and Cohesion. It is a rather significant discovery since it confirms the possibility of developing competitive small-scale architectures for specific tasks through proper fine-tuning.

6.5 Comparative Discussion and Optimal Model Selection

The comprehensive evaluation of the proposed architectures reveal valuable conclusions regarding the existing practices in the field of Natural Language Generation. First, it must be mentioned that one of the key findings from the analysis of the results is that a purposefully designed fine-tuning process for a small (580 million parameters) encoder-decoder architecture like mT5-base, may outperform, or at least effectively compete with a state-of-the-art Large Language Model such as Qwen3.6:27b with 27 billion parameters.

The qualitative evaluation (LLM-as-a-Judge) and semantic embeddings indicate that the large LLM still has a marginal advantage in terms of overall fluency and general semantic similarity. Nevertheless, the decrease in quality related to fluency is significantly offset by the high precision of instruction fulfillment by smaller architectures. The main objective for this thesis is the strict conditional generation and without any doubt is ac-

accomplished more efficiently using the fine-tuned models. As the metrics demonstrate, the customized architectures achieve a perfect compression accuracy (1.0000) and superior entity preservation (NER Recall > 0.94), whereas the Qwen3.6:27b model struggles with mathematical consistency in constraints, showing compression accuracies fluctuating between 0.88 and 0.97, and lower NER preservation (around 0.87). Therefore, the results of this research disprove the popular industrial notion that "bigger is always better." An architecturally optimal design coupled with proper data preparation and processing appears to be more advantageous compared to resource-intensive LLMs and provides an opportunity for democratization of Artificial Intelligence with much lower hardware costs.

When directly comparing the two fine-tuning strategies, the natural language sentence conditioning approach might initially appear superior. It presents a simpler architecture with minimal performance degradation during conditional generation. However, a critical methodological limitation must be explicitly acknowledged: the evaluation relies entirely on prompts that are identical to those utilized during the training phase. Although a degree of linguistic variability was intentionally introduced during the creation of these training sentences, it is arguably insufficient to guarantee robustness in real-world, out-of-distribution applications. This reliance introduces a significant risk of prompt dependency, meaning the model's conditioning effectiveness could be severely compromised by unforeseen variations in the phrasing of the input prompt. To avoid this vulnerability, the custom token conditioning approach could be used. By utilizing explicit control tokens, it provides a stricter and highly reliable conditioning mechanism that is entirely independent of natural language ambiguity or input phrasing, thereby ensuring deterministic and robust generation; however, this strict structural adherence inherently sacrifices the flexibility typically associated with free-form natural language prompting.

Therefore, taking into consideration the strict operational requirements of controlled text generation, the Custom Token Conditioning architecture is selected as the optimal model. This architecture provides the most robust and deterministic control mechanism for language and compression constraints, eliminates the prompt-dependency vulnerability inherent to natural language conditioning, and achieves exceptional semantic preservation, all while operating within a highly efficient and accessible parameter footprint.

CHAPTER 7

Paper for the CLEF 2026 SimpleText Track

This chapter summarizes our participation in the SimpleText shared-task (Ermakova et al., 2026) at the CLEF 2026 international conference. For this participation, we used the same text-generation conditioning strategy developed in this work, but applied to a completely different domain: the simplification of biomedical texts. The results obtained, fourth place in the group ranking and several runs among the best in the competition, support the suitability of the proposed approach for the task.

The results of our participation are detailed in a paper, which has already been accepted for publication. The preprint of the paper has been included as the Appendix B

7.1 Introduction

The extensive specialized vocabulary and complex structures of biomedical literature make it difficult for non-expert audiences to understand the meaning of the text, leading to a barrier that can promote misinformation. To mitigate this, automated scientific text simplification has gained substantial traction. The recent state-of-the-art approaches, including the top-performing systems in last year’s SimpleText competition, have been heavily dominated by prompting massive, private Large Language Models (LLMs). Even though these approaches are effective, their reliance on closed APIs limits reproducibility and incurs high computational, energy, and water costs. Beyond simple prompting, different techniques allow small open models to achieve competitive, reproducible results locally. Specifically, input-level conditioning guides generation by injecting explicit control tokens into the source sequence, whereas model-level conditioning adapts the internal architecture to inherently process these constraints.

Our methodology relies on a two-module architecture dividing the task into planning and generation. An initial module predicts specific simplification actions, which are then processed by a generation module using either advanced prompting techniques or a combination of fine-tuning and Direct Preference Optimization (DPO) applied to encoder-decoder models, (Rafailov et al., 2023) .

A core principle of our approach is its alignment with Green AI guidelines. Instead of relying on massive, proprietary models with high computational, energy, and water

costs, we advocate for the use of efficient and optimized open models. By doing so, we aim to democratize access to controlled text generation technologies, demonstrating that it is possible to achieve highly competitive simplification systems that can run in local environments while minimizing environmental impact.

7.2 Tasks and Datasets

Specifically, we present our proposal for Task 1 (Simplify Scientific Text), addressing the challenge by providing results for both sentence-level (Task 1.1) and document-level simplification (Task 1.2). (Bakker et al., 2026)

Task 1 focuses on the simplification of biomedical literature, utilizing abstracts and their corresponding Plain Language Summaries (PLS) from Cochrane systematic reviews. The objective is to make highly technical medical texts accessible to non-expert audiences. To achieve this, Task 1.1 asks systems to simplify individual "shredded" sentences extracted from these abstracts. Conversely, Task 1.2 requires the simplification of entire abstracts as a single document, with the constraint that the generated simplification must be readable on its own and its overall message must remain consistent with the conclusions of the full review.

The evaluation of the current edition is carried out on the official 2026 test set. This new set presents a dataset divided into 11 different languages, with a total of 48,809 sentences for task 1.1 and 8,069 documents. However, the main evaluation is based on the newly published Cochrane abstracts from the last year. Specifically, this core evaluation subset focuses exclusively on the English versions and is structurally restricted to the "Main results" and "Authors' conclusions" sections of both the abstracts and their paired plain language summaries.

Despite the characteristics of this new 2026 test set, due to the absence of a new official training corpus for this edition, our system has been trained using the Cochrane-auto dataset bakker-kamps-2024-cochrane. This corpus consists of 1,085 documents, structured into 4,171 paragraphs and 14,719 sentences, all in English. An example of the data is shown in the Table 7.1

The main strength of this corpus is that it includes the label of the specific transformation (the action) applied to each source sentence during the simplification process. The possible actions are: Delete, Copy, Rephrase, Split, Merge.

It is important to highlight a methodological modification that we have introduced to the reference labels of the corpus. While the original *Cochrane-auto* dataset defines the merge blocks, the sequence of sentences to be joined with the labels Merge and None, we have decided to change the labeling scheme. Specifically, we assign the <Merge_B> label to the first sentence of a block and the <Merge_I> label to the remaining sentences of the same merge block. We implemented this change because it provides a much clearer and more understandable representation of the sequence boundaries.

Sentence-level example (Task 1.1)	
Complex	The one included trial suggests no evidence of an effect of fat supplementation of human milk on short-term growth and feeding intolerance in preterm infants.
Action	<Split>
Simple	Addition of extra fat to human milk for preterm infants showed no clear benefits with regards to short-term rates of weight gain, length gain, and head growth. There was no evidence that the extra fat increased the risk of feeding intolerance.
Document-level example (Task 1.2)	
Complex	'This review includes a total of 1897 children and 729 adults in 39 trials.', 'Thirty-three trials were conducted in the emergency room and equivalent community settings, and six trials were on inpatients with acute asthma (207 children and 28 adults).', 'The method of delivery of beta2-agonist did not show a significant difference in hospital admission rates.', 'In adults, the risk ratio (RR) of admission for spacer versus nebuliser was 0.94 (95% CI 0.61 to 1.43).', 'The risk ratio for children was 0.71 (95% CI 0.47 to 1.08, moderate quality evidence).', 'In children, length of stay in the emergency department was significantly shorter when the spacer was used.', 'The mean duration in the emergency department for children given nebulised treatment was 103 minutes, and for children given treatment via spacers 33 minutes less (95% CI -43 to -24 minutes, moderate quality evidence).', 'Length of stay in the emergency department for adults was similar for the two delivery methods.', 'Peak flow and forced expiratory volume were also similar for the two delivery methods.', 'Pulse rate was lower for spacer in children, mean difference -5% baseline (95% CI -8% to -2%, moderate quality evidence), as was the risk of developing tremor (RR 0.64; 95% CI 0.44 to 0.95, moderate quality evidence).'
Action	<Rephrase> → <Rephrase> → <Rephrase> → <Delete> → <Delete> → <Rephrase> → <Split> → <Split> → <Delete> → <Rephrase>
Simple	'We found 39 clinical trials involving 1897 children and 729 adults.', 'Thirty-three of the trials were conducted in an emergency room (or emergency department) and community settings (such as a GP's surgery), and six trials were on inpatients (people in hospital) with acute asthma (207 children and 28 adults).', 'Taking beta-agonists through either a spacer or a nebuliser in the emergency department did not make a difference to the number of adults being admitted to hospital, whilst in children we can be fairly confident that nebulisers are not better than spacers at preventing admissions.', 'In children, the length of stay in the emergency department was significantly shorter when the spacer was used instead of a nebuliser.', 'The average stay in the emergency department for children given nebulised treatment was 103 minutes.', 'Children given treatment via spacers spent an average of 33 minutes less.', 'In adults, the length of stay in the emergency department was similar for the two delivery methods.', 'However the adult studies were conducted slightly differently which may have made it more difficult to show a difference in the length of stay in the emergency department.', 'Pulse rate was lower in children taking beta-agonists through a spacer (mean difference -5% baseline), and there was a lower risk of developing tremor.'

Table 7.1: Examples of sentence-level and document-level simplification extracted from the Cochrane-auto corpus, illustrating the transformation flow from complex to simple text.

7.3 System description

7.3.1. Architecture

To carry out this task, we proposed an architecture based on two distinct modules: an initial planning module responsible for predicting the specific simplification action for each sentence, followed by a generation module that uses these predicted actions to guide the final text production. For this generation module, we explored and compared two different strategies. On the one hand, we apply advanced prompting techniques on the Gemma 4 model by directly injecting the predicted actions into its context. On the other hand, we used our conditional text generation via tokens combined with DPO. This approach allows the models to inherently integrate and process these explicit instructions during the simplification process. The architecture schema is detailed in Figure 7.1

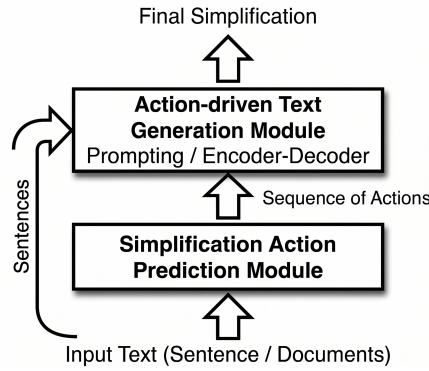


Figure 7.1: Architecture Diagram of the Text Simplification System

As established in the collaborative framework of this research, it is pertinent to clarify that the Simplification Action Prediction module was developed by co-authors of the SimpleText publication. Its operational logic is summarized in this subsection exclusively to provide a holistic overview of the joint pipeline and to properly contextualize the scope of the submitted paper.

7.3.2. Simplification Action Prediction

In the context of the SimpleText26 shared task, we framed the action prediction as a sentence-level sequence-labeling over the 5 editing actions defined by the shared task: Copy, Delete, Rephrase, Split, and Merge (split into two – Merge_B and Merge_I). Given a document, the system predicts exactly one action per input sentence, so that the full output is a structurally valid sequence of edits that can be executed downstream.

Architecture

The system that we propose aims to tackle the following task: *given a document, the system should decide which action for simplification should be done for each sentence*. The possible actions are the following ones: $\mathcal{A} = \{\text{Copy}, \text{Delete}, \text{Merge_B}, \text{Merge_I}, \text{Rephrase}, \text{Split}\}$, with $C = |\mathcal{A}| = 6$. A document is a sequence of N sentences $\mathbf{s} = (s_1, \dots, s_N)$ and the model assigns exactly one action $y_i \in \mathcal{A}$ to each sentence. The two-tag pair Merge_B/Merge_I encodes the multi-sentence merge groups annotated in the dataset in IOB style, so an arbitrary-length merge span can be described without enlarging the label set. Sentences are concatenated with a dedicated separator token `<sent>` before being passed to the

encoder:

$$\mathbf{x} = [\langle s \rangle, \langle \text{sent} \rangle, s_1, \langle \text{sent} \rangle, s_2, \dots, \langle \text{sent} \rangle, s_N, \langle /s \rangle].$$

The underlying model is a hybrid architecture built around a shared RoBERTa Y. Liu et al., 2019b, large version, encoder f_θ that processes the full document in a single pass, so that every sentence representation benefits from cross-sentence attention:

$$H = f_\theta(\mathbf{x}) \in \mathbb{R}^{L \times d}.$$

A learnable sentence pooler π then collapses, for each sentence i , the contextual states between its ' $\langle \text{sent} \rangle$ ' marker and the next separator into a single vector

$$\mathbf{h}_i = \pi(H, i) \in \mathbb{R}^{d_p}, \quad i = 1, \dots, N.$$

On top of this representation, we attach three specialist binary branches, indexed by $b \in \mathcal{B} = \{\text{delete}, \text{copy}, \text{rephrase}\}$, that probe three orthogonal semantic axes – “*is this information relevant?*”, “*is this readable as-is?*” and “*does this need rewriting?*”. Each branch consists of a low-dimensional projection $W_b^{\text{proj}} \in \mathbb{R}^{d_b \times d_p}$ and a binary classifier $W_b^{\text{cls}} \in \mathbb{R}^{2 \times d_b}$:

$$\mathbf{z}_i^{(b)} = W_b^{\text{proj}} \text{sg}(\mathbf{h}_i), \quad \ell_i^{(b)} = W_b^{\text{cls}} \mathbf{z}_i^{(b)} \in \mathbb{R}^2,$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator that decouples the binary objectives from the shared pooler. Each branch is trained against the one-vs-rest indicator $t_i^{(b)} = \mathbb{1}[y_i = y_b^*]$ derived from the dataset’s gold actions, with binary cross-entropy,

$$\mathcal{L}_b = -\frac{1}{N} \sum_{i=1}^N \log p(t_i^{(b)} | \ell_i^{(b)}).$$

The pooled vector and the (detached) branch projections are concatenated and passed through a linear fusion head to produce per-sentence emission scores over the C actions:

$$\mathbf{u}_i = [\mathbf{h}_i; \text{sg}(\mathbf{z}_i^{(\text{delete})}); \text{sg}(\mathbf{z}_i^{(\text{copy})}); \text{sg}(\mathbf{z}_i^{(\text{rephrase})})], \quad \mathbf{e}_i = W^{\text{emit}} \text{Dropout}(\mathbf{u}_i) \in \mathbb{R}^C.$$

The stop-gradients between the branches and the fusion layer fully decouple the specialist objectives from the main classifier: the binary heads provide stable auxiliary features without being reshaped by the main loss, while the fusion head and pooler are shaped only by the document-level objective.

The emissions are then passed through a linear-chain CRF that scores a full sequence with a learnable transition matrix $T \in \mathbb{R}^{C \times C}$ and boundary potentials $\boldsymbol{\pi}, \boldsymbol{\tau} \in \mathbb{R}^C$,

$$\Psi(\mathbf{y} | \mathbf{e}) = \pi_{y_1} + \sum_{i=1}^N e_{i, y_i} + \sum_{i=2}^N T_{y_{i-1}, y_i} + \tau_{y_N}.$$

Structural validity is enforced by masking forbidden transitions and boundaries to $-\infty$,

$$T_{a,b} = -\infty \quad \forall (a,b) \in \mathcal{F}, \quad \pi_a = -\infty \quad \forall a \in \mathcal{F}_{\text{start}}, \quad \tau_a = -\infty \quad \forall a \in \mathcal{F}_{\text{end}},$$

where $\mathcal{F}$, $\mathcal{F}_{\text{start}}$ and $\mathcal{F}_{\text{end}}$ encode the IOB rules implied by the SimpleText26 annotation scheme — Merge_I may only follow Merge_B or Merge_I, Merge_B must be followed by Merge_I, Merge_I cannot begin a document and Merge_B cannot end one — so that any decoded sequence corresponds to a well-formed group of merges by construction. The total training loss combines the CRF negative log-likelihood, an auxiliary class-weighted cross-entropy on the emissions that compensates the heavy imbalance observed in the

dataset between frequent (Copy, Delete) and rare (Split, Merge_*) actions, and the three branch BCE terms:

$$\mathcal{L}_{\text{CRF}} = -\Psi(\mathbf{y}^* | \mathbf{e}) + \log \sum_{\mathbf{y} \in \mathcal{A}^N} \exp \Psi(\mathbf{y} | \mathbf{e}), \quad \mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^N w_{y_i^*} \log \text{softmax}(\mathbf{e}_i)_{y_i^*},$$

$$\mathcal{L} = \mathcal{L}_{\text{CRF}} + \alpha \mathcal{L}_{\text{CE}} + \sum_{b \in \mathcal{B}} \lambda_b \mathcal{L}_b.$$

For our final prediction system, we train $S = 5$ independent replicas with different random seeds and combine them at inference time with an emission-level ensemble. Each replica produces pre-CRF emissions $\mathbf{e}_i^{(s)}$ and CRF parameters $(T^{(s)}, \pi^{(s)}, \tau^{(s)})$, which are averaged across seeds:

$$\bar{\mathbf{e}}_i = \frac{1}{S} \sum_{s=1}^S \mathbf{e}_i^{(s)}, \quad \bar{T} = \frac{1}{S} \sum_{s=1}^S T^{(s)}, \quad \bar{\pi} = \frac{1}{S} \sum_{s=1}^S \pi^{(s)}, \quad \bar{\tau} = \frac{1}{S} \sum_{s=1}^S \tau^{(s)}.$$

The averaged parameters are loaded into a fresh linear-chain CRF $\bar{\Psi}$ that re-applies the structural masks to $\bar{T}$, $\bar{\pi}$ and $\bar{\tau}$. Averaging at the emission level rather than voting on hard labels preserves the relative confidence between actions, while averaging the transitions keeps the structural prior consistent across replicas. Because the RoBERTa encoder is limited to $L_{\text{max}} = 512$ sub-word tokens, long test documents are processed with a sliding-window scheme: sentences are pre-tokenised and greedily packed into windows $W_k = [a_k, b_k)$ with $\sum_{i=a_k}^{b_k-1} (1 + |s_i|) + 2 \leq L_{\text{max}}$, and consecutive windows share an overlap of $\Delta = 4$ sentences so that border sentences still receive sufficient left and right context. The ensemble emissions of each sentence are accumulated across all windows in which it appears and averaged,

$$\bar{\mathbf{e}}_i = \frac{1}{|\mathcal{W}_i|} \sum_{k \in \mathcal{W}_i} \bar{\mathbf{e}}_i^{(k)}, \quad \mathcal{W}_i = \{k : a_k \leq i < b_k\},$$

and a single constrained Viterbi pass under $\bar{\Psi}$ over $(\bar{\mathbf{e}}_1, \dots, \bar{\mathbf{e}}_N)$,

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{A}^N} \bar{\Psi}(\mathbf{y} | \bar{\mathbf{e}}),$$

yields one globally consistent action sequence per document, independent of where window boundaries fall with respect to the encoder budget.

Simplification Action Prediction Performance

We evaluate the action prediction sub-task with the per-class F1-score, reported per action and as the unweighted macro average. The macro variant gives the same weight to every action and is therefore sensitive to the rarer classes (Split, Merge_B, Merge_I) that dominate the label imbalance of the provided dataset, while the weighted average reflects the overall sentence-level accuracy of the system on the empirical class distribution. Table ?? reports the per-action and aggregate F1-scores of our system.

Two observations stand out. First, the ranking of actions is consistent across the two splits and in line with the distribution of the dataset: Rephrase and Delete – the two most frequent classes, and the ones most clearly anchored in the input by the specialist branches – are predicted markedly better than the structurally constrained Merge_B/Merge_I pair, whose support is small and whose prediction additionally depends on the IOB transitions being correctly aligned along the sequence. Second, the macro F1 is essentially

Action	Validation	Test
Copy	0.293	0.253
Delete	0.489	0.440
Merge_B	0.090	0.115
Merge_I	0.165	0.123
Rephrase	0.548	0.534
Split	0.132	0.226
Macro	0.286	0.282
Weighted	0.458	0.434

stable between validation (0.286) and test (0.282), and the weighted F1 only drops mildly (0.458 $\rightarrow$ 0.434), which indicates that the system generalises without a noticeable validation/test gap; the most visible movement is on Split, whose F1 actually improves on the test split (0.132 $\rightarrow$ 0.226), and on Delete, which drops by about five points. The gap between the macro F1 (which gives equal weight to every action) and the weighted F1 (which reflects the empirical class distribution) further quantifies how much of the system’s overall accuracy comes from correctly handling the frequent actions, and how much headroom remains on the rarer ones.

7.3.3. Action-driven Text Generation

Once the actions are predicted, they are used to explicitly guide the final text generation process. To accomplish this, we detail two primary generation strategies.

LLM Prompting Approach

For our first generation strategy, we apply advanced prompting techniques to the open-weight model Gemma 4 (31B). This model was chosen because its relatively compact size enables local inference, perfectly aligning with our Green AI principles by avoiding reliance on private APIs and significantly reducing the energy footprint of the simplification process.

The prompt design is firmly grounded in the *National Institutes of Health* (NIH) guidelines for written health materials, ensuring that the generated text targets approximately an eighth-grade US reading level, appropriate for the general public. Crucially, the text generation is explicitly guided by the output of our initial planning module. To achieve this, the System Prompt is structured to define the six available simplification actions (<Split>, <Copy>, <Delete>, <Rephrase>, <Merge_B>, and <Merge_I>).

During the inference phase, the User Prompt supplies the model with the complete complex text to provide global context. This is immediately followed by a sequential breakdown of the text, where each source sentence is strictly paired with the exact simplification action predicted by the planning module. The complete design of both the System Prompt and the User Prompt utilized for this strategy is detailed in Appendix C.

Fine-tuning Approach

This approach is based on our custom architecture with special tokens, which is explained in Section 4.1. The only change is the amount of tokens we use. For the Simpletext task, we add the following: <Copy>, <Rephrase>, <Delete>, <Split>, <Merge_B>, and

`<Merge_I>`. In addition to action tokens, we defined a special token indicating the beginning of a sentence, `<Sentence>`.

Model Input

For the first subtask, simplification at the sentence level, there are two actions that do not require reasoning to perform the simplification action. The first is the copy action; when we detect that it has been predicted, we simply copy the complex text as the prediction. The second, the delete action, also receives special treatment: it is simply deleted and an empty string is passed as the prediction.

The input to the model remains almost identical: actions are included first, and the only difference lies in the utilization of the `<Sentence>` token to separate each sentence. For instance, an input sequence could be: "`<Split> <Sentence>` This review includes a total of 1897 children and 729 adults in 39 trials." And for the decoder is identical we use the `decoder_inputs_ids` to introduce the actions.

Regarding the second subtask, at the document level, the input preparation requires a more complex procedure given the limitations of the model's context window (512 tokens). To process complete documents without truncating relevant information, we apply a sequential clustering and block processing strategy, divided into two phases:

First, we establish "atomic or indivisible units"; this means that if the prediction module outputs a sequence of sentences to be combined, the `<Merge_B>` and `<Merge_I>` segments are immediately clustered. This guarantees that they are never split across different inference blocks, as they must be evaluated jointly. Second, these clusters are iteratively assembled to form larger text blocks. We use the tokenizer to continuously monitor the total block length, ensuring it stays within a safety margin to prevent truncating sentences, which would otherwise degrade performance.

Once the block is formed, the dual conditioning strategy is applied to multiple sequences. For the encoder, we concatenate all the predicted actions for the block followed by the original sentences, where each sentence is preceded by its respective marker. In this way, an input for a three-sentence block could have this structure:

```
<Split><Rephrase><Delete> <Sentence> Complex sentence 1. <Sentence>Complex
sentence 2. <Sentence> Complex sentence 3.
```

Direct Preference Optimization

After the initial supervised fine-tuning phase, a manual inspection of the generated simplifications revealed that the model frequently introduced lexical modifications even when the explicitly provided action was `<Copy>`. Conversely, it tended to leave the original sentence unchanged when the tag required a rewrite (`<Rephrase>`). To mitigate this behavior and enforce stricter adherence to the control instructions, we introduced a secondary training phase leveraging DPO. The algorithm requires a dataset formed by triplets (x, y_w, y_l) , where x is the input (the complex text along with the actions), y_w is the preferred response (chosen), and y_l is a rejected response. To build this corpus in an au-

tomated manner adapted to our task, we designed a generation and evaluation pipeline composed of two phases:

1. Candidate Generation and Evaluation (Dataset Construction). For each training instance, we use the previously trained SFT model to generate four different candidates. To ensure the syntactic diversity of these hypotheses, we apply a generation strategy based on *Diverse Beam Search*, configured with 4 beam groups and a diversity penalty of 1.5. Once the candidates are generated, we proceed to evaluate them to determine which represents the preferred output and which the rejected one. This scoring is based on two pillars:

- **Alignment Model (CRF-BERT):** Each candidate is evaluated using a sequential alignment model (based on BERT and CRF) that scores the quality of the alignment between the original text and the generated simplification. The choice of this model is not arbitrary; it is the exact same alignment system developed and used by the creators of the original Cochrane-auto corpus. Therefore, it serves as the most reliable and appropriate metric for assessing the quality of simplifications within this specialized dataset domain.
- **Control Correction Heuristics:** To resolve the specific problems observed in SFT, we apply strict rules to the base score. We severely penalize those candidates that maintain information when the tag is <Rephrase>, or that alter text when the tag is <Copy>, to force the model to perform the required simplification.

The candidate achieving the highest final score is designated as y_w (chosen response) while the remaining candidates are designated as y_l (rejected responses), when their scores fall below that of the best candidate by a predefined minimum margin.

2. DPO Training. Using the newly constructed preference dataset, we initialize the DPO training phase. We configure two models: the active model, whose weights will be updated, and a reference model, whose weights are kept frozen. Both models are initialized with the weights obtained from the SFT phase, which includes the custom embedding matrix designed to accommodate the control tokens. The training process directly optimizes the policy of the active model relative to the reference model via the standard DPO loss function. Through this alignment phase, the model not only learns the characteristics of a high-quality simplification but also actively identifies the patterns that lead to rejected outputs, thereby effectively calibrating its sensitivity to the action tags.

7.4 Results and Discussion

We evaluated our models across three settings: the Cochrane-auto validation set, its internal test set, and the CLEF 2026 SimpleText benchmark. To thoroughly assess performance, the evaluation utilizes three key metrics: SARI (Xu et al., 2016), which measures simplification quality by evaluating the accuracy of words kept, added, or deleted compared to human references; BLEU, which evaluates the general similarity and n-gram overlap between the generated text and the reference texts; and FKGL (Kincaid et al., 1980) (Flesch-Kincaid Grade Level), which assesses readability by indicating the US school grade level required to understand the generated text easily.

Systems	SARI↑		BLEU↑		FKLG↓	
	Val	Test	Val	Test	Val	Test
Gemma4:31b-No_Actions	39.32	39.1	12.69	11.72	9.25	9.43
Gemma4:31b-Action	40.1	38.99	18.77	16.16	10.96	11.15
Flan-T5-No_Actions	34.82	32.04	27.15	28.80	12.83	11.22
Flan-T5-Actions	43.90	39.92	25.59	24.60	11.93	12.54
Flan-T5-DPO_Actions	45.66	42.60	11.58	11.51	18.48	66.65
BART-No_Actions	34.98	33.21	35.19	30.50	13.20	14.13
BART-Actions	39.47	39.09	28.15	27.57	13.66	13.77

Table 7.2: Metrics for SimpleText Document Simplification Dataset

Systems	SARI↑		BLEU↑		FKLG↓	
	Val	Test	Val	Test	Val	Test
Gemma4:31b-No_Actions	38.17	37.43	5.95	5.67	9.1	9.3
Gemma4:31b-Action	49.38	47.08	10.79	9.23	7.7	7.75
Flan-T5-No_Actions	38.52	37.10	24.15	23.80	8.71	9.04
Flan-T5-Actions	48.88	46.68	26.71	24.95	9.66	9.74
Flan-T5-DPO_Actions	55.88	52.78	10.50	10.10	13.77	12.87
BART-No_Actions	37.45	34.63	25.10	23.49	8.39	8.74
BART-Actions	51.38	48.85	14.06	13.25	11.99	11.98

Table 7.3: Metrics for SimpleText Sentence Simplification Dataset

A detailed analysis of Tables 7.2 and 7.3, which present the document and sentence-level results for both the validation and test sets, reveals the performance improvement provided by action-based strategies (Actions) compared to the baseline configurations (No_Actions). This qualitative leap is generalized and strongly evident in both prompting methods and classic encoder-decoder architectures, as the action system manages to guide the models much more precisely.

Comparing the two subtasks clearly shows that sentence-level evaluation produces significantly higher metrics compared to the document level. This phenomenon is primarily due to the success of the Copy and Delete actions predictions, which achieve a perfect score of 100 in SARI, whereas errors are not as heavily penalized due to the action model’s tendency to predict Rephrase.

Finally, the overall evaluation solidifies Flan-T5 as the superior model; therefore, it was selected for fine-tuning using DPO. This additional training positively enhances the results to the point where the Flan-T5-DPO_Actions variant manages to compete closely with Gemma4. In this way, a more compact model is capable of challenging and even outperforming models with a vastly larger number of parameters. However, an unexpected anomaly was observed in the FKGL metric for the Flan-T5-DPO_Actions model on the test set, which recorded an unusually high score of 66.65, whereas the other configurations typically score between 11 and 13. The underlying cause of this outlier is currently being investigated.

Systems	SARI↑	BLEU↑	FKLG↓
Gemma4:31b-No_Actions	46.22	9.79	7.77
Gemma4:31b-Action	45.77	8.65	9.86
Flan-T5-No_Actions	35.52	3.04	10.77
Flan-T5-Actions	36.27	5.47	12.77
Flan-T5-DPO_Actions	37.56	6.23	12.68
BART-Actions	36.80	10.43	12.17

Table 7.4: Metrics for SimpleText 2026 Test Documents

Systems	SARI↑	BLEU↑	FKLG↓
Gemma4:31b-No_Actions	46.36	9.91	7.87
Gemma4:31b-Action	46.35	11.54	9.17
Flan-T5-No_Actions	29.95	16.20	11.93
Flan-T5-Actions	35.72	13.73	11.80
Flan-T5-DPO_Actions	36.90	12.64	12.55
BART-Actions	35.26	11.74	14.29

Table 7.5: Metrics for SimpleText 2026 Test Sentences

Evaluating the results of the official 2026 task, showed in Table 7.4 and Table 7.5 reveals a significant performance drop in both document and sentence simplification compared to prior evaluations. Currently, the exact cause of this decline remains unknown; it will be necessary to wait for the references to be made public in order to analyze and study the reasons for this drop in metrics in depth.

Unlike to previous phases where Flan-T5 achieved the highest results, this new test scenario highlights that its architecture does not generalize as well. While Gemma4, on the other hand, performs significantly better, taking a clear lead in SARI scores with values nearing 46 in both subtasks.

Nevertheless, a consistent, and positive improvement in performance can still be observed when using actions, which outperform the baseline version. Similarly, the application of DPO training also enhances the results.

7.5 Future Work

For future work, we propose several lines of research to continue improving the controlled generation of biomedical text efficiently. On the one hand, we plan to evaluate our action injection strategy on newer open models and explore more advanced prompting techniques to refine control in the generation stage. On the other hand, regarding the Flan-T5 based approach, we propose an evolution of our architectural modification. Currently, structural control is achieved by introducing the simplification action discretely. Alternatively, our goal is to investigate a fuzzy integration of the actions. This soft integration (soft prompting/embeddings) would allow the model to learn the nuances of each action more flexibly.

CHAPTER 8

Conclusions and Future Work

In this project, we have successfully achieved the main objective of developing and evaluating a controllable, computationally efficient conditioned text generation system. By systematically analyzing the state of the art (Objective 1), we established a solid foundation to justify the use of lightweight encoder-decoder architectures over massive autoregressive models for this specific domain.

Then, in order to address the lack of optimal resources for our specific task, we successfully created and adapted a structured trilingual dataset (Objective 2). That allowed us to train and optimize reduced-parameter models using on this corpus (Objective 3) obtaining a highly effective mechanism for conditional text generation. Specifically, regarding the evaluation of hallucination reduction and conditioning (Objective 4), the implementation of custom token conditioning proved superior in enforcing strict structural rules, mitigating the prompt dependency vulnerabilities observed in natural language conditioning, and maintaining high semantic fidelity. By abstracting generation rules into deterministic control variables, the system managed complex generation constraints without requiring the immense parameter scale of current state-of-the-art LLMs.

Furthermore, we also tested the model across the Spanish, Catalan, and English data partitions revealed that the lightweight architectures successfully maintain cross-lingual stability (Objective 5). When we compared our small model against huge models (Objective 6) demonstrates highly competitive performance across multiple metrics. Notably, the compact model successfully matches the larger baselines in general aspects and even outperforms them in specific conditioning constraints. Sometimes the text had to change a bit to follow our strict rules, but the actual meaning stayed exactly the same. This proves our special tokens really understand the meaning of the text, no matter what language is being used.

To prove our system works in the real world, we participated in the CLEF 2026 SimpleText Track (Objective 7). This allowed us to test our small-sized models in an international and standardized evaluation environment, culminating in a research paper that contrasts our system’s efficacy against contemporary proprietary LLMs.

8.1 Future Work

Despite these promising results, there are several avenues for future research worth exploring. First of all, to definitively validate the robustness of the natural language con-

ditioning approach (extending Objective 4), we must evaluate the model using prompt structures distinct from the training prompts. Furthermore, a double-blind human evaluation of the outputs needs to be performed since while scalable automated measures and LLM-as-a-Judge frameworks are undoubtedly convenient, they have their own inherent biases that must be overcome by independent expert evaluation of the outputs.

Lastly, the system's functionality should be enhanced both architecturally and linguistically. Future iterations will explore alternative state-of-the-art lightweight architectures, potentially integrating other parameter-efficient fine-tuning techniques to optimize the training pipeline further. Concurrently, expanding the corpus to encompass a significantly wider variety of languages and textual domains is essential. Scaling the linguistic diversity will not only consolidate the model's cross-lingual generation capabilities but also ensure its practical applicability in diverse, real-world multilingual environments.

Acknowledgements

This work is supported by the Spanish Secretaría de Estado de Digitalización e Inteligencia Artificial (TSI-100930-2023-9). And also, is partially supported by MCIN/AEI/10.13039/501100011033 and ERDF “ERDF A way of making Europe” (project PID2024-155948OB-C55).



Proyecto financiado por Secretaría de Estado de Digitalización e Inteligencia Artificial (TSI-100930-2023-9)



També voldria agrair als meus tutors, Antonio Molina i María José Castro, per ajudar-me durant tota la realització del treball, aconsellar-me i per la confiança depositada en mi.

A més, voldria donar les gràcies a Vicent Ahuir per ajudar-me constantment, guiar-me i facilitar-me la bona realització d'aquest treball.

I, finalment, l'agraïment més especial és per a mon pare, per guiar-me no sols durant tot el treball i la carrera, sinó durant tota la vida, estant sempre disposat a ajudar-me i donar-me suport cost el que cost.

Declaration on Generative AI

During the preparation of this work, the authors used Gemini to translate portions of the initial drafts from Catalan to English, and to rewrite specific fragments to improve the academic tone and overall readability. Additionally, Gemini was employed to generate the image for the architectural visualization, and Gemini 3.1 Pro was utilized to generate source code snippets along with their corresponding explanations. After using these tools, the authors thoroughly reviewed and edited all generated text and code, taking full responsibility for the final content and visual representations in the manuscript. Although Generative AI technologies are advancing rapidly and prove to be highly valuable tools for drafting text and writing code, it remains the strict responsibility of the human authors to ensure their ethical use, critically review all generated outputs, and guarantee that all bibliographical references have been researched, verified, and integrated manually.

Bibliography

- Automatic Language Processing Advisory Committee (ALPAC). (1966). *Languages and machines: Computers in translation and linguistics*. National Academy of Sciences. <https://aclanthology.org/www.mt-archive.info/50/ALPAC-1966.pdf>
- Bakker, J., et al. (2026). Overview of the CLEF 2026 SimpleText task 1: Simplify scientific text. In G. Faggioli et al. (Eds.), *Working notes of the conference and labs of the evaluation forum (CLEF 2026)*. CEUR-WS.org.
- Banerjee, S., & Lavie, A. (2005, June). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In J. Goldstein, A. Lavie, C.-Y. Lin, & C. Voss (Eds.), *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization* (pp. 65–72). Association for Computational Linguistics. <https://aclanthology.org/W05-0909/>
- Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166. <https://doi.org/10.1109/72.279181>
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Chan, G. (2020). Rnn many-to-one architecture [[Accessed: June 3, 2026]]. https://goodboychan.github.io/python/deep_learning/tensorflow-keras/2020/12/06/01-RNN-Many-to-one.html
- Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70), 1–53.
- Comisión Nacional de los Mercados y la Competencia (CNMC). (n.d.). *Presentación*. <https://gdo.cnmc.es/CNMC/accesoEtiquetado.do>
- DeepSeek-AI. (2026). Deepseek-v4: Towards highly efficient million-token context intelligence.
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6), 391–407.
- El-Kassas, W. S., Salama, C. R., Rafea, A. A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165, 113679. <https://doi.org/10.1016/j.eswa.2020.113679>
- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179–211. [https://doi.org/10.1016/0364-0213\(90\)90002-E](https://doi.org/10.1016/0364-0213(90)90002-E)
- Ermakova, L., et al. (2026). Overview of CLEF 2026 SimpleText track: Simplify scientific texts (and nothing more). In M. Hagen et al. (Eds.), *Experimental IR meets multilinguality, multimodality, and interaction. proceedings of the sixteenth international conference of the CLEF association (CLEF 2026)*. Springer-Verlag.
- Exxact Corporation. (2023). A deep dive into the transformer architecture: The development of transformer models [[Accessed: May 21, 2026]]. <https://www.exxactcorp.com/blog/Deep-Learning/a-deep-dive-into-the-transformer-architecture-the-development-of-transformer-models>

- Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R., & Radev, D. (2021). SummEval: Re-evaluating summarization evaluation (B. Roark & A. Nenkova, Eds.). *Transactions of the Association for Computational Linguistics*, 9, 391–409. https://doi.org/10.1162/tacl_a_00373
- GeeksforGeeks. (2025, July). History and evolution of nlp [Accessed: 13 april, 2026].
- Gemini Team. (2026). Gemini 3 pro model card. *Google DeepMind*. <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf>
- Glorot, X., Bordes, A., & Bengio, Y. (2011, November). Deep sparse rectifier neural networks. In G. Gordon, D. Dunson, & M. Dudík (Eds.), *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323, Vol. 15). PMLR. <https://proceedings.mlr.press/v15/glorot11a.html>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [<http://www.deeplearningbook.org>]. MIT Press.
- Goyanes, M., Ardévol-Abreu, A., & de Zúñiga, H. G. (2021). Antecedents of news avoidance: Competing effects of political interest, news overload, trust in news media, and “news finds me” perception. *Digital Journalism*, 11, 1–18. <https://doi.org/10.1080/21670811.2021.1990097>
- Graf, B., & Antoni, C. (2022). Drowning in the flood of information: A meta-analysis on the relation between information overload, behaviour, experience, and health and moderating factors. *European Journal of Work and Organizational Psychology*, 32, 173–198. <https://doi.org/10.1080/1359432x.2022.2118051>
- Grusky, M., Naaman, M., & Artzi, Y. (2018, June). Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)* (pp. 708–719). Association for Computational Linguistics. <https://doi.org/10.18653/v1/N18-1065>
- Gunasekar, S., Zhang, Y., Aneja, J., César Teodoro Mendes, C., Del Giorno, A., Gopi, S., Javaheripi, M., Kauffmann, P., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Singh Behl, H., Wang, X., Bubeck, S., Eldan, R., Tauman Kalai, A., Lee, Y. T., & Li, Y. (2023). Textbooks Are All You Need. *arXiv e-prints*, Article arXiv:2306.11644, arXiv:2306.11644. <https://doi.org/10.48550/arXiv.2306.11644>
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- Heiss, R., Nanz, A., & Matthes, J. (2023). Social media information literacy: Conceptualization and associations with information overload, news avoidance and conspiracy mentality. *Comput. Hum. Behav.*, 148, 107908. <https://doi.org/10.1016/j.chb.2023.107908>
- Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*. <http://arxiv.org/abs/1503.02531>
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9, 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). Spacy: Industrial-strength natural language processing in python. <https://doi.org/10.5281/zenodo.1212303>
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *arXiv e-prints*, Arti-

- cle arXiv:2106.09685, arXiv:2106.09685. <https://doi.org/10.48550/arXiv.2106.09685>
- Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2023). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43, 1–55. <https://doi.org/10.1145/3703155>
- Jegham, N., Abdelatti, M., Koh, C. Y., Elmoubarki, L., & Hendawi, A. (2025). How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference. *arXiv e-prints*, Article arXiv:2505.09598, arXiv:2505.09598. <https://doi.org/10.48550/arXiv.2505.09598>
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Chen, D., Dai, W., Shu Chan, H., Madotto, A., & Fung, P. (2022). Survey of Hallucination in Natural Language Generation. *arXiv e-prints*, Article arXiv:2202.03629, arXiv:2202.03629. <https://doi.org/10.48550/arXiv.2202.03629>
- Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Singh Chaplot, D., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Renard Lavaud, L., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., ... El Sayed, W. (2024). Mixtral of Experts. *arXiv e-prints*, Article arXiv:2401.04088, arXiv:2401.04088. <https://doi.org/10.48550/arXiv.2401.04088>
- Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec & C. Rouveirol (Eds.), *Machine learning: Ecml-98* (pp. 137–142, Vol. 1398). Springer. <https://doi.org/10.1007/BFb0026683>
- Junyan Wang, J., & Xiaoqi Wang, V. (2025). Assessing Consistency and Reproducibility in the Outputs of Large Language Models: Evidence Across Diverse Finance and Accounting Tasks. *arXiv e-prints*, Article arXiv:2503.16974, arXiv:2503.16974. <https://doi.org/10.48550/arXiv.2503.16974>
- Jurafsky, D., & Martin, J. H. (2026). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, with language models* (3rd) [Online manuscript released January 6, 2026]. <https://web.stanford.edu/~jurafsky/slp3/>
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv e-prints*, Article arXiv:2001.08361, arXiv:2001.08361. <https://doi.org/10.48550/arXiv.2001.08361>
- Kincaid, J. P., et al. (1980). *Development and test of a computer readability editing system (CRES)* (Final Report) (June 1978 through December 1979).
- Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *arXiv e-prints*, Article arXiv:1808.06226, arXiv:1808.06226. <https://doi.org/10.48550/arXiv.1808.06226>
- Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, 282–289.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv e-prints*, Article arXiv:1910.13461, arXiv:1910.13461. <https://doi.org/10.48550/arXiv.1910.13461>
- Lhoest, Q., Villanova del Moral, A., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., Davison, J., Šaško, M., Chhablani, G., Malik, B., Brandeis, S., Le Scao, T., Sanh, V., Xu, C., Patry, N., ... Wolf, T. (2021, November). Datasets: A community library for natural language processing. In

- H. Adel & S. Shi (Eds.), *Proceedings of the 2021 conference on empirical methods in natural language processing: System demonstrations* (pp. 175–184). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.emnlp-demo.21>
- Li, D., Jiang, B., Huang, L., Beigi, A., Zhao, C., Tan, Z., Bhattacharjee, A., Jiang, Y., Chen, C., Wu, T., Shu, K., Cheng, L., & Liu, H. (2024). From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv: 2411.16594*.
- Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 74–81. <https://aclanthology.org/W04-1013/>
- Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9). <https://doi.org/10.1145/3560815>
- Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., & Zhu, C. (2023). G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. *arXiv e-prints*, Article arXiv:2303.16634, arXiv:2303.16634. <https://doi.org/10.48550/arXiv.2303.16634>
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019a). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692. <http://arxiv.org/abs/1907.11692>
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019b). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692. <http://arxiv.org/abs/1907.11692>
- Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., Bossan, B., & Tietz, M. (2022). PEFT: State-of-the-art parameter-efficient fine-tuning methods.
- McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. *AAAI-98 workshop on learning for text categorization*, 752(1), 41–48.
- McKinney, W. (2010). Data structures for statistical computing in python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th python in science conference* (pp. 51–56).
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Nie, Z., Arshad, M., Khurshid, S. K., Javed, A., Waheed, T., Zhang, Q., & Shahzad, M. F. (2025). Domain-specific multi-document political news summarization using bart and act-gan. *IEEE Access*, 13, 143737–143750. <https://doi.org/10.1109/access.2025.3598058>
- NLLB Team, Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, A., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Barrault, L., Meja-Gonzalez, G., Hansanti, P., ... Wang, J. (2022). No language left behind: Scaling human-centered machine translation.
- OpenAI. (2026). Introducing gpt-5.2. *OpenAI Official Blog*. <https://openai.com/index/introducing-gpt-5-2/>
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv e-prints*, Article arXiv:1912.01703, arXiv:1912.01703. <https://doi.org/10.48550/arXiv.1912.01703>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Pennington, J., Socher, R., & Manning, C. (2014, October). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of*

- the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. <https://doi.org/10.3115/v1/D14-1162>
- Pombal, J., Yoon, D., Fernandes, P., Wu, I., Kim, S., Rei, R., Neubig, G., & Martins, A. F. T. (2025). M-Prometheus: A Suite of Open Multilingual LLM Judges. *arXiv e-prints*, Article arXiv:2504.04953, arXiv:2504.04953. <https://doi.org/10.48550/arXiv.2504.04953>
- Popović, M. (2015, September). ChrF: Character n-gram F-score for automatic MT evaluation. In O. Bojar, R. Chatterjee, C. Federmann, B. Haddow, C. Hokamp, M. Huck, V. Logacheva, & P. Pecina (Eds.), *Proceedings of the tenth workshop on statistical machine translation* (pp. 392–395). Association for Computational Linguistics. <https://doi.org/10.18653/v1/W15-3049>
- Post, M. (2018, October). A call for clarity in reporting BLEU scores. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, M. Negri, A. Névél, M. Neves, M. Post, L. Specia, M. Turchi, & K. Verspoor (Eds.), *Proceedings of the third conference on machine translation: Research papers* (pp. 186–191). Association for Computational Linguistics. <https://doi.org/10.18653/v1/W18-6319>
- Post, M., & Vilar, D. (2018, June). Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)* (pp. 1314–1324). Association for Computational Linguistics. <https://doi.org/10.18653/v1/N18-1119>
- Q-Success / W3Techs. (2026). Usage statistics and market share of content languages for websites [[Accessed: June 1, 2026]]. https://w3techs.com/technologies/overview/content_language
- Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286. <https://doi.org/10.1109/5.18626>
- Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., & Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36, 53728–53741.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140), 1–67. <http://jmlr.org/papers/v21/20-074.html>
- Rei, R., Stewart, C., Farinha, A. C., & Lavie, A. (2020, November). COMET: A neural framework for MT evaluation. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)* (pp. 2685–2702). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.emnlp-main.213>
- Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. <https://arxiv.org/abs/1908.10084>
- Richards, B. (1987). Type/token ratios: What do they really tell us? *Journal of Child Language*, 14(2), 201–209. <https://doi.org/10.1017/S0305000900012885>
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. <https://doi.org/10.1038/323533a0>
- RunPod Inc. (2026). *Gpu cloud pricing | per-hour a40*. Retrieved June 12, 2026, from <https://www.runpod.io/pricing>

- Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 613–620. <https://doi.org/10.1145/361219.361220>
- Segarra Soriano, E., Ahuir, V., Hurtado, L.-F., & González, J. (2022). DACSA: A large-scale dataset for automatic summarization of Catalan and Spanish newspaper articles. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 5931–5943. <https://doi.org/10.18653/v1/2022.naacl-main.434>
- Siddiq, M. L., Islam-Gomes, A., Sekerak, N., & Santos, J. C. S. (2025). Large Language Models for Software Engineering: A Reproducibility Crisis. *arXiv e-prints*, Article arXiv:2512.00651, arXiv:2512.00651. <https://doi.org/10.48550/arXiv.2512.00651>
- Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1), 11–21. <https://doi.org/10.1108/eb026526>
- Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. *arXiv e-prints*, Article arXiv:1906.02243, arXiv:1906.02243. <https://doi.org/10.48550/arXiv.1906.02243>
- The PyTorch Team. (2026). *Torch.nn.crossentropyloss* — *pytorch 2.12 documentation*. Retrieved June 12, 2026, from <https://docs.pytorch.org/docs/2.12/generated/torch.nn.CrossEntropyLoss.html>
- Tiedemann, J. (2020). The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. *Proceedings of the Fifth Conference on Machine Translation*, 1174–1182. <https://www.aclweb.org/anthology/2020.wmt-1.139>
- Tiedemann, J., & Thottingal, S. (2020, November). OPUS-MT – building open translation services for the world. In A. Martins, H. Moniz, S. Fumega, B. Martins, F. Batista, L. Coheur, C. Parra, I. Trancoso, M. Turchi, A. Bisazza, J. Moorkens, A. Guerberof, M. Nurminen, L. Marg, & M. L. Forcada (Eds.), *Proceedings of the 22nd annual conference of the european association for machine translation* (pp. 479–480). European Association for Machine Translation. <https://aclanthology.org/2020.eamt-1.61/>
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236), 433–460. <https://courses.cs.umbc.edu/471/papers/turing.pdf>
- Universitat Politècnica de València. (2026). Càtedres de l'Estratègia Nacional d'Intel·ligència Artificial (Càtedres ENIA) [Accessed: June 19, 2026].
- Van Rossum, G., & Drake Jr, F. L. (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Von Werra, L., Tunstall, L., Thakur, A., Luccioni, S., Thrush, T., Piktus, A., Marty, F., Rajani, N., Mustar, V., & Ngo, H. (2022, December). Evaluate & evaluation on the hub: Better best practices for data and model measurements. In W. Che & E. Shutova (Eds.), *Proceedings of the 2022 conference on empirical methods in natural language processing: System demonstrations* (pp. 128–136). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2022.emnlp-demos.13>
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E. H., Le, Q., & Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *CoRR, abs/2201.11903*. <https://arxiv.org/abs/2201.11903>
- Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1), 36–45. <https://doi.org/10.1145/365153.365168>

- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., ... Rush, A. (2020, October). Transformers: State-of-the-art natural language processing. In Q. Liu & D. Schlangen (Eds.), *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.emnlp-demos.6>
- Xu, W., Napoles, C., Pavlick, E., Chen, Q., & Callison-Burch, C. (2016). Optimizing statistical machine translation for text simplification (L. Lee, M. Johnson, & K. Toutanova, Eds.). *Transactions of the Association for Computational Linguistics*, 4, 401–415. https://doi.org/10.1162/tacl_a_00107
- Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2020). Mt5: A massively multilingual pre-trained text-to-text transformer. *CoRR*, abs/2010.11934. <https://arxiv.org/abs/2010.11934>
- Zhang, B., & Sennrich, R. (2019). Root mean square layer normalization. *Advances in Neural Information Processing Systems (NeurIPS)*, 32. <https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html>
- Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). BERTScore: Evaluating Text Generation with BERT. *arXiv e-prints*, Article arXiv:1904.09675, arXiv:1904.09675. <https://doi.org/10.48550/arXiv.1904.09675>
- Zoph, B., Raffel, C., Schuermans, D., Yogatama, D., Zhou, D., Metzler, D., Chi, E. H., Wei, J., Dean, J., Fedus, L. B., Bosma, M. P., Vinyals, O., Liang, P., Borgeaud, S., Hashimoto, T. B., & Tay, Y. (2022). Emergent abilities of large language models. *TMLR*.

APPENDIX A

Sustainable Development Goals (SDGs)

A.1 Degree of Relationship of the Work with the Sustainable Development Goals (SDGs)

On September 25, 2015, world leaders adopted a set of global goals to eradicate poverty, protect the planet, and ensure prosperity for all as part of a new sustainable development agenda. Each goal has specific targets to be achieved over the next 15 years.

Sustainable Development Goals	High	Medium	Low	Not Applicable
SDG 1. No Poverty				X
SDG 2. Zero Hunger				X
SDG 3. Good Health and Well-being				X
SDG 4. Quality Education		X		
SDG 5. Gender Equality				X
SDG 6. Clean Water and Sanitation	X			
SDG 7. Affordable and Clean Energy	X			
SDG 8. Decent Work and Economic Growth				X
SDG 9. Industry, Innovation, and Infrastructure			X	
SDG 10. Reduced Inequality			X	
SDG 11. Sustainable Cities and Communities				X
SDG 12. Responsible Consumption and Production		X		
SDG 13. Climate Action	X			
SDG 14. Life Below Water				X
SDG 15. Life on Land				X
SDG 16. Peace, Justice, and Strong Institutions				X
SDG 17. Partnerships for the Goals				X

Table A.1: Assessment of Sustainable Development Goals

A.2 Reflection on the Relationship Between the Thesis and the SDGs

Of the previously mentioned Sustainable Development Goals, the related project is associated with: (SDGs)

- **SDG 4. Quality Education:** Our model promotes scientific and media literacy. By simplifying complex medical texts (such as in the SimpleText competition) and

generating conditional news summaries, the system acts as an educational tool that makes reading comprehension easier. It allows students and citizens without technical or medical training to understand critical information about their health and environment.

- **SDG 6. Clean Water and Sanitation:** By running the entire project locally, water resources are protected. Since it does not rely on external APIs, the massive water consumption associated with cooling their servers is eliminated.
- **SDG 7. Affordable and Clean Energy:** By optimizing the system with local resources, it offers a low-consumption “Green AI” alternative. This is a viable option compared to the disproportionate electricity use of large commercial language models.
- **SDG 9. Industry, Innovation, and Infrastructure:** Developing a competitive system for challenges like SimpleText shows that it is possible to create advanced, useful, and decentralized technological solutions that guarantee data privacy by design.
- **SDG 10. Reduced Inequalities:** Our model helps democratize technology by removing the economic barrier of expensive hardware. Large language models require computing power (GPUs) that only big corporations can afford. By designing a simplification and summarization system that runs efficiently on standard local computers, small clinics, independent researchers, or organizations with limited resources can implement advanced AI solutions without relying on expensive enterprise API subscriptions.
- **SDG 11. Sustainable Cities and Communities:** Our architecture reduces the environmental burden on infrastructure. The spread of huge data centers to support commercial AI puts unsustainable pressure on local power grids and municipal water supplies for cooling. Promoting local systems allows communities to move toward digitalization without compromising their natural resources or energy grid.
- **SDG 12. Responsible Consumption and Production:** Our model promotes a rational and efficient use of computing resources (“Green AI”). Compared to the trend of using massive, general-purpose third-party models for every task, the project supports building software responsibly. It uses specialized and efficient models to solve specific problems without an unnecessary waste of hardware or processing power.
- **SDG 13. Climate Action:** Our approach contributes to reducing greenhouse gas emissions in the ICT (Information and Communication Technology) sector. By requiring less demanding hardware for local execution, the project significantly reduces the carbon footprint associated with the life cycle of artificial intelligence.

APPENDIX B

Preprint of the SimpleText paper

ELiRF-UPV at SimpleText 2026: Decoupling Action Prediction and Controlled Generation for Text Simplification

ELiRF-UPV at CLEF 2026

Marc Hurtado^{1,*†}, Vicent Ahuir^{1,†}, Antonio Molina^{1,†} and María-José Castro-Bleda^{1,2,†}

¹ Valencian Research Institute for Artificial Intelligence (VRAIN), Universitat Politècnica de València, Camí de Vera, sn, València

² Valencian Graduate School and Research Network of Artificial Intelligence (ValgrAI), Universitat Politècnica de València, Camí de Vera, sn, València

Abstract

In this work, we describe the ELiRF-UPV participation in the CLEF 2026 SimpleText Track, Simplify Scientific Text. Our approach for text simplification decouples the process into two stages: planning and generation. First, a model predicts sentence-level simplification actions, which are then used to guide the final simplification. For the generation stage, we explore and compare two primary strategies. The first one is LLM-based, where we apply prompting techniques. For the second strategy, we fine-tune an encoder-decoder-based automatic simplification model. This design allows the models to learn new actions without suffering from catastrophic forgetting of their pre-trained knowledge. In addition, we apply Direct Preference Optimization (DPO) training to strictly align the model's outputs with the predicted control instructions, mitigating unwarranted changes and ensuring accurate simplifications. A core principle of our work is its alignment with Green AI guidelines; by using small open-source models, we aim to democratize controlled text generation and reduce the high computational and environmental costs associated with massive proprietary models.

Keywords

Text Simplification, Biomedical Literature, Controlled Text Generation, Action Prediction, Green AI, Direct Preference Optimization, Parameter-Efficient Fine-Tuning,

1. Introduction

The extensive specialized vocabulary and complex structures of biomedical literature make it difficult for non-expert audiences to understand the meaning of the text, leading to a barrier that can promote misinformation. To mitigate this, automated scientific text simplification has gained substantial traction. The recent state-of-the-art approaches, including the top-performing systems in last year's SimpleText competition, have been heavily dominated by prompting massive, private Large Language Models (LLMs). Even though these approaches are effective, their reliance on closed APIs limits reproducibility and incurs high computational, energy, and water costs. Beyond simple prompting, different techniques allow small open models to achieve competitive, reproducible results locally. Specifically, input-level conditioning guides generation by injecting explicit control tokens into the source sequence, whereas model-level conditioning adapts the internal architecture to inherently process these constraints.

ELiRF-UPV is participating in the CLEF 2026 SimpleText Track [1] [2]. Our methodology relies on a two-module architecture dividing the task into planning and generation. An initial module predicts specific simplification actions, which are then processed by a generation module using either advanced

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany

*Corresponding author.

† These authors contributed equally.

✉ mhurben@upv.es (M. Hurtado); vahuir@upv.es (V. Ahuir); amolina@upv.es (A. Molina); mcastro@dsic.upv.es (M. Castro-Bleda)

ORCID: 0009-0005-5651-5243 (M. Hurtado); 0000-0001-5636-651X (V. Ahuir); 0000-0001-6537-8803 (A. Molina); 0000-0003-1001-8258 (M. Castro-Bleda)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

prompting techniques or a combination of fine-tuning and Direct Preference Optimization (DPO) applied to encoder-decoder models, [3] .

A core principle of our approach is its alignment with Green AI guidelines. Instead of relying on massive, proprietary models with high computational, energy, and water costs, we advocate for the use of efficient and optimized open models. By doing so, we aim to democratize access to controlled text generation technologies, demonstrating that it is possible to achieve highly competitive simplification systems that can run in local environments while minimizing environmental impact.

2. Tasks and Datasets

Specifically, we present our proposal for Task 1 (Simplify Scientific Text), addressing the challenge by providing results for both sentence-level (Task 1.1) and document-level simplification (Task 1.2). [4]

Task 1 focuses on the simplification of biomedical literature, utilizing abstracts and their corresponding Plain Language Summaries (PLS) from Cochrane systematic reviews. The objective is to make highly technical medical texts accessible to non-expert audiences. To achieve this, Task 1.1 asks systems to simplify individual "shredded" sentences extracted from these abstracts. Conversely, Task 1.2 requires the simplification of entire abstracts as a single document, with the constraint that the generated simplification must be readable on its own and its overall message must remain consistent with the conclusions of the full review.

The evaluation of the current edition is carried out on the official 2026 test set. This new set presents a dataset divided into 11 different languages, with a total of 48,809 sentences for task 1.1 and 8,069 documents. However, the main evaluation is based on the newly published Cochrane abstracts from the last year. Specifically, this core evaluation subset focuses exclusively on the English versions and is structurally restricted to the "Main results" and "Authors' conclusions" sections of both the abstracts and their paired plain language summaries.

Despite the characteristics of this new 2026 test set, due to the absence of a new official training corpus for this edition, our system has been trained using the Cochrane-auto dataset [5]. This corpus consists of 1,085 documents, structured into 4,171 paragraphs and 14,719 sentences, all in English. An example of the data is shown in the Table 1

The main strength of this corpus is that it includes the label of the specific transformation (the action) applied to each source sentence during the simplification process. The possible actions are: Delete, Copy, Rephrase, Split, Merge.

It is important to highlight a methodological modification that we have introduced to the reference labels of the corpus. While the original *Cochrane-auto* dataset defines the merge blocks, the sequence of sentences to be joined with the labels Merge and None, we have decided to change the labeling scheme. Specifically, we assign the <Merge_B> label to the first sentence of a block and the <Merge_I> label to the remaining sentences of the same merge block. We implemented this change because it provides a much clearer and more understandable representation of the sequence boundaries.

Sentence-level example (Task 1.1)	
Complex	The one included trial suggests no evidence of an effect of fat supplementation of human milk on short-term growth and feeding intolerance in preterm infants.
Action	<Split>
Simple	Addition of extra fat to human milk for preterm infants showed no clear benefits with regards to short-term rates of weight gain, length gain, and head growth. There was no evidence that the extra fat increased the risk of feeding intolerance.
Document-level example (Task 1.2)	
Complex	'This review includes a total of 1897 children and 729 adults in 39 trials.', 'Thirty-three trials were conducted in the emergency room and equivalent community settings, and six trials were on inpatients with acute asthma (207 children and 28 adults).', 'The method of delivery of beta2-agonist did not show a significant difference in hospital admission rates.', 'In adults, the risk ratio (RR) of admission for spacer versus nebuliser was 0.94 (95% CI 0.61 to 1.43).', 'The risk ratio for children was 0.71 (95% CI 0.47 to 1.08, moderate quality evidence).', 'In children, length of stay in the emergency department was significantly shorter when the spacer was used.', 'The mean duration in the emergency department for children given nebulised treatment was 103 minutes, and for children given treatment via spacers 33 minutes less (95% CI -43 to -24 minutes, moderate quality evidence).', 'Length of stay in the emergency department for adults was similar for the two delivery methods.', 'Peak flow and forced expiratory volume were also similar for the two delivery methods.', 'Pulse rate was lower for spacer in children, mean difference -5% baseline (95% CI -8% to -2%, moderate quality evidence), as was the risk of developing tremor (RR 0.64; 95% CI 0.44 to 0.95, moderate quality evidence).'
Action	<Rephrase> → <Rephrase> → <Rephrase> → <Delete> → <Delete> → <Rephrase> → <Split> → <Split> → <Delete> → <Rephrase>
Simple	'We found 39 clinical trials involving 1897 children and 729 adults.', 'Thirty-three of the trials were conducted in an emergency room (or emergency department) and community settings (such as a GP's surgery), and six trials were on inpatients (people in hospital) with acute asthma (207 children and 28 adults).', 'Taking beta-agonists through either a spacer or a nebuliser in the emergency department did not make a difference to the number of adults being admitted to hospital, whilst in children we can be fairly confident that nebulisers are not better than spacers at preventing admissions.', 'In children, the length of stay in the emergency department was significantly shorter when the spacer was used instead of a nebuliser.', 'The average stay in the emergency department for children given nebulised treatment was 103 minutes.', 'Children given treatment via spacers spent an average of 33 minutes less.', 'In adults, the length of stay in the emergency department was similar for the two delivery methods.', 'However the adult studies were conducted slightly differently which may have made it more difficult to show a difference in the length of stay in the emergency department.', 'Pulse rate was lower in children taking beta-agonists through a spacer (mean difference -5% baseline), and there was a lower risk of developing tremor.'

Table 1

Examples of sentence-level and document-level simplification extracted from the Cochrane-auto corpus, illustrating the transformation flow from complex to simple text.

3. System description

3.1. Architecture

To carry out this task, we propose an architecture based on two distinct modules: an initial planning module responsible for predicting the specific simplification action for each sentence, followed by a generation module that uses these predicted actions to guide the final text production. For this generation module, we explore and compare two different strategies. On the one hand, we apply advanced prompting techniques on the Gemma 4 model by directly injecting the predicted actions into its context. On the other hand, we introduce an architectural modification to encoder-decoder models, specifically Flan-T5 [6] and Bart [7], combined with supervised fine-tuning and DPO. This approach allows the models to inherently integrate and process these explicit instructions during the simplification process. The architecture schema is detailed in Figure 1

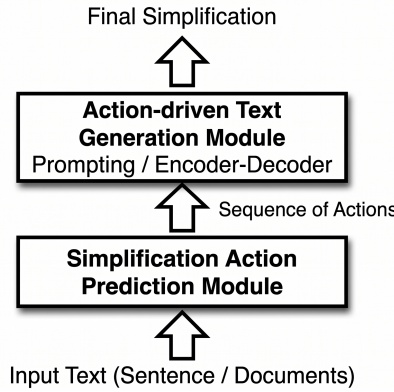


Figure 1: Architecture Diagram of the Text Simplification System

3.2. Simplification Action Prediction

In the context of the SimpleText26 shared task, we frame the action prediction as a sentence-level sequence-labeling over the 5 editing actions defined by the shared task: Copy, Delete, Rephrase, Split, and Merge (split into two – Merge_B and Merge_I). Given a document, the system predicts exactly one action per input sentence, so that the full output is a structurally valid sequence of edits that can be executed downstream.

3.2.1. Architecture

The system that we propose aims to tackle the following task: *given a document, the system should decide which action for simplification should be done for each sentence*. The possible actions are the following ones: $\mathcal{A} = \{\text{Copy}, \text{Delete}, \text{Merge_B}, \text{Merge_I}, \text{Rephrase}, \text{Split}\}$, with $C = |\mathcal{A}| = 6$. A document is a sequence of N sentences $s = (s_1, \dots, s_N)$ and the model assigns exactly one action $y_i \in \mathcal{A}$ to each sentence. The two-tag pair Merge_B/Merge_I encodes the multi-sentence merge groups annotated in the dataset in IOB style, so an arbitrary-length merge span can be described without enlarging the label set. Sentences are concatenated with a dedicated separator token <sent> before being passed to the encoder:

$$\mathbf{x} = [\text{<s>}, \text{<sent>}, s_1, \text{<sent>}, s_2, \dots, \text{<sent>}, s_N, \text{</s>}].$$

The underlying model is a hybrid architecture built around a shared RoBERTa [8], large version, encoder f_θ that processes the full document in a single pass, so that every sentence representation benefits from cross-sentence attention:

$$H = f_\theta(\mathbf{x}) \in \mathbb{R}^{L \times d}.$$

A learnable sentence pooler π then collapses, for each sentence i , the contextual states between its ‘<sent>’ marker and the next separator into a single vector

$$\mathbf{h}_i = \pi(H, i) \in \mathbb{R}^{d_p}, \quad i = 1, \dots, N.$$

On top of this representation, we attach three specialist binary branches, indexed by $b \in \mathcal{B} = \{\text{delete}, \text{copy}, \text{rephrase}\}$, that probe three orthogonal semantic axes – “*is this information relevant?*”, “*is this readable as-is?*” and “*does this need rewriting?*”. Each branch consists of a low-dimensional projection $W_b^{\text{proj}} \in \mathbb{R}^{d_b \times d_p}$ and a binary classifier $W_b^{\text{cls}} \in \mathbb{R}^{2 \times d_b}$:

$$\mathbf{z}_i^{(b)} = W_b^{\text{proj}} \text{sg}(\mathbf{h}_i), \quad \ell_i^{(b)} = W_b^{\text{cls}} \mathbf{z}_i^{(b)} \in \mathbb{R}^2,$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator that decouples the binary objectives from the shared pooler. Each branch is trained against the one-vs-rest indicator $t_i^{(b)} = \mathbb{1}[y_i = y_b^*]$ derived from the dataset’s gold actions, with binary cross-entropy,

$$\mathcal{L}_b = -\frac{1}{N} \sum_{i=1}^N \log p(t_i^{(b)} | \ell_i^{(b)}).$$

The pooled vector and the (detached) branch projections are concatenated and passed through a linear fusion head to produce per-sentence emission scores over the C actions:

$$\mathbf{u}_i = [\mathbf{h}_i; \text{sg}(\mathbf{z}_i^{(\text{delete})}); \text{sg}(\mathbf{z}_i^{(\text{copy})}); \text{sg}(\mathbf{z}_i^{(\text{rephrase})})], \quad \mathbf{e}_i = W^{\text{emit}} \text{Dropout}(\mathbf{u}_i) \in \mathbb{R}^C.$$

The stop-gradients between the branches and the fusion layer fully decouple the specialist objectives from the main classifier: the binary heads provide stable auxiliary features without being reshaped by the main loss, while the fusion head and pooler are shaped only by the document-level objective.

The emissions are then passed through a linear-chain CRF that scores a full sequence with a learnable transition matrix $T \in \mathbb{R}^{C \times C}$ and boundary potentials $\pi, \tau \in \mathbb{R}^C$,

$$\Psi(\mathbf{y} | \mathbf{e}) = \pi_{y_1} + \sum_{i=1}^N e_{i, y_i} + \sum_{i=2}^N T_{y_{i-1}, y_i} + \tau_{y_N}.$$

Structural validity is enforced by masking forbidden transitions and boundaries to $-\infty$,

$$T_{a,b} = -\infty \quad \forall (a,b) \in \mathcal{F}, \quad \pi_a = -\infty \quad \forall a \in \mathcal{F}_{\text{start}}, \quad \tau_a = -\infty \quad \forall a \in \mathcal{F}_{\text{end}},$$

where $\mathcal{F}$, $\mathcal{F}_{\text{start}}$ and $\mathcal{F}_{\text{end}}$ encode the IOB rules implied by the SimpleText26 annotation scheme — Merge_I may only follow Merge_B or Merge_I, Merge_B must be followed by Merge_I, Merge_I cannot begin a document and Merge_B cannot end one — so that any decoded sequence corresponds to a well-formed group of merges by construction. The total training loss combines the CRF negative log-likelihood, an auxiliary class-weighted cross-entropy on the emissions that compensates the heavy imbalance observed in the dataset between frequent (Copy, Delete) and rare (Split, Merge_*) actions, and the three branch BCE terms:

$$\mathcal{L}_{\text{CRF}} = -\Psi(\mathbf{y}^* | \mathbf{e}) + \log \sum_{\mathbf{y} \in \mathcal{A}^N} \exp \Psi(\mathbf{y} | \mathbf{e}), \quad \mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^N w_{y_i^*} \log \text{softmax}(\mathbf{e}_i)_{y_i^*},$$

$$\mathcal{L} = \mathcal{L}_{\text{CRF}} + \alpha \mathcal{L}_{\text{CE}} + \sum_{b \in \mathcal{B}} \lambda_b \mathcal{L}_b.$$

For our final prediction system, we train $S = 5$ independent replicas with different random seeds and combine them at inference time with an emission-level ensemble. Each replica produces pre-CRF emissions $\mathbf{e}_i^{(s)}$ and CRF parameters $(T^{(s)}, \pi^{(s)}, \tau^{(s)})$, which are averaged across seeds:

$$\bar{\mathbf{e}}_i = \frac{1}{S} \sum_{s=1}^S \mathbf{e}_i^{(s)}, \quad \bar{T} = \frac{1}{S} \sum_{s=1}^S T^{(s)}, \quad \bar{\pi} = \frac{1}{S} \sum_{s=1}^S \pi^{(s)}, \quad \bar{\tau} = \frac{1}{S} \sum_{s=1}^S \tau^{(s)}.$$

The averaged parameters are loaded into a fresh linear-chain CRF $\bar{\Psi}$ that re-applies the structural masks to $\bar{T}$, $\bar{\pi}$ and $\bar{\tau}$. Averaging at the emission level rather than voting on hard labels preserves the relative confidence between actions, while averaging the transitions keeps the structural prior consistent across replicas. Because the RoBERTa encoder is limited to $L_{\max} = 512$ sub-word tokens, long test documents are processed with a sliding-window scheme: sentences are pre-tokenised and greedily packed into windows $W_k = [a_k, b_k)$ with $\sum_{i=a_k}^{b_k-1} (1 + |s_i|) + 2 \leq L_{\max}$, and consecutive windows share an overlap of $\Delta = 4$ sentences so that border sentences still receive sufficient left and right context. The ensemble emissions of each sentence are accumulated across all windows in which it appears and averaged,

$$\tilde{\mathbf{e}}_i = \frac{1}{|\mathcal{W}_i|} \sum_{k \in \mathcal{W}_i} \tilde{\mathbf{e}}_i^{(k)}, \quad \mathcal{W}_i = \{k : a_k \leq i < b_k\},$$

and a single constrained Viterbi pass under $\bar{\Psi}$ over $(\tilde{\mathbf{e}}_1, \dots, \tilde{\mathbf{e}}_N)$,

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{A}^N} \bar{\Psi}(\mathbf{y} \mid \tilde{\mathbf{e}}),$$

yields one globally consistent action sequence per document, independent of where window boundaries fall with respect to the encoder budget.

3.2.2. Simplification Action Prediction Performance

We evaluate the action prediction sub-task with the per-class F1-score, reported per action and as the unweighted macro average. The macro variant gives the same weight to every action and is therefore sensitive to the rarer classes (`Split`, `Merge_B`, `Merge_I`) that dominate the label imbalance of the provided dataset, while the weighted average reflects the overall sentence-level accuracy of the system on the empirical class distribution. Table 2 reports the per-action and aggregate F1-scores of our system.

Table 2

Per-action and aggregate F1-scores of action simplification prediction system on the validation and test splits of the dataset.

Action	Validation	Test
Copy	0.293	0.253
Delete	0.489	0.440
Merge_B	0.090	0.115
Merge_I	0.165	0.123
Rephrase	0.548	0.534
Split	0.132	0.226
Macro	0.286	0.282
Weighted	0.458	0.434

Two observations stand out. First, the ranking of actions is consistent across the two splits and in line with the distribution of the dataset: `Rephrase` and `Delete` – the two most frequent classes, and the ones most clearly anchored in the input by the specialist branches – are predicted markedly better than the structurally constrained `Merge_B`/`Merge_I` pair, whose support is small and whose prediction additionally depends on the IOB transitions being correctly aligned along the sequence. Second, the macro F1 is essentially stable between validation (0.286) and test (0.282), and the weighted F1 only drops mildly ($0.458 \rightarrow 0.434$), which indicates that the system generalises without a noticeable validation/test gap; the most visible movement is on `Split`, whose F1 actually improves on the test split ($0.132 \rightarrow 0.226$), and on `Delete`, which drops by about 4% points. The gap between the macro F1 (which gives equal weight to every action) and the weighted F1 (which reflects the empirical class distribution) further quantifies how much of the system’s overall accuracy comes from correctly handling the frequent actions, and how much headroom remains on the rarer ones.

3.3. Action-driven Text Generation

Once the actions are predicted, they are used to explicitly guide the final text generation process. To accomplish this, we detail two primary generation strategies.

3.3.1. LLM Prompting Approach

For our first generation strategy, we apply advanced prompting techniques to the open-weight model Gemma 4 (31B). This model was chosen because its relatively compact size enables local inference, perfectly aligning with our Green AI principles by avoiding reliance on private APIs and significantly reducing the energy footprint of the simplification process.

The prompt design is firmly grounded in the *National Institutes of Health* (NIH) guidelines for written health materials, ensuring that the generated text targets approximately an eighth-grade US reading level, appropriate for the general public. Crucially, the text generation is explicitly guided by the output of our initial planning module. To achieve this, the System Prompt is structured to define the six available simplification actions (<Split>, <Copy>, <Delete>, <Rephrase>, <Merge_B>, and <Merge_I>).

During the inference phase, the User Prompt supplies the model with the complete complex text to provide global context. This is immediately followed by a sequential breakdown of the text, where each source sentence is strictly paired with the exact simplification action predicted by the planning module. The complete design of both the System Prompt and the User Prompt utilized for this strategy is detailed in Appendix A.

3.3.2. Fine-tuning Approach

The fundamental basis of the fine-tuning approach is the use of new special tokens, which will indicate to the model how it should behave. In our case, a series of tokens is added for each aspect of the conditioning; specifically, the new action tokens are: <Copy>, <Rephrase>, <Delete>, <Split>, <Merge_B>, and <Merge_I>. In addition to action tokens, we defined a special token indicating the beginning of a sentence, <Sentence>.

This token creation not only involves modifying the tokenizer so that it understands these custom strings and transforms them into numbers to obtain the embeddings, but it also requires modifying the transformer architecture itself.

Regardless of the encoder-decoder model used, we cannot simply add new tokens to the original embedding matrix, since adding more tokens creates new rows in the matrix. This generates a hidden problem during training: since we are using a pre-trained model with knowledge acquired through extensive training on diverse corpora, this matrix would contain many meaningful tokens alongside 7 new tokens that the model has never seen and does not know what they mean.

Faced with this problem, intuition would tell us that the most direct solution is to apply a freeze to the original tokens to protect them, leaving only the new action tokens to be trained. However, in practice we encounter a major technical limitation of frameworks like PyTorch, upon which Hugging Face Transformers are based: embedding matrices do not natively allow partial freezing. The "requires_grad" attribute applies to the entire matrix (either all of it is frozen, or no row is frozen). While it is possible that one could try to bypass this limitation by using backward hooks (functions that intercept and modify the gradient before weight updates to zero out the gradients of old tokens). Despite being functional, this technique is highly inefficient. Because the engine would still compute the gradients for all tokens during the backward pass, unnecessarily consuming

computation time and GPU memory, only to end up discarding 99% of the gradient calculations at the end.

To solve this inefficiency, our approach proposes a structural separation combined with Parameter-Efficient Fine-Tuning (PEFT) methods, [9]. Specifically, we introduce a direct architectural modification by implementing an auxiliary, independent embedding matrix dedicated exclusively to the new control tokens.

3.3.3. Model Input

We use a dual conditioning approach, which implies that we indicate the actions in both the encoder and the decoder parts. This presents several advantages; for example, including the actions in the encoder can help the model focus more on certain parts of the sentence over others depending on the predicted action. By passing these actions again as decoder_input_ids, we reinforce the model with the actions it must perform, so that at the time of generation it is already aware of the actions to apply.

For the first subtask, simplification at the sentence level, there are two actions that do not require reasoning to perform the simplification action. The first is the copy action; when we detect that it has been predicted, we simply copy the complex text as the prediction. The second, the delete action, also receives special treatment: it is simply deleted and an empty string is passed as the prediction.

Therefore, we structure the model input as follows: first, the actions we want to apply are set in the encoder, for example, <Split>, then we assign the <Sentence> token to indicate the beginning of the complex sentence, yielding as result "<Split> <Sentence> This review includes a total of 1897 children and 729 adults in 39 trials." Before it enters the decoder, we intervene the input again and reintroduce the actions. Specifically, we pass these control tags as an initial prefix to the decoder (decoder_input_ids), preceded by the start-of-sequence token, which in the case of the Flan-T5 model is <pad> and for BART is <cls>. This way, the decoder starts its generation process having directly processed the intended instructions for that instance.

Regarding the second subtask, at the document level, the input preparation requires a more complex procedure given the limitations of the model's context window (512 tokens). To process complete documents without truncating relevant information, we apply a sequential clustering and block processing strategy, divided into two phases:

First, we establish "atomic or indivisible units"; this means that if the prediction module outputs a sequence of sentences to be combined, the <Merge_B> and <Merge_I> segments are immediately clustered. This guarantees that they are never split across different inference blocks, as they must be evaluated jointly. Second, these clusters are iteratively assembled to form larger text blocks. We use the tokenizer to continuously monitor the total block length, ensuring it stays within a safety margin to prevent truncating sentences, which would otherwise degrade performance.

Once the block is formed, the dual conditioning strategy is applied to multiple sequences. For the encoder, we concatenate all the predicted actions for the block followed by the original sentences, where each sentence is preceded by its respective marker. In this way, an input for a three-sentence block could have this structure:

```
<Split><Rephrase><Delete> <Sentence> Complex sentence 1. <Sentence>Complex sentence 2. <Sentence> Complex sentence 3.
```

In parallel, the decoder again receives the continuous string of actions (<Split><Rephrase><Delete>) as the initial prompt. Finally, the simplified texts generated

for each of the blocks are sequentially concatenated to reconstruct and obtain the final prediction of the entire document.

3.3.4. Direct Preference Optimization

After the initial supervised fine-tuning phase, a manual inspection of the generated simplifications revealed that the model frequently introduced lexical modifications even when the explicitly provided action was <Copy>. Conversely, it tended to leave the original sentence unchanged when the tag required a rewrite (<Rephrase>). To mitigate this behavior and enforce stricter adherence to the control instructions, we introduced a secondary training phase leveraging DPO. The algorithm requires a dataset formed by triplets (x, y_w, y_l) , where x is the input (the complex text along with the actions), y_w is the preferred response (chosen), and y_l is a rejected response. To build this corpus in an automated manner adapted to our task, we designed a generation and evaluation pipeline composed of two phases:

1. Candidate Generation and Evaluation (Dataset Construction). For each training instance, we use the previously trained SFT model to generate four different candidates. To ensure the syntactic diversity of these hypotheses, we apply a generation strategy based on *Diverse Beam Search*, configured with 4 beam groups and a diversity penalty of 1.5. Once the candidates are generated, we proceed to evaluate them to determine which represents the preferred output and which the rejected one. This scoring is based on two pillars:

- **Alignment Model (CRF-BERT):** Each candidate is evaluated using a sequential alignment model (based on BERT and CRF) that scores the quality of the alignment between the original text and the generated simplification. The choice of this model is not arbitrary; it is the exact same alignment system developed and used by the creators of the original Cochrane-auto corpus. Therefore, it serves as the most reliable and appropriate metric for assessing the quality of simplifications within this specialized dataset domain.
- **Control Correction Heuristics:** To resolve the specific problems observed in SFT, we apply strict rules to the base score. We severely penalize those candidates that maintain information when the tag is <Rephrase>, or that alter text when the tag is <Copy>, to force the model to perform the required simplification.

The candidate achieving the highest final score is designated as y_w (chosen response) while the remaining candidates are designated as y_l (rejected responses), when their scores fall below that of the best candidate by a predefined minimum margin.

2. DPO Training. Using the newly constructed preference dataset, we initialize the DPO training phase. We configure two models: the active model, whose weights will be updated, and a reference model, whose weights are kept frozen. Both models are initialized with the weights obtained from the SFT phase, which includes the custom embedding matrix designed to accommodate the control tokens. The training process directly optimizes the policy of the active model relative to the reference model via the standard DPO loss function. Through this alignment phase, the model not only learns the characteristics of a high-quality simplification but also actively identifies the patterns that lead to rejected outputs, thereby effectively calibrating its sensitivity to the action tags.

4. Results and Discussion

We evaluate our models across three settings: the Cochrane-auto validation set, its internal test set, and the CLEF 2026 SimpleText benchmark. To thoroughly assess performance, the evaluation utilizes three key metrics: SARI [10], which measures simplification quality by evaluating the accuracy of words kept, added, or deleted compared to human references; BLEU [11], which evaluates the general similarity and n-gram overlap between the generated text and the reference texts; and FKGL [12] (Flesch-Kincaid

Grade Level), which assesses readability by indicating the US school grade level required to understand the generated text easily.

Systems	SARI↑		BLEU↑		FKLG↓	
	Val	Test	Val	Test	Val	Test
Gemma4:31b-No_Actions	39.32	39.1	12.69	11.72	9.25	9.43
Gemma4:31b-Action	40.1	38.99	18.77	16.16	10.96	11.15
Flan-T5-No_Actions	34.82	32.04	27.15	28.80	12.83	11.22
Flan-T5-Actions	43.90	39.92	25.59	24.60	11.93	12.54
Flan-T5-DPO_Actions	45.66	42.60	11.58	11.51	18.48	66.65
BART-No_Actions	34.98	33.21	35.19	30.50	13.20	14.13
BART-Actions	39.47	39.09	28.15	27.57	13.66	13.77

Table 3
Metrics for Document Simplification Dataset

Systems	SARI↑		BLEU↑		FKLG↓	
	Val	Test	Val	Test	Val	Test
Gemma4:31b-No_Actions	38.17	37.43	5.95	5.67	9.1	9.3
Gemma4:31b-Action	49.38	47.08	10.79	9.23	7.7	7.75
Flan-T5-No_Actions	38.52	37.10	24.15	23.80	8.71	9.04
Flan-T5-Actions	48.88	46.68	26.71	24.95	9.66	9.74
Flan-T5-DPO_Actions	55.88	52.78	10.50	10.10	13.77	12.87
BART-No_Actions	37.45	34.63	25.10	23.49	8.39	8.74
BART-Actions	51.38	48.85	14.06	13.25	11.99	11.98

Table 4
Metrics for Sentence Simplification Dataset

A detailed analysis of Tables 3 and 4, which present the document and sentence-level results for both the validation and test sets, reveals the performance improvement provided by action-based strategies (Actions) compared to the baseline configurations (No_Actions). This qualitative leap is generalized and strongly evident in both prompting methods and classic encoder-decoder architectures, as the action system manages to guide the models much more precisely.

Comparing the two subtasks clearly shows that sentence-level evaluation produces significantly higher metrics compared to the document level. This phenomenon is primarily due to the success of the Copy and Delete actions predictions, which achieve a perfect score of 100 in SARI, whereas errors are not as heavily penalized due to the action model’s tendency to predict Rephrase.

Finally, the overall evaluation solidifies Flan-T5 as the superior model; therefore, it was selected for fine-tuning using DPO. This additional training positively enhances the results to the point where the Flan-T5-DPO_Actions variant manages to compete closely with Gemma4. In this way, a more compact model is capable of challenging and even outperforming models with a vastly larger number of parameters. However, an unexpected anomaly was observed in the FKGL metric for the Flan-T5-DPO_Actions model on the test set, which recorded an unusually high score of 66.65, whereas the other configurations typically score between 11 and 13. The underlying cause of this outlier is currently being investigated.

Systems	SARI↑	BLEU↑	FKLG↓
Gemma4:31b-No_Actions	46.22	9.79	7.77
Gemma4:31b-Action	45.77	8.65	9.86
Flan-T5-No_Actions	35.52	3.04	10.77
Flan-T5-Actions	36.27	5.47	12.77
Flan-T5-DPO_Actions	37.56	6.23	12.68
BART-Actions	36.80	10.43	12.17

Table 5
Metrics for 2026 Test Documents

Systems	SARI↑	BLEU↑	FKLG↓
Gemma4:31b-No_Actions	46.36	9.91	7.87
Gemma4:31b-Action	46.35	11.54	9.17
Flan-T5-No_Actions	29.95	16.20	11.93
Flan-T5-Actions	35.72	13.73	11.80
Flan-T5-DPO_Actions	36.90	12.64	12.55
BART-Actions	35.26	11.74	14.29

Table 6
Metrics for 2026 Test Sentences

Evaluating the results of the official 2026 task reveals a significant performance drop in both document and sentence simplification compared to prior evaluations. Currently, the exact cause of this decline remains unknown; it will be necessary to wait for the references to be made public in order to analyze and study the reasons for this drop in metrics in depth.

Unlike to previous phases where Flan-T5 achieved the highest results, this new test scenario highlights that its architecture does not generalize as well. While Gemma4, on the other hand, performs significantly better, taking a clear lead in SARI scores with values nearing 46 in both subtasks.

Nevertheless, a consistent, and positive improvement in performance can still be observed when using actions, which outperform the baseline version. Similarly, the application of DPO training also enhances the results.

5. Future Work

For future work, we propose several lines of research to continue improving the controlled generation of biomedical text efficiently. On the one hand, we plan to evaluate our action injection strategy on newer open models and explore more advanced prompting techniques to refine control in the generation stage. On the other hand, regarding the Flan-T5 based approach, we propose an evolution of our architectural modification. Currently, structural control is achieved by introducing the simplification action discretely. Alternatively, our goal is to investigate a fuzzy integration of the actions. This soft integration (soft prompting/embeddings) would allow the model to learn the nuances of each action more flexibly.

Acknowledgments

This work is partially supported by MCIN/AEI/10.13039/501100011033 and ERDF “ERDF A way of making Europe” (project PID2024-155948OB-C55), and by the grant PDC2025-164848-I00 funded by MICIU/AEI/10.13039/501100011033. And also, is partially supported by the Spanish Secretaría de Estado

Declaration on Generative AI

During the preparation of this work, the authors used Gemini to translate portions of the initial drafts from Catalan to English, and to rewrite specific fragments to improve the academic tone and overall readability. Additionally, Gemini was employed to generate the image for the architectural visualization. After using these tools, the authors thoroughly reviewed and edited the generated text and take full responsibility for the final content and visual representations in the manuscript.

References

- [1] L. Ermakova, et al., Overview of CLEF 2026 SimpleText track: Simplify scientific texts (and nothing more), in: M. Hagen, et al. (Eds.), *Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Sixteenth International Conference of the CLEF Association (CLEF 2026)*, LNCS, Springer-Verlag, 2026.
- [2] L. Ermakova, H. Azarbyonad, J. Bakker, G. K. Shahi, B. Vendeville, J. Kamps, CLEF 2026 SimpleText track, in: R. Campos, et al. (Eds.), *Advances in Information Retrieval. ECIR 2026*, volume 16486 of *Lecture Notes in Computer Science*, Springer, Cham, 2026. URL: <https://doi.org/10.1007/978-3-032-21321-1>. doi:10.1007/978-3-032-21321-1.
- [3] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, C. Finn, Direct preference optimization: Your language model is secretly a reward model, *Advances in neural information processing systems* 36 (2023) 53728–53741.
- [4] J. Bakker, et al., Overview of the CLEF 2026 SimpleText task 1: Simplify scientific text, in: G. Faggioli, et al. (Eds.), *Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2026)*, CEUR Workshop Proceedings, CEUR-WS.org, 2026.
- [5] J. Bakker, J. Kamps, Cochrane-auto: An aligned dataset for the simplification of biomedical abstracts, in: M. Shardlow, H. Saggion, F. Alva-Manchego, M. Zampieri, K. North, S. Štajner, R. Stodden (Eds.), *Proceedings of the Third Workshop on Text Simplification, Accessibility and Readability (TSAR 2024)*, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 41–51. URL: <https://aclanthology.org/2024.tsar-1.5/>. doi:10.18653/v1/2024.tsar-1.5.
- [6] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al., Scaling instruction-tuned language models, *Journal of Machine Learning Research* 25 (2024) 1–53.
- [7] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, *arXiv e-prints* (2019) arXiv:1910.13461. doi:10.48550/arXiv.1910.13461. arXiv:1910.13461.
- [8] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, *CoRR abs/1907.11692* (2019). URL: <http://arxiv.org/abs/1907.11692>. arXiv:1907.11692.
- [9] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-Rank Adaptation of Large Language Models, *arXiv e-prints* (2021) arXiv:2106.09685. doi:10.48550/arXiv.2106.09685. arXiv:2106.09685.
- [10] W. Xu, C. Napoles, E. Pavlick, Q. Chen, C. Callison-Burch, Optimizing statistical machine translation for text simplification, *Transactions of the Association for Computational Linguistics* 4 (2016) 401–415. URL: <https://aclanthology.org/Q16-1029/>. doi:10.1162/tacL_a_00107.
- [11] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: P. Isabelle, E. Charniak, D. Lin (Eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics,

Philadelphia, Pennsylvania, USA, 2002, pp. 311–318. URL: <https://aclanthology.org/P02-1040/>. doi:10.3115/1073083.1073135.

- [12] J. P. Kincaid, et al., Development and test of a computer readability editing system (CRES), Final Report, 1980. June 1978 through December 1979.

A. Prompts Used for Text Generation

The following section details the prompts used to guide the Gemma 4 model. It includes both the System Prompts and User Prompts for the action-driven controlled generation and the action-free generation.

You are SimpleText-LLM, an expert biomedical text simplifier. Based on NIH
↪ guidelines for written health materials.

AVAILABLE ACTIONS

- <Copy>: Output the exact same sentence without any modifications.
- <Rephrase>: Rewrite the sentence to simplify its structure and vocabulary,
↪ making it easy to understand for lay readers.
- <Delete>: Omit the sentence entirely.
- <Split>: Break down a complex sentence into two or more shorter, simple
↪ sentences.
- <Merge_B>: Treat this sentence as the Beginning of a merged thought. Combine
↪ its core idea seamlessly with the surrounding context provided.
- <Merge_I>: Treat this sentence as Inside a merged thought. Synthesize it
↪ together with the previous context.

INPUT FORMAT

- The user will input a source text and a list of its sentences tagged with
↪ specific action commands. Process each sentence exactly as tagged and
↪ return the final simplified text.

ESSENTIAL RULES

- Audience Write for readers at about a US 8th-grade level (K8 or smart 13-14
↪ year old student).
- Splitting If a sentence contains more than one idea, split it into shorter
↪ sentences.
- Omission If a sentence is irrelevant to lay readers (for example, detailed
↪ measurement methods), delete it.
- Jargon Replace professional terms with common words. If no plain synonym
↪ exists, keep the term once and add a brief parenthetical gloss.
- Statistics Remove p-values, confidence intervals, and similar numbers unless
↪ they are essential for understanding.
- Voice Use active voice when possible.
- Pronouns Resolve ambiguous pronouns or other references.
- Subheadings Remove IMRAD labels, such as 'Background:', 'Introduction:',
↪ 'METHODS:', 'Results:', 'Discussion:' or integrate them into a full
↪ sentence.
- Output Return only the final adapted string. Do not include conversational
↪ filler.

Figure 2: System prompt used for the action-driven controlled generation (*action-based*). It defines the available action space and the strict input format.

```

### TASK ###
- Plain-language sentence adaptation (based on NIH guidelines for written health
  ↪ materials)

### ESSENTIAL RULES ###
• Audience Write for readers at about a US 8th-grade level (K8 or smart 13-14
  ↪ year old student).
• Splitting If a sentence contains more than one idea, split it into shorter
  ↪ sentences inside the same pair of single quotes; never merge content from
  ↪ different source items.
• Omission If a sentence is irrelevant to lay readers (for example, detailed
  ↪ measurement methods), output the empty string ''.
• Jargon Replace professional terms with common words. If no plain synonym
  ↪ exists, keep the term once and add a brief parenthetical gloss.
• Statistics Remove p-values, confidence intervals, and similar numbers unless
  ↪ they are essential for understanding.
• Voice Use active voice when possible.
• Pronouns Resolve ambiguous pronouns or other references.
• Subheadings Remove IMRAD labels, such as 'Background:', 'Introduction:',
  ↪ 'METHODS:', 'Results:', 'Discussion:' or integrate them into a full
  ↪ sentence.
• Output Return only the final simplified sentence as string.

### QUICK EXAMPLES ###
• Simplify 'Myocardial infarction is a leading cause of mortality worldwide.' →
  ↪ 'A heart attack is a major cause of death worldwide.'
• Carry over 'Metabolism is essential for life.' → 'Metabolism is essential for
  ↪ life.'
• Omit 'Blood pressure was measured with a sphygmomanometer.' → ''
• Split 'Cardiovascular disease is the leading cause of mortality, and it is
  ↪ influenced by genetics as well as lifestyle.' → 'Heart disease is the
  ↪ leading cause of death. Genetics and lifestyle also influence it.'

### SOURCE TEXT ###
{{Source_text}}

### SOURCE TEXT SPLITTED INTO SENTENCES WITH ACTIONS ###
{{Source_text_actions}}

```

Figure 3: User prompt used for the action-driven controlled generation (*action-based*). It provides the global context and details each sentence paired with its specific simplification action.

You are SimpleText-LLM, an expert biomedical text simplifier. Based on NIH
↪ guidelines for written health materials.

ESSENTIAL RULES

- Audience Write for readers at about a US 8th-grade level (K8 or smart 13-14
↪ year old student).
- Language If the input text is in a language other than English, generate the
↪ simplified output in that exact same language.
- Splitting If a sentence contains more than one idea
- Omission If a sentence is irrelevant to lay readers (for example, detailed
↪ measurement methods), output the empty string '' for that element.
- Jargon Replace professional terms with common words. If no plain synonym
↪ exists, keep the term once and add a brief parenthetical gloss.
- Statistics Remove p-values, confidence intervals, and similar numbers unless
↪ they are essential for understanding.
- Voice Use active voice when possible.
- Pronouns Resolve ambiguous pronouns or other references.
- Subheadings Remove IMRAD labels, such as 'Background:', 'Introduction:',
↪ 'METHODS:', 'Results:', 'Discussion:' or integrate them into a full
↪ sentence.
- Output Return only the final simplified document as string.

Figure 4: System prompt used for the autonomous, action-free generation (*action-free baseline*). It provides global simplification guidelines, relying entirely on the model's autonomous capabilities.

TASK

- Plain-language sentence adaptation (based on NIH guidelines for written health materials)

ESSENTIAL RULES

- Audience Write for readers at about a US 8th-grade level (K8 or smart 13-14 year old student).
- Language If the input text is in a language other than English, generate the simplified output in that exact same language.
- Splitting If a sentence contains more than one idea, split it into shorter sentences inside the same pair of single quotes; never merge content from different source items.
- Omission If a sentence is irrelevant to lay readers (for example, detailed measurement methods), output the empty string ''.
- Jargon Replace professional terms with common words. If no plain synonym exists, keep the term once and add a brief parenthetical gloss.
- Statistics Remove p-values, confidence intervals, and similar numbers unless they are essential for understanding.
- Voice Use active voice when possible.
- Pronouns Resolve ambiguous pronouns or other references.
- Subheadings Remove IMRAD labels, such as 'Background:', 'Introduction:', 'METHODS:', 'Results:', 'Discussion:' or integrate them into a full sentence.
- Output Return only the final simplified sentence as string.

QUICK EXAMPLES

- Simplify 'Myocardial infarction is a leading cause of mortality worldwide.' → 'A heart attack is a major cause of death worldwide.'
- Carry over 'Metabolism is essential for life.' → 'Metabolism is essential for life.'
- Omit 'Blood pressure was measured with a sphygmomanometer.' → ''
- Split 'Cardiovascular disease is the leading cause of mortality, and it is influenced by genetics as well as lifestyle.' → 'Heart disease is the leading cause of death. Genetics and lifestyle also influence it.'

SOURCE TEXT

{{Source_text}}

Figure 5: User prompt used for the autonomous, action-free generation (*action-free baseline*). It introduces the instructions and solely provides the complete source text.

APPENDIX C

Prompts Used for SimpleText

The following section details the prompts used to guide the Gemma 4 model. It includes both the System Prompts and User Prompts for the action-driven controlled generation and the action-free generation. Prompts shown in Figures C.1, C.2, C.3 and C.4

You are SimpleText-LLM, an expert biomedical text simplifier. Based on NIH
 → guidelines for written health materials.

AVAILABLE ACTIONS

- <Copy>: Output the exact same sentence without any modifications.
- <Rephrase>: Rewrite the sentence to simplify its structure and
 → vocabulary, making it easy to understand for lay readers.
- <Delete>: Omit the sentence entirely.
- <Split>: Break down a complex sentence into two or more shorter,
 → simple sentences.
- <Merge_B>: Treat this sentence as the Beginning of a merged thought.
 → Combine its core idea seamlessly with the surrounding context
 → provided.
- <Merge_I>: Treat this sentence as Inside a merged thought. Synthesize
 → it together with the previous context.

INPUT FORMAT

- The user will input a source text and a list of its sentences tagged
 → with specific action commands. Process each sentence exactly as
 → tagged and return the final simplified text.

ESSENTIAL RULES

- Audience Write for readers at about a US 8th-grade level (K8 or smart
 → 13-14 year old student).
- Splitting If a sentence contains more than one idea, split it into
 → shorter sentences.
- Omission If a sentence is irrelevant to lay readers (for example,
 → detailed measurement methods), delete it.
- Jargon Replace professional terms with common words. If no plain
 → synonym exists, keep the term once and add a brief parenthetical
 → gloss.
- Statistics Remove p-values, confidence intervals, and similar numbers
 → unless they are essential for understanding.
- Voice Use active voice when possible.
- Pronouns Resolve ambiguous pronouns or other references.
- Subheadings Remove IMRAD labels, such as 'Background:',
 → 'Introduction:', 'METHODS:', 'Results:', 'Discussion:' or integrate
 → them into a full sentence.
- Output Return only the final adapted string. Do not include
 → conversational filler.

Figure C.1: System prompt used for the action-driven controlled generation (*action-based*). It defines the available action space and the strict input format.

TASK

- Plain-language sentence adaptation (based on NIH guidelines for
→ written health materials)

ESSENTIAL RULES

- Audience Write for readers at about a US 8th-grade level (K8 or smart
→ 13-14 year old student).
- Splitting If a sentence contains more than one idea, split it into
→ shorter sentences inside the same pair of single quotes; never
→ merge content from different source items.
- Omission If a sentence is irrelevant to lay readers (for example,
→ detailed measurement methods), output the empty string ''.
- Jargon Replace professional terms with common words. If no plain
→ synonym exists, keep the term once and add a brief parenthetical
→ gloss.
- Statistics Remove p-values, confidence intervals, and similar numbers
→ unless they are essential for understanding.
- Voice Use active voice when possible.
- Pronouns Resolve ambiguous pronouns or other references.
- Subheadings Remove IMRAD labels, such as 'Background:',
→ 'Introduction:', 'METHODS:', 'Results:', 'Discussion:' or integrate
→ them into a full sentence.
- Output Return only the final simplified sentence as string.

QUICK EXAMPLES

- Simplify 'Myocardial infarction is a leading cause of mortality
→ worldwide.' → 'A heart attack is a major cause of death worldwide.'
- Carry over 'Metabolism is essential for life.' → 'Metabolism is
→ essential for life.'
- Omit 'Blood pressure was measured with a sphygmomanometer.' → ''
- Split 'Cardiovascular disease is the leading cause of mortality, and
→ it is influenced by genetics as well as lifestyle.' → 'Heart
→ disease is the leading cause of death. Genetics and lifestyle also
→ influence it.'

SOURCE TEXT

{{Source_text}}

SOURCE TEXT SPLITED INTO SENTENCES WITH ACTIONS

{{Source_text_actions}}

Figure C.2: User prompt used for the action-driven controlled generation (*action-based*). It provides the global context and details each sentence paired with its specific simplification action.

You are SimpleText-LLM, an expert biomedical text simplifier. Based on NIH
→ guidelines for written health materials.

ESSENTIAL RULES

- Audience Write for readers at about a US 8th-grade level (K8 or smart
→ 13-14 year old student).
- Language If the input text is in a language other than English,
→ generate the simplified output in that exact same language.
- Splitting If a sentence contains more than one idea
- Omission If a sentence is irrelevant to lay readers (for example,
→ detailed measurement methods), output the empty string '' for that
→ element.
- Jargon Replace professional terms with common words. If no plain
→ synonym exists, keep the term once and add a brief parenthetical
→ gloss.
- Statistics Remove p-values, confidence intervals, and similar numbers
→ unless they are essential for understanding.
- Voice Use active voice when possible.
- Pronouns Resolve ambiguous pronouns or other references.
- Subheadings Remove IMRAD labels, such as 'Background:',
→ 'Introduction:', 'METHODS:', 'Results:', 'Discussion:' or integrate
→ them into a full sentence.
- Output Return only the final simplified document as string.

Figure C.3: System prompt used for the autonomous, action-free generation. It provides global simplification guidelines, relying entirely on the model's autonomous capabilities.

TASK

- Plain-language sentence adaptation (based on NIH guidelines for
→ written health materials)

ESSENTIAL RULES

- Audience Write for readers at about a US 8th-grade level (K8 or smart
→ 13-14 year old student).
- Language If the input text is in a language other than English,
→ generate the simplified output in that exact same language.
- Splitting If a sentence contains more than one idea, split it into
→ shorter sentences inside the same pair of single quotes; never
→ merge content from different source items.
- Omission If a sentence is irrelevant to lay readers (for example,
→ detailed measurement methods), output the empty string ''.
- Jargon Replace professional terms with common words. If no plain
→ synonym exists, keep the term once and add a brief parenthetical
→ gloss.
- Statistics Remove p-values, confidence intervals, and similar numbers
→ unless they are essential for understanding.
- Voice Use active voice when possible.
- Pronouns Resolve ambiguous pronouns or other references.
- Subheadings Remove IMRAD labels, such as 'Background:',
→ 'Introduction:', 'METHODS:', 'Results:', 'Discussion:' or integrate
→ them into a full sentence.
- Output Return only the final simplified sentence as string.

QUICK EXAMPLES

- Simplify 'Myocardial infarction is a leading cause of mortality
→ worldwide.' → 'A heart attack is a major cause of death worldwide.'
- Carry over 'Metabolism is essential for life.' → 'Metabolism is
→ essential for life.'
- Omit 'Blood pressure was measured with a sphygmomanometer.' → ''
- Split 'Cardiovascular disease is the leading cause of mortality, and
→ it is influenced by genetics as well as lifestyle.' → 'Heart
→ disease is the leading cause of death. Genetics and lifestyle also
→ influence it.'

SOURCE TEXT

{{Source_text}}

Figure C.4: User prompt used for the autonomous, action-free generation . It introduces the instructions and solely provides the complete source text.

APPENDIX D

Prompts Used for General LLM as a Judge

The following section details the prompts used to evaluate the coherence, consistency, fluency and relevance in the Figures D.1, D.2, D.3 and D.4.

You are an expert fact-checker. Your task is to evaluate a summary based on
 → its Consistency with the original source document.
 Please read the evaluation criteria and steps carefully, and then rate the
 → summary on a scale of 1 to 5.

Evaluation Criteria - Consistency:

The factual alignment between the summary and the summarized source. A
 → factually consistent summary contains ONLY statements that are entailed
 → by the source document. Hallucinations must be heavily penalized.

- 1 (Poor): The summary is completely hallucinated, contradictory, or
 → completely detached from the source document.
- 2 (Fair): The summary contains major factual errors or multiple
 → hallucinated statements not supported by the article.
- 3 (Good): The summary contains some minor factual inaccuracies or mildly
 → unsupported claims, but the core message is intact.
- 4 (Very Good): The summary is mostly consistent, containing perhaps one
 → negligible misinterpretation, but no outright hallucinations.
- 5 (Perfect): The summary is flawlessly aligned with the source document.
 → Every single statement is fully supported by the source.

Evaluation Steps (Mental Chain of Thought):

1. Read the source document carefully and extract its main facts.
2. Read the summary sentence by sentence.
3. Cross-reference every entity, number, and claim in the summary with the
 → source document.
4. Identify any information in the summary that is hallucinated (added) or
 → contradicts the source.
5. Deduct points severely for hallucinations to assign the final 1-5 score.

Source Text:

{{Document}}

Summary:

{{Summary}}

Evaluation Form (scores ONLY):

- Consistency (1-5):

Figure D.1: Evaluation prompt used by the LLM-as-a-Judge to assess the *Consistency* dimension. It assigns the model the role of an expert editor to evaluate the logical flow, structural organization, and narrative unity of the generated summary on a discrete 1-5 scale.

You are an expert copyeditor and linguist. Your task is to evaluate a
 → summary based on its Fluency.
 Please read the evaluation criteria and steps carefully, and then rate the
 → summary on a scale of 1 to 5.

Evaluation Criteria - Fluency:

The quality of the summary in terms of grammar, spelling, punctuation, word
 → choice, and sentence structure.

- 1 (Poor): The summary has many severe errors that make it unintelligible
 → or highly unnatural.
- 2 (Fair): The summary has multiple errors that significantly affect the
 → clarity or smoothness of the text.
- 3 (Good): The summary has noticeable errors, but the main points are
 → still comprehensible.
- 4 (Very Good): The summary has only a few minor errors and is generally
 → easy to read and follow.
- 5 (Perfect): The summary has no grammatical or structural errors, uses
 → excellent word choice, and sounds completely natural.

Evaluation Steps (Mental Chain of Thought):

1. Read the summary carefully.
2. Identify any grammatical mistakes, awkward phrasing, spelling, or
 → punctuation errors.
3. Assess how smoothly the text reads: does it sound like natural human
 → language or is it broken?
4. Weigh the frequency and severity of the errors against the 1-5 scale to
 → determine the final score.

Summary:

{{Summary}}

Evaluation Form (scores ONLY):

- Fluency (1-5):

Figure D.2: Evaluation prompt used by the LLM-as-a-Judge to assess the *Fluency* dimension. It assigns the model the role of an expert editor to evaluate the logical flow, structural organization, and narrative unity of the generated summary on a discrete 1-5 scale.

You are an expert content curator. Your task is to evaluate a summary based
→ on its Relevance.
Please read the evaluation criteria and steps carefully, and then rate the
→ summary on a scale of 1 to 5.

Evaluation Criteria - Relevance:

The selection of important content from the source. The summary should
→ include only crucial information. Redundancies and excess, irrelevant
→ information must be penalized.

- 1 (Poor): The summary completely misses the main points of the article
→ and is filled with irrelevant or highly redundant information.
- 2 (Fair): The summary captures a few minor points but misses the core
→ topic, or it includes a massive amount of excess "filler" information.
- 3 (Good): The summary captures the main points reasonably well but
→ includes noticeable redundancies or unnecessary minor details.
- 4 (Very Good): The summary covers the main points very well with only
→ slight inclusions of trivial or mildly redundant information.
- 5 (Perfect): The summary strictly extracts the most vital information. It
→ is highly concise, capturing all core points with zero redundancy.

Evaluation Steps (Mental Chain of Thought):

1. Read the source document and mentally highlight the absolute core topic
→ and key points.
2. Read the summary and identify what information was selected.
3. Check if the summary missed the most important points from the source.
4. Check if the summary includes repetitive statements or trivial details
→ that waste space.
5. Assign a final score based on how perfectly the summary distills only
→ the essential information.

Source Text:

{{Document}}

Summary:

{{Summary}}

Evaluation Form (scores ONLY):

- Relevance (1-5):

Figure D.3: Evaluation prompt used by the LLM-as-a-Judge to assess the *Relevance* dimension. It assigns the model the role of an expert editor to evaluate the logical flow, structural organization, and narrative unity of the generated summary on a discrete 1-5 scale.

You are an expert editor. Your task is to evaluate a summary based on its
 → Coherence.
 Please read the evaluation criteria and steps carefully, and then rate the
 → summary on a scale of 1 to 5.

Evaluation Criteria - Coherence:

The collective quality of all sentences. The summary should be
 → well-structured and well-organized. It should build from sentence to a
 → coherent body of information, not just be a heap of random facts.
 - 1 (Poor): The summary is completely disjointed, illogical, and impossible
 → to follow as a unified text.
 - 2 (Fair): The summary lacks clear structure; sentences feel disconnected
 → or transition abruptly, making it hard to follow.
 - 3 (Good): The summary has a basic structure and covers the main topic,
 → but the logical flow between some sentences is clunky.
 - 4 (Very Good): The summary is well-organized and presents the key points
 → in a clear, logical order with good transitions.
 - 5 (Perfect): The summary flows seamlessly. It builds a highly coherent
 → and logical narrative from beginning to end.

Evaluation Steps (Mental Chain of Thought):

1. Read the source text to understand the intended narrative arc.
2. Read the summary and focus on the transitions between sentences.
3. Check if the ideas connect logically or if the text jumps abruptly
 → between unrelated concepts.
4. Evaluate if the summary reads as a unified, standalone text.
5. Map these structural observations to the 1-5 scale.

Source Text:

{{Document}}

Summary:

{{Summary}}

Evaluation Form (scores ONLY):

- Coherence (1-5):

Figure D.4: Evaluation prompt used by the LLM-as-a-Judge to assess the *Coherence* dimension. It assigns the model the role of an expert editor to evaluate the logical flow, structural organization, and narrative unity of the generated summary on a discrete 1-5 scale.

APPENDIX E

Prompts Used for M-Prometheus Evaluation

The following section details the specialized prompt templates used to evaluate coherence, consistency, fluency, and relevance using the M-Prometheus architecture. Unlike general-purpose LLMs, M-Prometheus requires a highly structured input format to strictly adhere to the provided grading rubrics, as shown in Figures E.1, E.2, E.3, and E.4.

Task Description:

An instruction (might include an Input inside it), a response to evaluate,

→ a reference answer that gets a score of 5, and a score rubric

→ representing a evaluation criteria are given.

1. Write a detailed feedback that assess the quality of the response

→ strictly based on the given score rubric, not evaluating in general.

2. After writing a feedback, write a score that is an integer between 1 and

→ 5. You should refer to the score rubric.

3. The output format should look as follows: "Feedback: (write a feedback

→ for criteria) [RESULT] (an integer number between 1 and 5)"

4. Please do not generate any other opening, closing, and explanations.

The instruction to evaluate:

Evaluate the coherence of the summary. Coherence measures logical structure

→ and narrative flow. The summary must be a unified whole, not a heap of

→ isolated facts. Ideas should be presented in a logical order, with

→ smooth transitions that make it easy to follow. The full source

→ document is provided as an example of a coherent text (score 5), but

→ evaluation focuses on the internal organization of the summary.

Important: The summary can be in any language (e.g., English, Spanish,

→ Catalan). Assess coherence based on the rubric below, independent of

→ the language.

Response to evaluate:

{{Summary}}

Reference Answer (Score 5):

{{Document}}

Score Rubrics:

[Coherence]

Score 1: The summary is completely disjointed, illogical, and impossible to

→ follow as a unified text.

Score 2: The summary lacks clear structure; sentences feel disconnected or

→ transitions are abrupt, making it hard to see how ideas relate.

Score 3: The summary has a basic structure and covers the topic, but the

→ logical flow is clunky; some sentences don't connect well, causing

→ jumps.

Score 4: The summary is well-organized. Key points are presented in a

→ clear, logical sequence with good transitions.

Score 5: The summary flows seamlessly from start to finish. It builds a

→ highly coherent narrative, with each sentence naturally leading to the

→ next, forming a polished, unified text.

Feedback:

Figure E.1: Evaluation prompt specifically formatted for the *M-Prometheus* model to assess the *Coherence* dimension. It directs the architecture to evaluate the structural organization and logical flow of the summary.

Task Description:

An instruction (might include an Input inside it), a response to evaluate,

→ a reference answer that gets a score of 5, and a score rubric

→ representing a evaluation criteria are given.

1. Write a detailed feedback that assess the quality of the response
 - strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and
 - 5. You should refer to the score rubric.
3. The output format should look as follows: "Feedback: (write a feedback
 - for criteria) [RESULT] (an integer number between 1 and 5)"
4. Please do not generate any other opening, closing, and explanations.

The instruction to evaluate:

Evaluate the factual consistency of the summary with respect to the source

→ document. The summary must contain only statements explicitly supported

→ by the document. Any added, invented, or contradictory information is a

→ hallucination and must be penalized. The full source document is

→ provided as the absolute truth reference (score 5).

Important: The source document and the summary can be in any language

→ (e.g., English, Spanish, Catalan) and may not match. You must verify

→ factual alignment across languages, using the rubric below strictly.

Response to evaluate:

{{Summary}}

Reference Answer (Score 5):

{{Document}}

Score Rubrics:

[Consistency]

Score 1: The summary is completely hallucinated, contradictory, or detached

→ from the source. It invents almost everything or directly contradicts

→ facts.

Score 2: The summary contains major factual errors (wrong numbers, dates,

→ names, events) or multiple invented statements that alter meaning.

Score 3: The summary has some minor inaccuracies or a few unsupported

→ details, but the core message remains intact.

Score 4: The summary is almost entirely consistent, with perhaps one

→ negligible misinterpretation or trivial unsupported detail that does

→ not change the overall message.

Score 5: The summary is perfectly aligned with the source. Every claim,

→ including figures, entities and nuances, can be directly verified in

→ the source, with no additions or distortions.

Feedback:

Figure E.2: Evaluation prompt specifically formatted for the *M-Prometheus* model to assess the *Consistency* dimension. It restricts the evaluation strictly to factual alignment and the penalization of hallucinations.

Task Description:

An instruction (might include an Input inside it), a response to evaluate,

- a reference answer that gets a score of 5, and a score rubric
- representing a evaluation criteria are given.

1. Write a detailed feedback that assess the quality of the response
 - strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You should refer to the score rubric.
3. The output format should look as follows: "Feedback: (write a feedback for criteria) [RESULT] (an integer number between 1 and 5)"
4. Please do not generate any other opening, closing, and explanations.

The instruction to evaluate:

Evaluate the fluency of the summary. Fluency assesses linguistic quality:

- grammar, spelling, punctuation, word choice, and sentence structure.
- The summary should read as if written by a competent native speaker,
- with no errors that hinder comprehension. The full source document is
- given as a reference of a correct text (score 5), but the evaluation
- focuses solely on the summary's language quality.

Important: The summary can be in any language (e.g., English, Spanish,

- Catalan). Evaluate fluency strictly based on the rubric below,
- regardless of the language. If the summary contains code-switching or
- multiple languages, assess the overall linguistic correctness and
- naturalness.

Response to evaluate:

{{Summary}}

Reference Answer (Score 5):

{{Document}}

Score Rubrics:

[Fluency]

Score 1: The summary has many severe errors (grammar, spelling,

- punctuation) that make it unintelligible or extremely unnatural.

Score 2: The summary has multiple errors that significantly affect clarity

- or smoothness, forcing re-reading.

Score 3: The summary has noticeable errors (e.g., agreement mistakes,

- awkward phrases) but the main points are still understandable.

Score 4: The summary has only a few minor errors (e.g., a missing comma, a

- slightly unnatural phrase) and is generally easy to read.

Score 5: The summary is flawless: no grammar, spelling, or punctuation

- errors; precise and varied word choice; well-constructed sentences; the
- text flows naturally and elegantly.

Feedback:

Figure E.3: Evaluation prompt specifically formatted for the *M-Prometheus* model to assess the *Fluency* dimension. It focuses the model's analytical capabilities on linguistic quality, grammar, and naturalness.

Task Description:

An instruction (might include an Input inside it), a response to evaluate,

→ a reference answer that gets a score of 5, and a score rubric

→ representing a evaluation criteria are given.

1. Write a detailed feedback that assess the quality of the response
→ strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and
→ 5. You should refer to the score rubric.
3. The output format should look as follows: "Feedback: (write a feedback
→ for criteria) [RESULT] (an integer number between 1 and 5)"
4. Please do not generate any other opening, closing, and explanations.

The instruction to evaluate:

Evaluate the relevance of the summary with respect to the source document.

→ Relevance measures whether the summary includes only the most important

→ information from the document, without redundant or irrelevant content.

→ The full source document is provided as a reference (score 5), as it

→ contains all necessary information.

Important: The source document and the summary can be in any language

→ (e.g., English, Spanish, Catalan) and may not necessarily be in the

→ same language. Evaluate using the criteria below, regardless of the

→ language.

Response to evaluate:

{{Summary}}

Reference Answer (Score 5):

{{Document}}

Score Rubrics:

[Relevance]

Score 1: The summary captures no main ideas from the source. It is filled

→ with irrelevant or highly redundant information.

Score 2: The summary captures only a few minor points but omits the core

→ topic, or contains a massive amount of filler that drowns the main

→ points.

Score 3: The summary covers the main points reasonably well but includes

→ noticeable redundancies or unnecessary minor details.

Score 4: The summary covers the main points very well with only slight

→ inclusions of trivial or mildly redundant information.

Score 5: The summary extracts strictly the most vital information. It is

→ highly concise, capturing all core points with zero redundancy, and

→ only omits non-essential information.

Feedback:

Figure E.4: Evaluation prompt specifically formatted for the *M-Prometheus* model to assess the *Relevance* dimension. It forces the model to weigh the presence of core facts against redundant or trivial information.

APPENDIX F

Prompts Used for Qwen3.6 Model

The following section details the specialized prompt used to execute the summaries using the Qwen3.6:27b model. The detailed prompt is shown in Figure F.1

You are an advanced AI specialized in conditional text summarization. Your
→ task is to read the user's prompt and generate a summary strictly
→ following the parameters provided.

The user will provide a prompt divided into three specific sections. Here
→ is exactly how you must interpret and use each section:

1. ### CONTROL TAGS

The user will provide specific tags to control your output. You must scan
→ for these tags and apply their rules:

- LANGUAGE TAG (You will receive one of these):
 - * <LANG-SAME-AS-INPUT>: Write the summary in the exact same language as
→ the source text.
 - * <SUMMARIZE-IN-CATALAN>: Translate and write the summary in Catalan.
 - * <SUMMARIZE-IN-ENGLISH>: Translate and write the summary in English.
 - * <SUMMARIZE-IN-SPANISH>: Translate and write the summary in Spanish.
- NAMED ENTITIES (NER) TAG (You will receive one of these):
 - * <SELECT-RELEVANT-NERS>: You have full freedom to extract and include
→ the most relevant entities (people, organizations, locations, etc.)
→ from the source text to make a good summary.
 - * <USE-REQUIRED-NERS>: (Implicit if SELECT-RELEVANT-NERS is missing). You
→ must look at the "REQUIRED NERS" section and strictly include those
→ entities.
- LENGTH TAG (You will receive one of these):
 - * <DECIDE-SUMMARY-LENGTH>: You choose the optimal length to convey the
→ main points without losing critical context.
 - * <SHORT-LENGTH>: Provide a very concise, high-level summary ensuring the
→ compression ratio is strictly less than 0.1.
 - * <LARGE-LENGTH>: Provide a detailed, comprehensive summary covering main
→ points and important supporting details, ensuring the compression
→ ratio is greater than or equal to 0.1.

2. ### REQUIRED NERS

This section contains a specific list of entities (people, places,
→ concepts). If the user provided this list (and did not use the
→ <SELECT-RELEVANT-NERS> tag), you MUST include ALL of these entities in
→ your summary. You must build the context of your summary around them.
→ If this section says "N/A", ignore it.

3. ### SOURCE TEXT

This is the original text you must summarize. You must extract all your
→ information EXCLUSIVELY from this text. Do not hallucinate or add
→ outside knowledge.

OUTPUT FORMAT (JSON REQUIRED):

You must output your response EXCLUSIVELY as a valid JSON object.
Do not include any conversational filler, explanations, or markdown code
→ blocks (like ```json).

Your JSON must strictly follow this exact structure:

```
{  
  "summary": "Your final generated summary text goes here"  
}
```