

Politecnico di Torino & EURECOM



**Politecnico
di Torino**



Double Master Degree in
**Cinema and Media Engineering &
Data Science and Engineering**

Academic Year 2025–2026

AI Scientist Storyteller

An LLM-Based Framework and Web Platform for
Persona-Adaptive Scientific Storytelling

Supervisors:

Prof. Raphaël TRONCY
Prof. Luigi DE RUSSIS
Dr. Pasquale LIENA

Candidate:

Alex ARGESE

Abstract (English)

People who are not specialists in a particular field often find it difficult to access scientific research. Large language models (LLMs) are capable of summarizing research articles, but are currently unable to automatically modify the tone, depth, and style for different audiences. This thesis introduces *SciTeller*, an artificial intelligence system designed to transform scientific articles into understandable, engaging stories tailored to specific audiences.

The work integrates narrative generation and natural language processing. It also contains a taxonomy of eight representative personas, along with a curated dataset of scientific articles and publicly shared stories.

Building on this foundation, the system adopts a two-stage pipeline: a *Splitter* module that extracts and organizes the key content of an article into a structured outline, and a *Storyteller* module that rewrites each section into a narrative tailored to the selected persona.

In addition to developing the model, the thesis presents a comprehensive web platform that allows users to upload an article, generate a story, refine it at different levels of granularity, and track all changes through a versioning system. The system was evaluated through quantitative and qualitative analyses, ablation studies, and an initial series of tests with individuals selected based on their profile, specific to each role chosen for this study.

In general, this work demonstrates that personalized adaptive storytelling is a promising approach to improving the accessibility, comprehensibility, and engagement of scientific information for a diverse audience.

Résumé (French)

Les personnes qui ne sont pas spécialistes dans un domaine particulier ont souvent du mal à accéder à la recherche scientifique. Les grands modèles linguistiques (LLM) sont capables de résumer des articles de recherche, mais ne sont actuellement pas en mesure de modifier automatiquement le ton, la profondeur et le style en fonction des différents publics. Cette thèse présente *SciTeller*, un système d'intelligence artificielle conçu pour transformer des articles scientifiques en récits compréhensibles et captivants, adaptés à des publics spécifiques.

Ce travail intègre la génération narrative et le traitement du langage naturel. Elle contient également une taxonomie de huit personnages représentatifs, ainsi qu'un ensemble de données sélectionnées d'articles scientifiques et d'histoires partagées publiquement.

Sur cette base, le système adopte un pipeline en deux étapes : un module *Splitter* qui extrait et organise le contenu clé d'un article dans un plan structuré, et un module *Storyteller* qui réécrit chaque section en un récit adapté au personnage sélectionné.

En plus de développer le modèle, la thèse présente une plateforme web complète qui permet aux utilisateurs de télécharger un article, de générer une histoire, de la peaufiner à différents niveaux de granularité et de suivre toutes les modifications grâce à un système de gestion des versions. Le système a été évalué à l'aide d'analyses quantitatives et qualitatives, d'études d'ablation et d'une première série de tests réalisés auprès de personnes sélectionnées en fonction de leur profil, spécifique à chaque rôle choisi pour cette étude.

De manière générale, ce travail démontre que la narration adaptative personnalisée est une approche prometteuse pour améliorer l'accessibilité, la compréhensibilité et l'intérêt des informations scientifiques pour un public diversifié.

Abstract (Italian)

Le persone che non sono specialiste in un determinato ambito incontrano spesso difficoltà nell'accedere alla ricerca scientifica. I grandi modelli linguistici (LLM) sono in grado di riassumere articoli di ricerca, ma attualmente non riescono a modificare automaticamente tono, profondità e stile in funzione di pubblici diversi. Questa tesi introduce *SciTeller*, un sistema di intelligenza artificiale progettato per trasformare articoli scientifici in racconti comprensibili e coinvolgenti, adattati a specifici tipi di pubblico.

Il lavoro integra tecniche di generazione narrativa e di elaborazione del linguaggio naturale. Include inoltre una tassonomia di otto personas rappresentative, insieme a un dataset curato di articoli scientifici e storie condivise pubblicamente.

Su queste basi, il sistema adotta una pipeline in due fasi: un modulo *Splitter*, che estrae e organizza i contenuti chiave di un articolo in una struttura formale, e un modulo *Storyteller*, che riscrive ciascuna sezione sotto forma di narrazione, adattandola alla persona selezionata.

Oltre allo sviluppo del modello, la tesi presenta una piattaforma web completa che consente agli utenti di caricare un articolo, generare una storia, raffinarla a diversi livelli di granularità e tracciare tutte le modifiche attraverso un sistema di versionamento. Il sistema è stato valutato mediante analisi quantitative e qualitative, studi di ablation e una prima serie di test condotti su individui selezionati in base al loro profilo, specifico per ciascun ruolo considerato in questo studio.

Nel complesso, questo lavoro dimostra che la narrazione adattiva personalizzata rappresenta un approccio promettente per migliorare l'accessibilità, la comprensibilità e il coinvolgimento delle informazioni scientifiche per un pubblico eterogeneo.

Contents

Abstract (English)	I
Résumé (French)	II
Abstract (Italian)	III
Contents	VI
1 Introduction	1
1.1 Challenges in Scientific Communication	1
1.2 Thesis Objectives	2
1.3 Contributions	2
1.4 Structure of the Thesis	3
I AI System Development: Data Pipeline and Generative Modeling	5
2 Background and Related Work	6
2.1 Scientific Communication and Personas	6
2.1.1 Personas in Communication and HCI	6
2.1.2 Taxonomies of Audiences for Scientific Content	7
2.2 Large Language Models for Summarization and Storytelling	7
2.3 Persona-Aware Scientific Communication with LLMs	8
2.4 Grounded Generation through Structure Extraction and Alignment	9
2.5 Evaluation and Remaining Gaps in Scientific Storytelling Systems	9
3 Methodology	10
3.1 Challenges and Requirements	10
3.2 System Overview	11
3.3 Methodology for the Splitter	11
3.4 Methodology for the Storyteller	14
3.5 Design Rationale and Modularity	15
4 Dataset Construction	16
4.1 Collection of Scientific Papers	16
4.2 Parsing and Cleaning	17
4.2.1 Parsing of Scientific Papers (PDF)	17
4.2.2 Parsing of Human-Written Stories (Web Articles)	19
4.3 Logical Segmentation of Papers	20
4.4 Story–Paper Alignment	22

4.5	Construction of the Splitter Dataset	24
4.5.1	Persona Mapping from Human-Written Stories	24
4.5.2	Synthetic Outline Generation to Increase Persona Coverage	25
4.5.3	Examples of Dataset Entries	26
4.5.4	Dataset Size and Splits	27
4.6	Construction of the Storyteller Dataset	27
4.6.1	Section-Level Mapping from Stories to Outlines	27
4.6.2	LLM-Assisted Cleaning and Normalisation	28
4.6.3	Assembling Input–Output Pairs for Storyteller	29
4.6.4	Dataset size & Splits	30
4.7	Dataset Statistics	30
5	Experiments	32
5.1	Baselines	32
5.2	Configuration and Training Setup	33
5.3	Experimental Setup	35
5.4	Preliminary Experiments	36
5.4.1	Experiments with Flan-T5	37
5.4.2	Experiments with LongT5	38
5.4.3	Experiments with Mistral 7B	39
5.4.4	Experiments with Qwen 32B (Unified Model)	40
5.4.5	Emergence of the Two-Stage Pipeline	41
5.5	LLM Selection	42
5.6	Summary and Lessons Learned	43
6	Evaluation	45
6.1	Evaluation Metrics	45
6.2	Quantitative Results	48
6.2.1	Performance of the Final Splitter–Storyteller Pipeline	49
6.2.2	Results of the Baseline and Intermediate Models	49
6.2.3	Progression Across Models	50
6.3	Hallucination Evaluation	51
6.3.1	Capitalised Words as Entity Proxies	51
6.3.2	NER-Based Detection (SpaCy PERSON/ORG)	52
6.3.3	MIRAGE Rewrite-Consistency Scoring	52
6.3.4	LLM-as-a-Judge (Qwen-7B)	52
6.3.5	Hybrid Hallucination Detection (HHD)	53
6.3.6	LLM-as-a-judge (GPT 5.1)	54
6.3.7	Final Operational Choice	54
6.4	Ablation Study	55
6.4.1	Quantitative Results	55
6.4.2	Qualitative Results	61
6.4.3	Qualitative vs. Quantitative Evidence	66
6.5	Discussion	67
6.5.1	Effectiveness of the Two-Stage Architecture	67
6.5.2	Persona Adaptation and Controlled Creativity	68
6.5.3	Interpretation of the Evaluation Methodology	68
6.5.4	Implications for Model Design and Future Work	69

II Web Platform Development: Application Architecture and User Interaction	70
7 Design	71
7.1 Related Systems: A Benchmarking Perspective	71
7.1.1 Landscape of AI Systems for Scientific Content	71
7.1.2 Benchmarking Criteria	72
7.1.3 Comparative Analysis of Existing Systems	72
7.1.4 Synthesis and Positioning of SciTeller	73
7.2 Design Goals	74
7.3 Core Features	75
7.4 User Workflow & Interface Components	78
7.5 Design Rationale	85
8 UI and Implementation	88
8.1 Overall Architecture	88
8.2 Frontend Implementation	89
8.2.1 Routing and main pages	89
8.2.2 Progress feedback via SSE (generation pipeline)	90
8.2.3 Stories workspace: state model and interactions	90
8.2.4 ControlPanel and paragraph variants	91
8.2.5 “Read on paper” workflow	91
8.3 Backend Implementation (FastAPI)	92
8.3.1 Reverse proxy and request routing	93
8.4 Backend Implementation (Next.js API Layer)	93
8.5 Database Architecture	96
8.5.1 Database schema overview	97
8.6 GPU Inference Service (VM)	98
8.7 Dockerization and Deployment	100
8.8 Scalability and Performance Considerations	100
9 User Study	102
9.1 Participants	102
9.2 Tasks	103
9.3 Questionnaire Design	104
9.4 Results	104
9.4.1 Quantitative Results	104
9.4.2 Qualitative Insights	107
9.5 Implications for the System	107
10 Conclusions	109
10.1 Answers to Research Questions	109
10.2 Future Work	111
10.3 Final Remarks	112

Chapter 1

Introduction

Scientific knowledge is increasingly abundant, but this does not necessarily mean that it is more accessible. Most research articles are written for specialists and assume a level of background knowledge that non-experts generally do not possess.

As widely discussed in scientific communication research [16], this gap makes it significantly more difficult for a wider audience to understand the relevance and implications of new discoveries, keeping scientific research a matter for the few.

Large language models (LLMs) can help summarize and rephrase complex content, but struggle to adapt tone, depth, and style to different audiences. Current LLM-based summarization systems [19] are effective in compressing information, but rarely take into account prior knowledge, the reader’s communication needs, the level of detail required for the summary, and therefore which concepts to explain and which can be omitted.

Furthermore, it is well known that LLMs generate hallucinations (i.e., factually incorrect statements presented with confidence), which poses additional challenges for scientific communication. Recent surveys [10] highlight how these risks complicate the adoption of LLMs in high-risk or information-sensitive areas.

To address these issues, this thesis introduces *SciTeller*, an artificial intelligence system designed to transform scientific articles into stories tailored to different audience profiles. Personas, widely used in the interaction design to represent heterogeneous user groups, provide a structured way to model the narrative style and level of detail. *SciTeller* integrates personas with narrative generation to create more accessible scientific stories.

1.1 Challenges in Scientific Communication

Scientific communication is inherently complex. Research articles are written primarily for experts in a given field and rely on shared background knowledge, mathematical formalism, and field-specific terminology. This makes them effective for academic exchange, but difficult to interpret for readers who do not possess the same level of expertise. Understanding an article often requires familiarity with previous work, methodological conventions, and implicit assumptions that are rarely explicitly explained.

One of the main obstacles is linguistic in nature: scientific writing uses technical jargon, acronyms, complex syntactic structures, and precise but abstract formulations. Even when the concepts are not inherently complex, the way they are expressed can place a significant cognitive burden on non-specialists. Length and structure further contribute to the difficulty: research articles are typically organized into sections such as *methodology*, *experimental setup*, and *related work*, which may not align with how non-experts naturally process information. These sections often require readers to navigate through multiple

levels of detail before arriving at an understanding of the main ideas and contributions, and this is often not enough, as simpler examples are also needed to grasp a concept.

Furthermore, the audience is not homogeneous. In addition to prior knowledge, as mentioned above, readers’ needs and expectations can also vary considerably. For example, a journalist may seek social impact and clarity; a politician may seek implications and actionable insights; a researcher may expect technical depth and accuracy; and an investor may want to know why to invest in a particular project and, therefore, explore all its economic aspects, perhaps with an emphasis on comparisons with similar systems. A single summary or explanation cannot adequately satisfy all these profiles.

To address this heterogeneity, the concept of *personas* (i.e., representative audience profiles) offers a structured way to think about the diversity of readers’ backgrounds and communication needs. Personas help identify the level of detail to include, the tone to adopt, the concepts that require explanation, and the aspects of the research that should be emphasized. Incorporating personas into scientific storytelling thus provides a principled approach to tailoring content to different audiences and reducing the communication barriers that typically arise between experts and non-experts.

1.2 Thesis Objectives

The purpose of this thesis is to encompass natural language processing, narrative generation, and human-computer interaction. The main objectives are as follows:

- **Characterize the problem of adapting scientific communication** to different audience profiles, identifying the key barriers that prevent non-experts from understanding research articles.
- **Investigate how large language models can support scientific storytelling**, with a focus on adapting tone, depth, and style to audience needs, while preserving the factual basis of the original paper.
- **Develop a methodological framework** for transforming scientific papers into structured, audience-appropriate narratives, integrating document understanding with controllable text generation.
- **Build the necessary datasets** to support this framework, including aligned scientific documents and public-facing stories suitable for training and evaluation.
- **Experiment with and evaluate different model families** to identify their strengths and limitations in document analysis, narrative generation, and audience adaptation.
- **Design and implement an interactive system** that enables users to generate, inspect, and refine audience-adapted stories starting from a scientific paper.
- **Assess the system through a user-centered evaluation**, measuring usability, clarity, and perceived usefulness across readers with different backgrounds.

1.3 Contributions

This thesis offers contributions across three complementary dimensions: methodological advances in AI-based storytelling, system engineering for interactive scientific communication, and empirical evaluation.

The main contributions are as follows:

Methodological and AI Contributions

It formalises how audience characteristics can guide narrative generation, while ensuring that the resulting stories remain faithful to the source article. To this end, the work introduces a two-stage system consisting of a Splitter, which extracts a structured representation of the article, and a Storyteller, which produces narratives, both tailored to specific reader profiles.

The development of this system required the creation of dedicated datasets for both modules, containing aligned document structures, relevant extracts, and examples of person-specific stories suitable for fine-tuning. Finally, the thesis examines the capabilities and limitations of current LLM families in understanding long documents, person conditioning, and hallucination control, motivating the architectural choices made in this work.

System and Engineering Contributions

This work also includes the development of a comprehensive web platform that integrates the entire pipeline, allowing users to upload a document, choose the target persona, and generate a story. A dual interactive view links narrative paragraphs to specific regions of the document through similarity-based mapping and visual highlights, promoting transparency and interpretability. In addition, a mechanism for editing at different levels of granularity (story, section, and paragraph) and versioning makes the review process controllable, traceable, and iterative.

Evaluation and User Study Contributions

The system is evaluated through studies focused on both the model and the user. Splitter and Storyteller undergo scientific evaluation that includes quantitative metrics, ablation analysis, and investigations into hallucination detection. Complementing this, a user study involving participants from diverse backgrounds evaluates the platform’s usability, clarity, effectiveness of personalisation, and overall value for scientific communication.

1.4 Structure of the Thesis

This thesis is organized into two main parts, reflecting both the methodological contributions in artificial intelligence and the interactive system.

Part I — AI System Development: Data Pipeline and Generative Modeling

It presents the scientific and technical foundations of the AI system. It introduces the theoretical context, formulates the problem of personalised scientific storytelling, constructs the datasets needed to train the model, and details the two-stage pipeline consisting of the Splitter and Storyteller modules. The part then describes the training procedures, preliminary and final experiments, and model evaluation through quantitative metrics, ablation studies, and hallucination analysis.

Research Questions for Part I:

- **RQ1:** How can long scientific papers be transformed into organised textual representation that support coherent storytelling?

- **RQ2:** How effectively can generative models adapt scientific narratives to different audience profiles?

Part II — Web Platform Development: Application Architecture and User Interaction It focuses on the design and implementation of the interactive system built around the AI pipeline. It presents the architecture of the web platform, its user interface, and the mechanisms that support versioning, refinement, and alignment between articles and stories. It also reports the results of a user study conducted to evaluate the usability, perceived clarity, and effectiveness of personalized scientific storytelling.

Research Questions for Part II:

- **RQ3:** How can users be given intuitive control over story generation, refinement, and version navigation while maintaining transparency and traceability?

Part I

AI System Development: Data Pipeline and Generative Modeling

Chapter 2

Background and Related Work

2.1 Scientific Communication and Personas

Scientific communication research examines how scientific knowledge can be translated, framed, and contextualised for audiences with varying levels of expertise. Key contributions about conceptual tools and models for understanding how scientific information can be adapted to diverse audiences include studies on audience framing, audience engagement, narrative strategies, and the cognitive processes involved in interpreting complex scientific content.

These works emphasise that effective communication requires deliberate choices about emphasis, analogy, structure, and level of detail, depending on the reader’s background and goals [16].

2.1.1 Personas in Communication and HCI

Personas are widely used in human-computer interaction and communication design to model the needs, expectations, and behaviors of different user groups. Personas are defined as fictional archetypes based on concrete evidence. They help designers, developers and communicators anticipate how different audiences interpret information and what forms of explanation, detail or contextualization they require [4], [9].

A key concept drawn from the literature on interaction design, particularly from *About Face* [4], is that personas serve as cognitive models of users: they encode levels of expertise, familiar terminology, mental models, cognitive abilities, and contexts of use. These dimensions are fundamental to communication, especially in multidisciplinary scientific contexts where audiences vary greatly in terms of background knowledge.

By explicitly modeling these attributes, characters allow scientific narratives to be calibrated along different axes: degree of technical detail, conceptual structure, use of metaphors or analogies, and appropriate granularity of explanation. In communication activities, all these points promote clearer and more consistent coordination between the message and the audience.

However, *About Face* also warns against building characters based on stereotypes, hunches, or anecdotal impressions. Personas must be based on concrete data, validated against actual user behavior, and updated as the audience evolves. This requirement is even more evident in data-driven or AI-mediated systems, because ensuring the accuracy of persona profiles requires realistic data sets, consistent modeling of user characteristics, and safeguards to prevent projection biases or oversimplified generalizations.

Complementing this perspective, research on data-driven personas [9] highlights that

personas are not only representative tools, but also communicative artifacts whose reception depends on the transparency with which the data and methods on which they are based are conveyed. Transparency can increase the clarity and perceived completeness of personas, thereby enhancing their communicative value, but it can also reduce their perceived credibility by making their algorithmic construction more explicit. Taken together, these findings position personas as structured mediators between complex information and diverse audiences, enabling communication systems, including LLM-based storytellers, to deliver personalized, intelligible, and user-centered narratives.

2.1.2 Taxonomies of Audiences for Scientific Content

Audience taxonomies play a central role in scientific communication research, as they describe how scientific information is received, interpreted and used by different social groups. Classic models distinguish audiences based on dimensions such as competence, prior knowledge, professional role and communication objectives. In particular, the National Academies of Sciences, Engineering, and Medicine [16] and Capili and Anastasi [2] identify recurring target categories, including the general public, students, educators, researchers, policymakers, and journalists. These groups represent the main recipients of scientific communication and form the backbone of contemporary research on scientific communication.

Based on these established classifications, this project defines a fixed and consistent set of characters used in all annotation and narration activities. The design criteria were twofold: coverage, to represent the actors most involved in AI communication in society and industry; and clarity, to ensure distinct and well-separated characters that would allow for controlled narrative adaptation.

While six characters correspond directly to canonical categories found in the literature (general public, students, professors, researchers, policymakers, journalists), two additional roles, *Investor* and *Developers*, have been incorporated to represent central stakeholders in modern AI ecosystems. Their inclusion is justified by both of the sources mentioned above, considering that they explicitly mention adjacent categories such as financiers, sponsors, technologists and engineers, who occupy similar positions in the research and innovation chain.

However, given the significant overlap in terms of skills, technical knowledge and communication needs between developers and academic researchers, these two roles have been merged into a single highly specialised profile, that of *researchers and engineers*. This combined profile includes those who produce, implement and evaluate advanced AI systems and therefore require precise methodological details, guidance on reproducibility and specialised terminology.

The final taxonomy, ordered according to increasing levels of expertise, is presented in Table 2.1. These personas serve as structured audience models that enable the AI system to modulate granularity, tone, terminology, and framing in a consistent and interpretable manner, ensuring that the generated narratives remain aligned with the expectations and capabilities of different reader groups.

2.2 Large Language Models for Summarization and Storytelling

Summarization techniques are divided into extractive methods, which select key passages directly from the source text, and abstractive methods, which generate new sentences that

Table 2.1: Final personas selected for this project.

Persona	Expertise	Communication Need	Goal
General Public	Low	Understand what AI is and why it matters	Accessible language and intuitive metaphors.
Investor	Low–Medium	Identify AI trends, opportunities, risks	Concise and high-level strategic insights.
Student (Bachelor/Master/PHD)	Medium	Learn AI fundamentals	Clear explanations and structured scaffolding.
Journalist	Medium	Report accurately while simplifying technical details	Ensure clarity, accuracy, and contextual framing.
Policy Maker	Medium–High	Societal, ethical, and regulatory implications of AI	Receive impact summaries and decision-ready guidance.
Professor (University approach)	High	Teach AI concepts effectively	Reframe explanations pedagogically for learners.
Researchers & Engineers	High	Require methodological detail and reproducibility cues	Work with technical terminology and rigorous detail.

convey the underlying meaning. Transformer-based models, such as T5 [19] and BART [14], have greatly improved abstractive summarization, enabling the coherent generation of multiple sentences in long texts. The latest decoder-only models, such as LLaMA [22] and Mistral [11], extend these capabilities with longer context windows and improved generation quality. However, scientific summarization remains challenging due to domain-specific terminology, the internal complexity of documents, and the need to maintain factual accuracy [1].

These challenges are amplified in domain-specific abstract summarization, where models must handle extremely long inputs, frequent jargon, and complex rhetorical structures. As the article points out, current LLMs often struggle to meet the computational requirements of processing long documents, exhibit hallucinations when they lack sufficient domain grounding, and face domain shift when trained predominantly on generic corpora. Furthermore, evaluation becomes problematic: automated metrics are poorly correlated with human judgment in specialized fields, making it difficult to reliably assess the quality of summarization. No existing approach yet provides an integrated solution to all these issues, highlighting the need for advances in long-context modeling, domain adaptation, and more robust evaluation methods.

Research on narrative generation, on the other hand, explores how models can generate coherent, structured stories rather than isolated sentences. Previous work has examined controllable narration, educational narrative support, and coherence modeling. Generating long texts requires maintaining coherence between sections, managing discursive transitions, and producing engaging explanations. An ongoing challenge is controllability: adjusting tone, depth, style, and framing while preserving coherence and fidelity to the original content[21].

2.3 Persona-Aware Scientific Communication with LLMs

Large language models (LLMs) can adapt tone, style, and level of detail to different audiences by conditioning generation based on personalities. Approaches include prompt-based guidance, tuning on personality-specific corpora, and embedding-level control signals. Al-

though these methods enable models to emulate communication styles and adjust complexity, it remains difficult to achieve personality adaptation without sacrificing factual accuracy, especially when simplifying scientific content for non-expert readers. Previous studies show that LLMs tend to overgeneralize, omit crucial methodological details, or introduce unsupported claims when reformulating technical material [10].

Ensuring readability and suitability for the audience therefore requires a careful balance between stylistic adaptation and generation that is grounded in and consistent with the source, evaluated through metrics of consistency, readability indices, and human judgments of clarity, engagement, and tone. Furthermore, person-based adaptation is particularly relevant in scientific storytelling because audiences vary greatly in terms of prior knowledge, expectations, and goals. Communication research highlights that effective audience design involves choosing appropriate metaphors, adjusting cognitive load, and emphasizing different aspects of the source material depending on the recipient. LLMs, however, do not have an intrinsic understanding of these communicative constraints and often treat person-related instructions superficially rather than as an in-depth modality [20].

2.4 Grounded Generation through Structure Extraction and Alignment

Scientific articles contain well-defined rhetorical structures (background, methods, results, contributions) that strongly influence how information should be summarized or narrated. Document structure extraction techniques address this problem by segmenting articles into coherent units, detecting section boundaries, and classifying rhetorical roles [12]. When narrative generation is conditioned by these structured units, models can ground their outputs more reliably, avoiding the need to process the entire document at once and reducing the risk of fabricated or misplaced information. Fidelity evaluation highlights that hallucinations occur when generated statements are not supported by the source; surveys such as that by Ji et al. [10] distinguish between intrinsic and extrinsic hallucinations and outline evaluation methods that include entity consistency checks, retrieval-based comparisons, and LLM-as-judge techniques [15]. Effective alignment between document structure and generated narrative is therefore critical for producing accurate and verifiable scientific stories.

2.5 Evaluation and Remaining Gaps in Scientific Storytelling Systems

Despite advances in people management, document segmentation, and text evaluation, existing work addresses these components in isolation. No current system integrates:

1. scientific narration tailored to the target audience;
2. generation based on explicit extraction and alignment of rhetorical structure;
3. interactive refinement cycles that allow users to explore, compare, and modify narratives.

This combination is essential to ensure both accessibility and scientific reliability. These gaps motivate the development of an integrated framework that unifies document understanding, person-conditioned generation, and user-centered interaction, which is the main focus of this project.

Chapter 3

Methodology

3.1 Challenges and Requirements

Transforming complete scientific articles into coherent narratives tailored to the target audience involves addressing a series of interconnected scientific and technical challenges. Scientific documents are often lengthy, written in a very dense style using complicated words, and contain heterogeneous structures such as sections, subsections, figures, tables, and mathematical expressions. Models must therefore handle large contexts while preserving the logical flow of the original text. If a given model does not accept more than a certain number of tokens, some possible solutions would be either to truncate the text, losing essential meaning, or to identify which parts of an article are relevant to the narrative, which requires selection. But even with this approach, condensing and reorganizing information is crucial because there is a risk of losing essential meaning.

A second major challenge concerns adaptation to personas. Different audiences expect different levels of technical depth, tone, and explanatory detail, which requires control not only over the information presented, but also over how it is phrased. This makes generation a conditional process in which style, complexity, and narrative choices must be aligned with a predefined persona profile. Performing this consistently is no easy task, especially when it comes to balancing clarity, accuracy, and expertise.

Ensuring faithfulness is equally crucial. Large language models are prone to hallucinations, introducing entities, statements, or numerical values that are not present in the source. In the context of scientific communication, such deviations can limit the educational or operational value of the system and, above all, undermine its reliability. To avoid hallucinations, both architectural strategies and adequate evaluation signals are necessary. In addition, users and reviewers need forms of controllability and interpretability that make the system’s behavior traceable, allowing them to inspect intermediate steps or verify that the generated content is based on the input document.

These challenges give rise to the fundamental requirements of the system. First, the system must accurately extract the structure of scientific documents, preserving section boundaries and allowing downstream models to reason over well-defined units of text. Second, the generation process must be faithful and aligned with the persona, ensuring that the results remain based on the article while adapting the tone and depth appropriately. The architecture must be modular and scalable, allowing components such as extraction, schema generation, and narration to evolve independently. Furthermore, since users may need to refine or iterate on the generated story, the system must support interactive editing and versioning, allowing local regeneration of sections while maintaining global consistency. Finally, users may want to verify the reliability of what they read by comparing it with the

original text, so the system must also provide a mechanism for comparing input (scientific article) and output (generated story).

3.2 System Overview

The system is designed as a modular pipeline that transforms a scientific article into a persona-adapted narrative through two main stages. This architecture separates content understanding from content generation, ensuring better control, scalability, and interpretability throughout the workflow.

The first phase, the **Splitter**, analyzes the entire scientific article and produces a schematic structure that highlights the fundamental ideas useful for the narrative. Instead of directly generating text, this component organizes the content into a reduced set of high-level sections that form the backbone of the story. The resulting structure reflects the main contributions, methods, and results, but above all, it is tailored to the target audience. The goal is to allow the system to adapt the selection and organization of information: the same article may require an introductory section that provides context and background for the general public, while for a researcher it would be more useful to focus on technical aspects, methodological details, or quantitative results. In this way, the Splitter does not simply summarize the content and identify its main general parts, but builds a narrative base that varies dynamically according to the communication needs of the chosen persona.

The second phase, called **Storyteller**, is responsible for generating the final narrative. It takes the outline extracted from the Splitter, along with the persona specifications and the text of the scientific article, and generates the final narrative, producing text for each element of the outline. In this way, the model is based not only on scientific content, but also on the tone, depth, style, and expected content associated with the chosen persona. This design allows for consistent stylistic control throughout the story.

To handle the entire process, the system retrieves only the relevant parts of the document needed for the current generation phase, improving focus and reducing the likelihood of errors, especially for a problem related to the context window. Basic alignment mechanisms ensure that the narrative produced remains faithful to the scientific content while following the structure defined by the Splitter. Furthermore, although the complete interaction model is described in detail in later chapters, it is worth noting that the process supports iterative refinement workflows, allowing users to regenerate individual sections and paragraphs while preserving overall consistency.

3.3 Methodology for the Splitter

The Splitter is designed as a dedicated component whose main objective is to reduce the complexity of a complete scientific article and transform it into a compact structure that is, above all, customized for personas. Rather than attempting to generate a complete narrative directly, the Splitter operates at a higher level of abstraction: it reads the article and produces a structured list of section titles accompanied by short summaries and key concepts. In this way, the model distills the main contributions, methods, and results of the article into a limited number of coherent segments that can then be expanded by the Storyteller into complete narrative sections.

A separate module is necessary because language models for long documents, when used in a single pass, tend to mix content selection, structuring, and style adaptation in a way that is not very transparent. This makes it more difficult to control and interpret

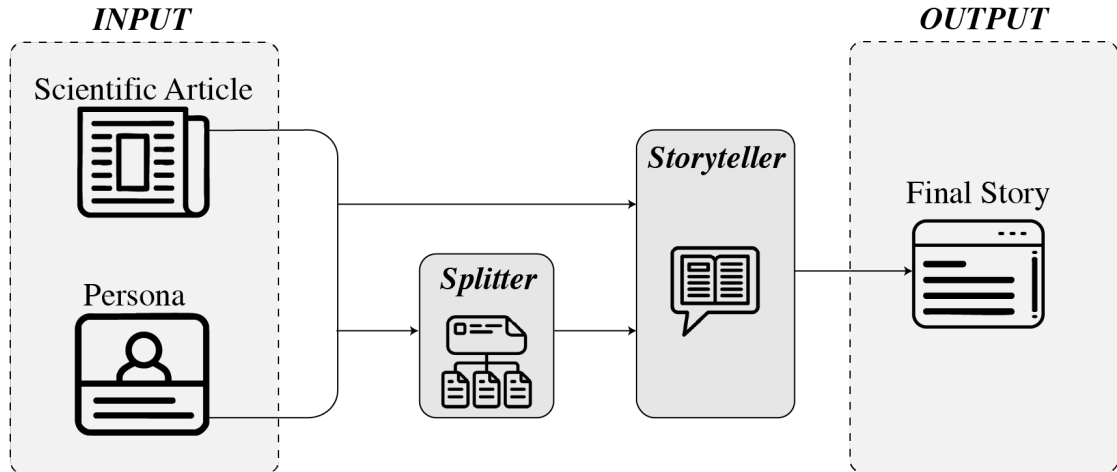


Figure 3.1: High-level overview of the Splitter–Storyteller pipeline.

the system’s behavior: it becomes difficult to understand which parts of the document are actually driving the generation and how the narrative structure emerges.

With the introduction of the Splitter, the process is broken down into an explicit planning phase followed by a generation phase. This makes the process of story structure and adaptability per persona much more flexible. The planning phase focuses on identifying what should be said and in what order, delegating to the Storyteller the simple task of deciding how it should be said for a given character. In simple terms, the Splitter decides **what to tell** and the Storyteller decides **how to tell** it. This separation also mitigates some of the limitations of long-context models: instead of relying on a single, very long prompt, the system can work with shorter, more focused inputs and maintain a clear intermediate representation that is easy to inspect and debug.

Conceptually, the expected input for the Splitter is both a scientific article, provided as pre-segmented text obtained from PDF text extraction, and the chosen persona. The expected output is a structured outline represented as a list of sections, where each item contains: (i) a concise title appropriate to the persona and (ii) a brief description with one or two sentences summarizing the content and highlighting key concepts, designed to be useful to the Storyteller. This outline is sensitive to the persona: the same article can produce different section titles and emphases depending on whether the target persona is (a professor, a student, a policy maker, etc.).

To ensure robustness and controllability, the Splitter operates according to a set of explicit constraints. First, it is programmed to produce a fixed and limited number of sections (usually five), favoring an overview of the article that is not fragmented. Second, the structure must follow a simple hierarchy: section titles must be short, distinct from each other, and ordered to form a coherent progression from context and rationale to methodology, results, and implications, when appropriate for the persona. Thirdly, the Splitter does not work blindly on the entire raw document: context retrieval and safe truncation mechanisms are used to select and prioritize the most informative parts of the article, keeping the input within the contextual window of the model. Finally, all titles and descriptions are constrained to remain anchored to the source document, avoiding invented entities or statements; this anchoring is subsequently exploited by the Storyteller, which uses the outline as a faithful and interpretable template to generate the final story.

To concretely illustrate the behaviour of the Splitter, below there are two real outline examples generated from the well-known paper *Attention Is All You Need* [23]. Despite operating on the same underlying scientific content, the system produces markedly different high-level structures depending on the selected persona, adapting both the narrative focus and the level of detail.

Example: Output for the *General Public* persona. In this case, the Splitter prioritises accessibility, conceptual clarity, and introductory explanations. The resulting outline highlights motivations and simplified descriptions of the model components.

```
[
  { "title": "Introduction to Transformers",
    "description": "Introduces the Transformer model, explaining why
    ↪ traditional RNNs are limited." },

  { "title": "Why Attention Matters",
    "description": "Explains how attention mechanisms improve sequence
    ↪ transduction models compared to RNNs and CNNs." },

  { "title": "Model Architecture Overview",
    "description": "Describes the basic components of the Transformer
    ↪ model, focusing on self-attention and feed-forward networks." },

  { "title": "Experimental Results",
    "description": "Details the performance of the Transformer model on
    ↪ translation tasks, showing improved BLEU scores." },

  { "title": "Positional Encoding Explanation",
    "description": "Describes how positional encodings help the Transformer
    ↪ model understand sequence order." }
]
```

Example: Output for the *Policy Maker* persona. For an audience of policy makers, the Splitter shifts the emphasis towards societal implications, governance aspects, and broader impacts, while still preserving the essential technical framing of the paper.

```
[
  { "title": "Introduction to Transformers",
    "description": "Introduces the limitations of recurrent models and
    ↪ proposes the Transformer architecture." },

  { "title": "Model Architecture Overview",
    "description": "Describes the basic components of the Transformer
    ↪ model, including self-attention and feed-forward networks." },

  { "title": "Ethical Implications and Governance",
    "description": "Discusses potential ethical concerns and governance
    ↪ issues related to the widespread adoption of Transformer models."
    ↪ },
]
```

```

{ "title": "Experimental Results and Performance",
  "description": "Details the experimental setup and results of the
    ↪ Transformer model on machine translation tasks." },

{ "title": "Future Directions and Applications",
  "description": "Outlines future research directions and potential
    ↪ applications of the Transformer model beyond translation." }
]

```

3.4 Methodology for the Storyteller

The Storyteller, as mentioned above and as may have been inferred so far, has the task of transforming the outline produced by the Splitter into a complete scientific story tailored to a specific audience. Its goal is to transform each element of the outline, defined by a section title and a brief description, into a coherent narrative section written in natural language.

Persona adaptation is fundamental to this component. Each persona is characterized by a small set of attributes, such as expected skills, communication goal, and preferred style (e.g., educational tone for students, governance-oriented approach for policymakers, or concise technical language for researchers and engineers). These attributes are encoded in a person specification that is included in the prompt as a high-level guide. The Storyteller uses this guidance to modulate the tone, depth, and use of examples or metaphors: for example, favoring intuitive analogies and simplified explanations for the general public, while emphasizing methodological details, metrics, and limitations for researchers and engineers.

The generation process operates at the level of individual sections. For each item in the structure, Storyteller receives three main inputs: (i) the person specification, (ii) the title and description of the target section from the structure, and (iii) a portion of the paper context retrieved from the original document. A retrieval module first segments the clean paper text into overlapping paragraphs, then selects the most relevant fragments for the current section through a similarity search. The retrieved context is inserted into a section-specific prompt, which instructs the model to produce a short narrative passage corresponding to the target title and faithful to the context provided. The result is a narrative text, which is then stored together with the section title; once all sections have been generated, a separate prompt is used to propose a concise, personalized title based on the same document.

Generating sections is preferable to generating the entire article for several reasons. First, conditioning the model on a single section at a time reduces the effective length of the context and allows the retrieval phase to focus on a small, relevant subset of the article, improving fidelity and reducing distraction from unrelated content. Second, it makes the system more controllable and interpretable: each section can be inspected, regenerated, or revised independently, while preserving the overall structure of the schema. Consistency between sections is maintained by reusing the same person specification throughout the document, following the order defined in the schema, and including a slight transition guide (e.g., linking the beginning of each section to the previous one).

To mitigate hallucinations, Storyteller combines prompt-level constraints with retrieval-based grounding. Prompts explicitly instruct the model to rely only on entities and details that appear in the context of the retrieved article and avoid inventing datasets, numerical values, or institutional affiliations. Furthermore, the retrieved context is constructed to

be sufficiently rich and specific for each section, so that key concepts and terminology are always available to the model. Lightweight post-processing further checks for unsupported capitalized entities and can discard or give less weight to sentences that introduce names not present in the source context. Overall, Storyteller is designed as a purely generative component: versioning, editing, and interactive refinement mechanisms, such as the regeneration of individual sections at the user’s request, are implemented at the web application level and are therefore described separately in Part II.

Illustrative example of persona-adapted generation. As in the previous illustrations, the following example is based on the paper *Attention Is All You Need* [23] and is included solely to show how the Storyteller adapts tone and depth for different personas. The same section, *Model Architecture Overview*, is presented below in two persona-specific versions.

General Public. *Transformers revolutionized machine translation by simplifying the architecture and removing the need for recurrent networks. Instead of processing words one at a time, the model uses attention mechanisms to look at all parts of a sentence simultaneously, allowing for faster and more accurate understanding. This parallel processing makes training and inference significantly more efficient.*

Policy Makers. *Transformer models eliminate both recurrent and convolutional layers, relying solely on self-attention mechanisms to process sequences. The encoder maps the input into continuous vector representations, which the decoder then uses to generate outputs through attention-based interactions. This architecture underpins many state-of-the-art systems and raises governance considerations regarding transparency and model interpretability.*

3.5 Design Rationale and Modularity

The overall system is built around a deliberately modular architecture that separates content planning from narrative generation. This design choice improves interpretability: each component exposes a clear intermediate representation, allowing users and developers to verify how the document is transformed before the final story is produced. Modularity also improves error isolation, as problems in content extraction, structure creation, or narrative generation can be diagnosed and corrected without affecting the entire pipeline.

A modular structure offers considerable flexibility for future extensions. Each component can be independently updated, replaced, or optimized, allowing the system to evolve as new models, retrieval strategies, or prompting techniques become available. For example, the Splitter and Storyteller can be evaluated and improved separately, allowing for targeted experimentation on narrative structure, personas adaptation, or grounding mechanisms without interfering with other parts of the pipeline.

This architecture also facilitates systematic evaluation. Because components operate independently, ablation studies can be conducted by disabling or varying specific modules, for example, testing Storyteller with or without recovery-based grounding, or comparing different pattern generation strategies. Similarly, the system can be extended to new characters or domains by retraining or adapting a single module, rather than redesigning the entire pipeline. The modular design thus supports robustness, extensibility, and principle-based experimentation, making the system particularly well-suited for long-term development and interdisciplinary applications.

Chapter 4

Dataset Construction

4.1 Collection of Scientific Papers

The creation of the dataset begins with the identification of scientific articles that are not only technically relevant, but also accompanied by at least one story written by humans. This criterion is fundamental: a paper is included in the corpus only if there are one or more articles in the public domain (blog posts, news articles, technical explanations) that explicitly describe, summarize, or comment on its scientific contribution. This ensures that each article forms the basis of a matched *article-story* unit, the fundamental structure required for training the Splitter and Storyteller models.

The initial collection process involved scanning selected sources of scientific communication, including research-oriented media platforms, AI-focused blogs, university press releases, technology magazines, and industry-specific news outlets. These sources were chosen for their editorial quality and their tendency to produce explanatory content written by humans and based on peer-reviewed research. Examples include *Wired*, *IEEE Spectrum*, *MIT News*, *EurekAlert*, *TechCrunch*, and numerous Medium-based publications, each of which caters to a different audience and thus supporting persona diversity.

Automated and semi-automated procedures collected candidate stories for each potential scientific article, following hyperlinks, verifying their availability, and extracting the associated textual content. An article was retained only if it met the following conditions:

1. The story made explicit reference to the scientific article;
2. The story was written in English;
3. The story contained sufficient narrative structure (sections, paragraphs, or discursive flow);
4. The article passed artificial intelligence detection checks that ensured the prevalence of human authorship;
5. The link between the paper and the story was substantial rather than superficial.

The application of these criteria led to the creation of a curated collection of 62 research papers spanning three high-level domains: *AI in entertainment and media*, *AI for accessibility and inclusive design*, and *AI for health and medicine*. These articles serve as the factual basis from which the guidelines and character-tailored stories are derived. They exhibit considerable variability in terms of length, structure, and technical density, an essential property for training models intended to generalize heterogeneous scientific communication tasks.

Table 4.1 summarises the main characteristics of the collected papers and their associated human-written stories.

Table 4.1: Summary of collected scientific papers and associated stories.

Category	Count	Examples	Notes
Scientific papers	62	arXiv, ACM DL, IEEE Xplore	Included only if accompanied by at least one human-written explanatory story.
Candidate stories	249	Medium, Wired, MIT News, EurekAlert	Retrieved through link crawling, source curation, and content filtering.
Stories selected	190	Human-authored narratives	Filtered by language, length, structure, factual consistency, and no-AI detection.
Domains	3	Entertainment, Accessibility, Health	Ensures topical diversity across different scientific communication contexts.
Persona coverage	8	Student, Journalist, etc.	Derived from story content and mapped onto eight audience personas.

This dual-source approach, combining scientific articles with authentic stories written by humans, creates a dataset in which factual content and narrative interpretations coexist. Such a structure is essential for the modeling process: the Splitter uses the article to deduce a structured schema, while the Storyteller learns to reproduce the narrative strategies used by human authors when explaining scientific work to a different audience.

4.2 Parsing and Cleaning

Both the scientific articles and the accompanying human-written stories required significant preprocessing before they could be used to train Splitter and Storyteller. Although the two sources differ in format—PDF for the articles and HTML for the stories—each was subjected to a structured pipeline designed to extract clean, logically organized, and machine-readable text. The PDF-to-text processing phase is essential not only for building the dataset but also for the functioning of the web application, where users typically upload articles in PDF format and expect immediate and reliable processing.

4.2.1 Parsing of Scientific Papers (PDF)

The research articles were initially available only as PDF files, a format optimised for visual presentation rather than machine processing and therefore lacking explicit structural markup. All PDFs were processed with Lettria Document Parsing¹, an advanced parsing system that combines OCR, geometric layout analysis, and rule-based post-processing.

For each document, the parser produces a structured Markdown file in which every content block (e.g., section heading, paragraph, list item, table or figure caption) is associated with metadata describing its type, page number, and normalised bounding box

¹<https://www.lettria.com/features/document-parsing>

coordinates. This representation preserves the logical reading order while abstracting away low-level layout details such as multi-column formatting, font changes, and page breaks.

It also normalises common PDF artefacts, including hyphenated line breaks, duplicated headers and footers, and encoding noise. The same parsing pipeline is reused in the web application: when a user uploads a PDF, the backend invokes the parser, obtains the annotated Markdown, and forwards the cleaned text to the Splitter and Storyteller components.

To illustrate how the parsing and cleaning pipeline operates in practice, we provide a concrete example based on the same reference paper used in the previous chapter, *Attention Is All You Need* [23]. This example shows how a raw PDF is transformed into a structured textual representation suitable for downstream processing.

Listing 1 presents a short excerpt of the paper immediately after PDF parsing. At this stage, the text still contains the layout metadata generated by the parser, including block type, page information, and normalized bounding boxes. These annotations allow the system to preserve the logical and spatial organisation of the original article, which is essential for tasks such as section detection, retrieval-based grounding, and alignment during narrative generation.

Listing 2 shows the corresponding cleaned and normalised text after the removal of HTML metadata, layout markers, repeated whitespace, and retrieval-irrelevant blocks (such as headers, footers, and author metadata). This resulting Markdown structure is the form actually consumed by the Splitter and Storyteller. By exposing these two representations side by side, the transformation from PDF layout to model-ready structured text becomes transparent and interpretable.

Listing 1 Excerpt of parsed PDF produced by Docparse, including layout metadata and bounding boxes.

```
<!-- {"split_candidate": 0, "type": "title", "pages": [0],  
      "bboxes": [[0.34443, 0.12428, 0.65402, 0.14726]]} -->  
# Attention Is All You Need  
  
<!-- {"split_candidate": 1, "type": "title", "pages": [0],  
      "bboxes": [[0.46312, 0.42313, 0.53738, 0.43985]]} -->  
## Abstract  
  
<!-- {"split_candidate": 4, "type": "text", "pages": [0],  
      "bboxes": [[0.23359, 0.45270, 0.76742, 0.63235]]} -->  
The dominant sequence transduction models are based on complex recurrent  
or convolutional neural networks that include an encoder and a decoder...
```

Listing 2 Cleaned and normalised Markdown used as input for the Splitter and Storyteller.

```
# Attention Is All You Need  
  
## Abstract  
The dominant sequence transduction models are based on complex recurrent  
or convolutional neural networks that include an encoder and a decoder...
```

4.2.2 Parsing of Human-Written Stories (Web Articles)

The explanatory stories associated with each scientific article were collected as HTML pages from online media (blogs, news sites, institutional press releases, and technology journals). Unlike PDF articles, these documents are already in digital format, but they still require significant preprocessing to extract a clear narrative suitable for modeling.

Starting from a curated list of source URLs (see Table 4.1), pages were downloaded using a lightweight HTTP client. For static pages, a simple `GET` request was sufficient, while dynamically rendered sites (e.g., those loading content via JavaScript) were served via an automated Chromium instance controlled via Selenium. In the latter case, the page was scrolled programmatically to enable lazy loading, and the final DOM tree was captured once all visible content had been rendered..

The raw HTML was then processed with `BeautifulSoup` to retrieve the main body of the article. Non-content elements such as navigation bars, cookie banners, sidebars, comments, and footers were removed by deleting the `<nav>`, `<header>`, `<footer>`, `<aside>`, and `<script>/<style>` blocks, as well as boilerplate text snippets ("share this article", newsletter subscriptions, copyright notices, etc.). When available, semantic tags such as `<article>` or `<main>` were used as anchors; otherwise, the central column was approximated by selecting the longest contiguous sequence of paragraph and heading tags. From this filtered DOM, the pipeline extracted the page title and the sequence of text blocks corresponding to headings and paragraphs.

The extracted text was then normalized to obtain a clean paragraph-level representation. The cleanup functions removed residual HTML entities and URLs, compressed more whitespace, and corrected common artifacts such as broken lines and hyphenation between line breaks. The same normalization rules used for scientific PDFs (Section 4.2.1) were applied here to ensure that articles such as papers also had a consistent text format, with explicit paragraph boundaries and no layout artifacts.

Finally, the collection was filtered and deduplicated at the story level. Extremely short pages (below a certain minimum character length) were discarded, and linguistic checks ensured that only English stories were retained. To remove near-duplicate content, such as syndicated press releases or mirrored blog posts, each story was embedded using the `all-MiniLM-L6-v2` sentence-transformer model, and the embeddings were clustered using a density-based algorithm. Within each cluster, only one representative story was retained, while the others were flagged as duplicates and excluded from training. Finally, a sample of 20 stories was human-checked so that the resulting corpus of cleaned and deduplicated stories would provide high-quality target texts for the Storyteller and reliable narrative supervision for persona-aware generation.

Unified Cleaning and Normalisation Pipeline

After extraction, both articles and data were subjected to the same cleaning and normalization pipeline. This ensured consistency between sources and removed systematic noise. The pipeline includes:

1. **Removal of non-textual or low-level elements**, including figures, captions, equations, and metadata.
2. **Removal of bibliographic or editorial sections**, such as references, acknowledgments, "insights," and promotional content.
3. **Correction of formatting artifacts**, including spurious line breaks, hyphenation, duplicate segments, and Unicode inconsistencies.

4. **Normalization of punctuation and spacing**, for uniform prose suitable for LLM-based processing.
5. **Paragraph-level structuring**, to ensure that both articles and stories are represented as coherent blocks of text.

The final output of each document consists of the cleaned-up continuous text along with its segmentation into paragraphs. This representation satisfies the requirements of the Splitter, providing structurally coherent scientific prose, and the Storyteller, which relies on a clean and retrievable context to produce faithful and persona-adapted narratives.

4.3 Logical Segmentation of Papers

After analysis, each scientific article must be transformed from a linear representation in Markdown into a structured hierarchy of sections and paragraphs. This segmentation phase provides the organizational structure needed by both the Splitter and the Storyteller, allowing them to work on semantically coherent units rather than raw text.

Section Boundary Detection

The Markdown produced by the parsing stage preserves heading markers (e.g. #, ##, ###), which correspond to section, subsection, and subsubsection titles. Logical segmentation begins by scanning the document for these markers and constructing an ordered list of structural nodes. Each node retains:

- the heading text,
- its hierarchical level,
- page-level metadata inherited from the parser,
- the span of paragraphs associated with that section.

This procedure yields a lightweight rhetorical outline of the paper, reflecting the original organisation without reproducing layout artefacts.

Paragraph Segmentation

Within each detected section, the body text is segmented into paragraphs. Paragraph boundaries are determined by blank-line separation in the cleaned Markdown, ensuring that each paragraph corresponds to a coherent discourse unit rather than a visual line break from the PDF. Low-information blocks such as repeated headers, caption text, affiliations, and bibliography entries have already been removed during cleaning, so paragraphs primarily capture meaningful scientific content.

Preservation of Contextual Ordering

The segmentation process preserves the article’s reading order and associates the metadata required for downstream grounding with each paragraph. Paragraphs inherit their position within the section, their page index, and the structural context established by the heading hierarchy. While this information is not directly used by the templates, it allows retrieval mechanisms to prioritize relevant content and ensures that the alignment between article sections and document regions remains faithful.

Final Representation

The output of the segmentation phase is a hierarchical representation composed of:

- a list of sections (each with a title and level),
- the ordered paragraphs belonging to each section,
- optional metadata linking each paragraph to the source PDF.

This structured format serves as the basis for creating training pairs for the Splitter, where high-level reasoning about sections is required, and for retrieval-based embedding in the Storyteller, which operates at the paragraph level.

Canonical Section Inventory

Beyond paragraph-level segmentation, each paper is also mapped onto a canonical set of scientific sections. Starting from the headings extracted in Section 4.2, different variants (e.g. *Overview*, *Previous work*, *Materials and methods*) are normalised into a shared schema. This yields a super-structure of twelve section types that is used throughout the analysis and in the alignment experiments:

1. **Abstract**
2. **Introduction**
3. **Background**
4. **Method**
5. **Experiments / Results**
6. **Discussion**
7. **Related Work**
8. **Limitations**
9. **Conclusion / Future Work**
10. **Acknowledgments**
11. **References**
12. **Appendix**

On the collected corpus of 62 papers, almost all documents contain an *Abstract*, *Introduction*, *Method*, *Experiments / Results*, *Conclusion / Future Work*, and *References*, while sections such as *Background*, *Limitations*, and *Appendix* appear less consistently.

4.4 Story–Paper Alignment

To train models capable of rewriting scientific content for different audiences, it’s necessary to establish a conceptual correspondence between the structure of a scientific article and the narrative produced by human authors. This alignment doesn’t aim to reproduce sentence-level parallels; rather, it seeks to connect large narrative segments of a story with the underlying scientific units from which they derive meaning. The result is a high-level mapping that allows the Splitter and Storyteller to learn how human authors transform technical material into accessible explanations.

Human-written stories rarely mirror the structure of the original article. They mix, reorder, or selectively emphasize concepts depending on their intended audience. Nevertheless, these stories are based on identifiable parts of the article: an introductory paragraph may be related to the abstract, a methodological description to the model section, and a discussion of social implications at the conclusion of the article. Capturing these relationships is essential to:

- provide training cues that connect scientific content to its narrative reinterpretation,
- enable the retrieval of relevant fragments during supervised Storyteller training,
- ensure that human-written explanations remain faithful to the scientific source.

Alignment Strategy

The alignment process begins with the segmented scientific article (Section 4.3) and the cleaned story text obtained from preprocessing. Both documents are represented at the paragraph level. The goal is to map each story paragraph to one or more semantically related paragraphs in the article.

This is achieved conceptually by comparing the discourse units of the two documents. Story paragraphs typically refer to concepts that appear in specific areas of the article, such as the problem statement, methodology, experiments, or results. By identifying these underlying conceptual anchors, the system constructs a set of mappings in the form:

$$\text{story_paragraph}_i \rightarrow \{\text{paper_paragraph}_{j_1}, \dots, \text{paper_paragraph}_{j_k}\}.$$

These mappings are high-level and many-to-many: a single story paragraph may draw from multiple scientific units, and multiple story paragraphs may reference the same section of the article.

Outcome of Alignment

The result of the alignment phase is a structured dataset in which each paragraph of the story is matched to a small set of paper paragraphs that support its content. This representation serves several purposes:

- provides concrete guidance for the Storyteller, allowing it to learn how human writers transform technical paragraphs into narrative prose;
- enables context retrieval during training, limiting the model’s conditioning to the most relevant scientific fragments;
- encourages fidelity by ensuring that each generated section remains anchored to the source material.

Alignment does not attempt to enforce strict textual similarity or sentence-level correspondences. Rather, it captures the conceptual relationship between scientific content and its human-mediated interpretation, thus forming the basis on which human-aware narrative generation can be effectively trained.

Analysis of Story–Paper Alignment Patterns

To better understand how human-written stories reorganise the content of scientific papers, an analysis of the distribution of aligned paper sections across the narrative progression of each story was conducted. Figure 4.1 reports a heatmap in which the y-axis enumerates canonical scientific sections (Abstract, Introduction, Method, etc.), while the x-axis represents the normalised story position (from the beginning to the end, divided into ten equal bins). The colour intensity indicates how often paragraphs aligned to a given paper section appear in a specific portion of the story.

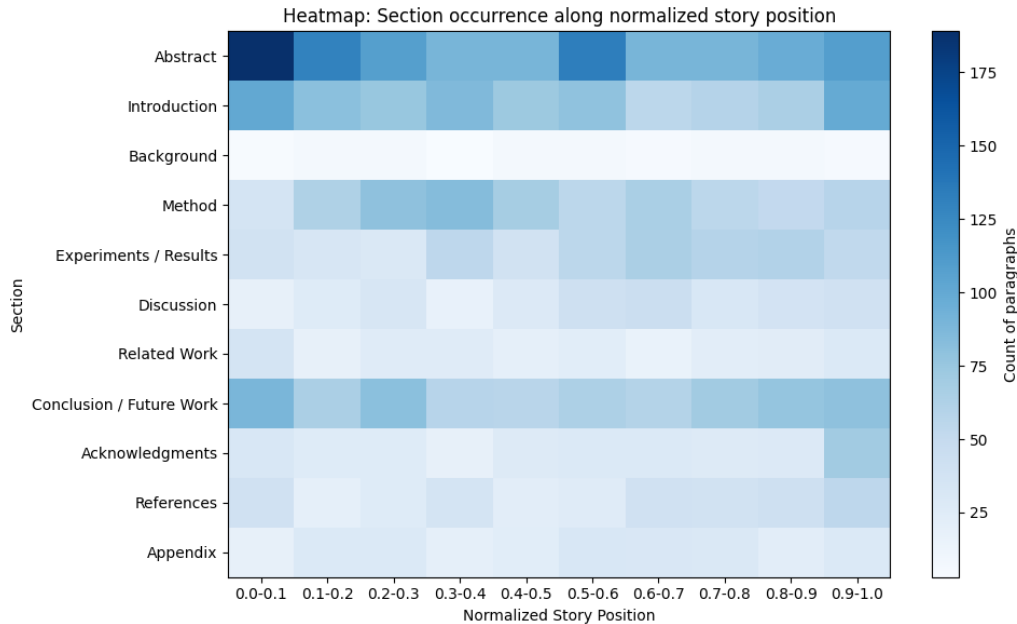


Figure 4.1: Heatmap showing how scientific paper sections are distributed across the narrative progression of aligned human-written stories. Darker regions indicate higher paragraph counts.

The heatmap reveals several systematic patterns:

- **Abstract and Introduction dominate the early story phases (0.0–0.3).** Stories tend to open by summarising and contextualising the research, mirroring the high-level summaries found in the original paper.
- **Background information is rarely included.** Human writers often omit dense contextual or historical sections, likely because the articles are often aimed at an expert audience with a solid background.
- **Method and Experiments/Results appear in the mid-story region (0.3–0.6).** Technical details are introduced only after the reader has been oriented, but before concluding considerations.

- **Conclusion/Future Work is concentrated at the story end (0.6–1.0).** This matches its functional role in scientific discourse. A secondary peak near the beginning suggests that conclusions sometimes overlap conceptually with introductory summaries.
- **Appendix, References, Acknowledgments, and Related Work are almost absent.** These components are rarely included in stories, confirming that human authors strip away academic formalities not relevant to public communication.

Overall, the alignment patterns indicate that human-written stories follow a **consistent narrative flow**:

Abstract/Introduction → Method/Results → Conclusion.

This provides strong evidence that stories preserve the macro-structure of the scientific paper, while selectively reorganising or simplifying content for accessibility. Such consistency supports the design decisions behind the Splitter and Storyteller models, confirming that high-level alignment is both meaningful and learnable.

4.5 Construction of the Splitter Dataset

The Splitter dataset provides the necessary oversight to train a model capable of converting a full scientific article into a concise, persona-specific outline. Its construction was based on two complementary components: (i) mapping human-written stories to the eight audience personas introduced in Chapter 2 and (ii) the controlled generation of additional outline examples to ensure uniform personality coverage across all articles.

4.5.1 Persona Mapping from Human-Written Stories

Before building a complete dataset for the Splitter, it was necessary to determine how the personas defined in Section 2.1.2 should be associated with the human-written stories collected for each scientific article. Because the stories came from diverse sources (science media outlets, university press releases, technical blogs, newspapers, and specialized technology magazines), their narrative tone and communicative intent naturally varied. This variation provided an empirical basis for mapping real stories onto predefined personas.

Each story was first preprocessed and segmented (Section 4.2). A qualitative, rule-based annotation process was then applied to assign a person to each story. The mapping criteria were based on three complementary dimensions:

- **Stylistic Register:** Stories that featured informal language, explanatory metaphors, or motivational framing were associated with personas such as *General Public*, *Student*, or *Journalist*. Conversely, stories that used precise terminology, detailed methodological descriptions, or explicit technical assumptions were associated with personas such as *Researcher* or *Professor*.
- **Editorial Context:** The place of publication often indicated the intended audience. Articles from outlets like *Wired*, *MIT News*, and *IEEE Spectrum* tended to align with the categories of *Journalist*, *Professional*, or *Policy Maker*, while highly technical blog posts or research lab reports were indicative of *Researcher* or *Professor*.

- **Thematic Emphasis:** Articles focusing on societal implications, risks, governance, or high-level impact were associated with *Policy Maker*. Articles focusing on system design or implementation challenges most closely matched *Engineer*. Narratives emphasizing commercial potential, scalability, or applications aligned with *Investor*.

This annotation process produced an initial mapping between stories and characters, creating a set of narrative examples labeled as personas, derived entirely from human-authored content. However, the distribution of stories across the various personas was highly uneven. Some personas (e.g., *Journalist* and *General Public*) appeared frequently in natural sources, while others (e.g., *Investor*, *Policy Maker*, *Engineer*) were only sparsely represented in some articles.

This imbalance posed a methodological challenge: Splitter requires supervision of **each story-person** pair to learn to generate structurally consistent, person-specific patterns. Relying exclusively on human-written stories would have produced a dataset with large gaps, preventing systematic generalization across personas.

4.5.2 Synthetic Outline Generation to Increase Persona Coverage

To overcome this limitation and obtain a complete supervision matrix for all personas, the missing persona-story combinations were filled in using a controlled generation procedure based on the instruction-based **Qwen/Qwen2.5-14B-Instruct** model. For each paper lacking a story for a specific character, the model was primed with the document’s clean text and a character specification and instructed to produce a structure (titles and descriptions) following strict constraints. This approach preserved the strengths of the persona-written examples (stylistic foundation and real-world narrative strategies) while ensuring complete character coverage for all papers.

The final Splitter dataset thus combines:

- **empirically derived persona mappings** from authentic stories, and
- **synthetically generated outlines** that fill the gaps while adhering to a uniform structural format.

This hybrid construction ensures consistency, coverage, and stylistic diversity, providing a robust foundation for training the Splitter to generalise across both scientific content and audience-specific communication practices.

The model **Qwen/Qwen2.5-14B-Instruct** was selected because it demonstrated:

- strong adherence to hard constraints,
- reliable JSON-only output,
- low hallucination tendency under restrictive prompts,
- efficient batch generation suitable for scaling to hundreds of examples.

For each paper, the model received the cleaned full text (Section 4.2), the target persona label, and a strict set of generation constraints:

1. **5 sections**, each with a concise title;
2. **1–2 sentence descriptions** (max ~240 characters);
3. **No invented entities, numbers, datasets, citations, URLs;**

4. **Paraphrased and factual** descriptions grounded in the input;
5. **Persona-specific emphasis**, adjusting tone, focus, and intent.

This procedure served two goals simultaneously: (i) it increased the dataset size and regularity, ensuring that each paper had a balanced set of persona-specific outlines; (ii) it made persona variation explicit and consistent, enabling the Splitter to learn how scientific content is reorganised depending on communicative intent.

4.5.3 Examples of Dataset Entries

The following two outline examples demonstrate how the same paper (*Jukebox: A Generative Model for Music* [6], one of the paper of the dataset) is restructured depending on the persona. The *Investor* version foregrounds controllability and applicability, whereas the *Researcher* version emphasises methodological contributions and evaluation procedures.

```
{
  "id_paper": "01",
  "persona": "Investor",
  "sections": [
    { "title": "Introduction to Jukebox",
      "description": "Overview of Jukebox, a model that generates raw-audio
        ↳ music, addressing long-range dependency modeling challenges." },
    { "title": "Background on Generative Models",
      "description": "Brief history of generative models, focusing on
        ↳ advances enabling high-quality music synthesis." },
    { "title": "Jukebox Architecture",
      "description": "Description of the hierarchical VQ-VAE and
        ↳ autoregressive Transformers used for high-fidelity music
        ↳ generation." },
    { "title": "Controlling Music Generation",
      "description": "Methods for conditioning on artist, genre, lyrics,
        ↳ and timing to steer output style." },
    { "title": "Sampling Methods",
      "description": "Summary of ancestral, windowed, and primed sampling
        ↳ strategies." }
  ]
}
```

```
{
  "id_paper": "01",
  "persona": "Researcher",
  "sections": [
    { "title": "Introduction to Jukebox",
      "description": "Overview of Jukebox, a raw-audio generative model
        ↳ addressing limitations of symbolic music approaches." },
    { "title": "Model Architecture",
      "description": "Details on the hierarchical VQ-VAE, prior models, and
        ↳ autoregressive Transformers." },
    { "title": "Training and Evaluation",

```

```

    "description": "Training procedures including spectral loss, dataset
    ↪ composition, and upsampler design." },
  { "title": "Controlling Generation",
    "description": "Technical explanation of conditioning on artist
    ↪ identity, lyrics, and timing." },
  { "title": "Results and Applications",
    "description": "Empirical performance and qualitative examples across
    ↪ multiple musical styles." }
]
}

```

4.5.4 Dataset Size and Splits

Once completed, the dataset consisted of **496 outline examples**:

$$62 \text{ papers} \times 8 \text{ personas.}$$

After shuffling, it was divided into standard machine-learning splits:

$$396 \text{ train, } 49 \text{ validation, } 51 \text{ test.}$$

Each item represents a supervised instance of the form:

$$\langle \text{paper text, persona} \rangle \longrightarrow \text{persona-specific outline.}$$

This dataset forms the foundation for training the Splitter, enabling it to plan structured, persona-adapted story skeletons from full scientific articles.

4.6 Construction of the Storyteller Dataset

The Storyteller dataset provides supervision for the second stage of the pipeline, in which a complete, persona-adapted narrative is generated from a scientific article, a persona description and a high-level outline. While the Splitter operates at the level of section planning, the Storyteller learns to transform each planned section into coherent prose that resembles human-written stories. The construction of this dataset starts from the cleaned stories and their alignment to paper sections (Section 4.3), and then applies an LLM-assisted refinement step to obtain high-quality section texts suitable for training.

4.6.1 Section-Level Mapping from Stories to Outlines

The starting point is the corpus of human-written stories collected for the 62 papers (Section 4.1), already parsed and cleaned as described in Section 4.2. For each paper–story pair, the story was aligned with the logical structure of the corresponding scientific article and with the five-section outline produced by the Splitter. Concretely, each story was partitioned into five narrative segments, one per outline item, yielding section-level groupings of the form:

$$\langle \text{section_title, paragraphs} \rangle,$$

where **section_title** is the title from the Splitter outline (e.g. “Introduction to AI in Music”) and **paragraphs** is a list of one or more story paragraphs mapped to that section. This mapping uses the underlying story–paper alignments to ensure that each group of

paragraphs corresponds, at least conceptually, to a specific part of the original article (introduction, method, results, implications, etc.). At this stage, however, sections are often noisy: some are too short or too long, others contain boilerplate (e.g. links, disclaimers), or mix loosely related fragments that break narrative continuity.

4.6.2 LLM-Assisted Cleaning and Normalisation

To transform these raw groups of paragraphs into clean supervision signals, each section underwent an LLM-assisted refinement procedure based on Qwen/Qwen2.5-14B-Instruct. Before rewriting, a lightweight evaluation step decided whether a given section required intervention. If a section was already coherent, well-balanced in length, and free of artefacts, it was kept as is; otherwise, it was passed to the model with a targeted instruction specifying the type of refinement needed. Three main edit operations were used:

- **Extension or shortening:** applied when a section was too short, too long, or redundant, so that the final text would better capture the intended meaning of the corresponding outline item.
- **Paraphrasing:** used when the grouped paragraphs were semantically related but lacked continuity (e.g. concatenated fragments from different parts of the story), requiring a single, smoothly flowing passage.
- **Cleaning:** Applied to remove noisy or off-topic content, such as references to arXiv identifiers, promotional text, or links to websites that might introduce unwanted bias during training, which may have escaped the first cleaning step described above.

The model was instructed to produce a single, coherent section faithful to the original story content and stylistically aligned with the inferred persona, while strictly avoiding fabricated details. The following example illustrates the transformation applied to a section about *Jukebox: A Generative Model for Music* [6]:

```
{
  "section_title": "Introduction to AI in Music",
  "paragraphs": [
    "Jukebox, an innovative AI system created by OpenAI, leverages the
    ↪ power of deep learning to generate music, complete with lyrics and
    ↪ vocals, in a variety of genres and styles. By training on a dataset
    ↪ of 1.2 million songs, Jukebox showcases an unparalleled level of
    ↪ sophistication in music generation, pushing the boundaries of what
    ↪ AI can achieve in the creative arts.",
    "Additionally, Jukebox employs a hierarchical transformer architecture
    ↪ to model long-range dependencies in music. This architecture allows
    ↪ the AI system to capture and reproduce the intricate structures and
    ↪ patterns found in real music, such as melodies, harmonies, and
    ↪ rhythm.",
    "The Creative Process with Jukebox: Collaboration between Humans and
    ↪ AI",
```

```

"Jukebox offers a unique and powerful platform for artists, producers,
→ and music enthusiasts to collaborate with AI in the creative
→ process. By providing users with a set of intuitive tools and
→ controls, Jukebox allows for the fine-tuning of generated music to
→ better align with the user's artistic vision. This collaborative
→ approach not only helps users unleash their creativity but also
→ opens up new possibilities for music creation, pushing the
→ boundaries of traditional music production.",
"The introduction of Jukebox heralds a new era in the music industry,
→ with AI-generated music poised to become an integral part of the
→ creative landscape. As Jukebox continues to evolve, the
→ possibilities for artists, producers, and music enthusiasts are
→ virtually limitless.",
"Moreover, Jukebox could pave the way for novel forms of artistic
→ expression and collaboration, as musicians and AI systems learn
→ from and inspire each other. As AI-generated music becomes more
→ widespread, the creative possibilities will continue to expand,
→ ushering in a new age of artistic innovation."
]
}

```

After refinement, the same section is represented as a single, cleaned narrative block:

```

{
  "title": "Introduction to AI in Music",
  "text": "Jukebox, an innovative AI system developed by OpenAI, uses deep
→ learning to generate music, including lyrics and vocals across
→ various genres and styles. Trained on a vast dataset of 1.2 million
→ songs, Jukebox demonstrates remarkable sophistication in music
→ creation, setting new benchmarks for AI-generated music. The system
→ employs a hierarchical VQ-VAE architecture alongside a transformer
→ model to capture complex musical structures like melodies, harmonies,
→ and rhythms. This architecture enables Jukebox to model long-range
→ dependencies in music, ensuring the generated pieces sound natural
→ and coherent. With its ability to produce diverse, high-fidelity
→ songs, Jukebox represents a significant leap forward in AI's role in
→ music generation, promising a future where AI and human creativity
→ intersect in unprecedented ways."
}

```

In total, the corpus contained **4,090** section instances. Of these, **2,462** sections (approximately 60%) were deemed to require LLM refinement, while the remaining 40% were retained in their original, cleaned form. The outcome of this process is a set of uniformly structured, high-quality section texts, each consisting of a title and a coherent passage of narrative prose.

4.6.3 Assembling Input–Output Pairs for Storyteller

Once all section texts had been cleaned and normalised, they were recombined at the story level to create training instances for the Storyteller. For each paper and persona, the five

refined sections were grouped into a single JSON object:

```
output = { "sections":  [{ "title":  ..., "text":  ...  }, ...  ]  },
```

which acts as the target narrative. The corresponding input encapsulates all the information that the Storyteller model will receive at inference time, namely:

- the **persona** specification (e.g. Policy Maker, Student, Investor);
- a brief **system instruction** reminding the model of its role as an AI Scientist Storyteller;
- the **target outline** produced by the Splitter (five section titles and descriptions);
- a slice of **paper context**, consisting of the cleaned scientific article text used to ground generation.

These components are serialised into a single input string, while the target is a strict JSON object containing the five sections. From a dataset perspective, each example can be summarised as:

$\langle \text{paper text, persona, outline} \rangle \longrightarrow \text{JSON with 5 sections (title + text)}.$

This structured supervision allows the Storyteller to learn how to expand persona-specific outlines into full narratives that remain faithful to the scientific article while adapting tone, depth, and focus to the target audience.

4.6.4 Dataset size & Splits

Each refined story is represented as a group of exactly five cleaned sections corresponding to the five-item outline generated by the Splitter. After the refinement procedure, the corpus consists of **4,090** individual sections, which naturally group into:

$$4090/5 = 818 \text{ story-level instances.}$$

These 818 stories form the actual supervision units for the Storyteller. Using the same 80/10/10 partitioning adopted for the Splitter yields:

654 training examples, 82 validation examples, 82 test examples.

All experiments on Storyteller are conducted using these splits.

4.7 Dataset Statistics

This section summarises the quantitative properties of the corpora used to train the Splitter and Storyteller models. The goal is to provide an integrated overview of the amount, structure, and distribution of the collected material, without repeating the procedural details already described in earlier sections.

The final corpus combines scientific papers, human-written narratives, and LLM-refined sections. Table 4.2 reports the global statistics of the dataset. The 62 papers cover three scientific domains and come with at least one aligned narrative, resulting in a corpus where each article has multiple interpretations produced for different audiences. This diversity in narrative style is essential for training persona-adapted generation.

Table 4.2: Overview of the main components of the SciTeller dataset.

Item	Count
Scientific papers	62
Human-written stories (after filtering)	190
Personas represented	8
Outline examples for Splitter	496
Story sections (cleaned & refined)	4 090

Length and Structural Characteristics

The human-written stories exhibit considerable variation:

- typical length ranges from 500 to 1500 words, with extremes above 3000 words;
- most stories contain between 3 and 30 paragraphs;
- cleaned paper texts range from approximately 1,500 to more than 10,000 tokens.

This variability reflects the heterogeneous nature of public scientific communication and provides a realistic training signal for the Storyteller.

Refined Story Sections

The LLM-assisted cleaning pipeline produced:

- **4,090** section candidates grouped from aligned stories;
- **2,462** sections requiring refinement (**60%**);
- **1,628** sections kept after minimal cleaning.

Each final narrative instance contains exactly five refined sections matched to the corresponding Splitter outline.

Final Dataset Splits

Table 4.3: Train/validation/test splits for Splitter and Storyteller datasets.

Dataset	Train	Validation	Test
Splitter	396	49	51
Storyteller	654	82	82

Together, these statistics show that the dataset combines (i) realistic, heterogeneously authored scientific narratives, (ii) structured outlines for eight personas, and (iii) high-quality, LLM-cleaned section texts. This combination provides strong supervision for both structural planning (Splitter) and persona-aware scientific storytelling (Storyteller).

Chapter 5

Experiments

5.1 Baselines

A set of baseline systems was defined to contextualise the performance of the two-stage Splitter–Storyteller architecture. These baselines represent common approaches to scientific text generation with large language models and serve to highlight the specific contributions of structural planning, persona adaptation, and grounding mechanisms introduced in the proposed pipeline.

- **Zero-shot summarisation with general-purpose LLMs**

The first baseline consists of prompting general-purpose instruction-following models in zero-shot mode to produce a persona-oriented summary of a full scientific article. No fine-tuning, structural guidance, or persona-specific supervision is provided. This baseline assesses the inherent tendency of pre-trained models to adapt tone, complexity, and narrative structure without additional training. Although fluent text is typically produced, zero-shot systems often struggle with long-context coherence, frequently omitting central concepts and introducing unsupported statements when required to condense an entire article into a single output.

- **Persona-agnostic summarisation**

A second baseline generates a generic summary that does not incorporate any persona information. This setting approximates a system that captures the general content of a scientific article but does not adjust its framing, terminology, or level of detail to the needs of different audiences. Comparison with persona-aware models reveals whether explicit conditioning leads to measurable improvements in accessibility, emphasis, and narrative alignment with diverse reader profiles.

- **Single-pass long-context generation**

A third baseline involves single-pass story generation, in which the model receives the complete article and is instructed to produce a full narrative in one step. This approach removes the explicit planning phase provided by the Splitter and evaluates the model’s ability to organise long and heterogeneous scientific material autonomously.

In practice, this strategy is limited by long-context degradation, loss of global structure, inconsistencies between sections, and a higher occurrence of hallucinations due to the absence of intermediate grounding signals.

These baselines jointly evaluate complementary dimensions of the task: zero-shot summarisation probes the generalisation capacity of large language models; persona-agnostic summarisation isolates the effect of persona conditioning; and single-pass generation tests robustness and faithfulness under long input without structural scaffolding. Their comparison with the proposed two-stage pipeline enables a clearer attribution of improvements in coherence, controllability, and factual consistency to the architectural choices introduced in this work.

5.2 Configuration and Training Setup

The system is composed of two independently trained modules: a Splitter responsible for producing a structured outline tailored to a target persona, and a Storyteller responsible for generating five narrative sections consistent with that outline. Both modules were trained on datasets specifically constructed for this project, combining human-written stories, refined LLM outputs, and automatically extracted paper segments. This section describes in detail the input and output formats, preprocessing, hyperparameters, hardware configuration, context-management strategies, and validation procedures used throughout training.

Input and Output Formats

Both models operate entirely through instruction-based prompting. The Splitter receives:

1. a persona,
2. the full paper text, and
3. strict formatting rules requiring a JSON output containing a fixed number of sections.

Prompts explicitly forbid speculation, URLs, citations, invented authors, affiliations, or numerical values, and require paraphrasing over extraction.

The Storyteller receives a more complex input:

1. a persona,
2. the five-section outline generated by the Splitter (titles + descriptions),
3. a paper context slice consisting of the most relevant paragraphs for that section.

The output must be a JSON object with exactly five sections reproducing the outline titles verbatim and containing persona-adapted narrative text. The model is required to generate 8–10 sentences per section, avoid hallucinations, and remain faithful to the scientific content.

Preprocessing and Context Management

Scientific papers exceed typical context limits; therefore multiple preprocessing strategies were employed. During dataset creation, papers were segmented into their main components (Abstract, Introduction, Methods, Results/Experiments, Conclusion). Segments exceeding the preprocessing cap (1,024 tokens) were split to stabilise dataset construction,

while the training models operated with significantly larger context windows (4,096 tokens for the Splitter and 8,192 tokens for the Storyteller). During inference, a retrieval procedure was used to extract the top relevant paragraphs for each outline section, reducing distraction and improving grounding. This ensures that the Storyteller processes focused information rather than the full document at once.

Tokenisation followed the standard tokenisers of T5 and Qwen families. Before training, all examples were checked for maximum lengths: approximately 5% (125 items) exceeded the 1,024-token cap and were truncated to comply with model constraints without compromising narrative consistency.

Hyperparameters and Training Configuration

Core experiments were executed on NVIDIA A100 and L40S GPUs. The Qwen2.5–32B Storyteller was fine-tuned on an NVIDIA A100 using BF16 mixed precision together with 4-bit NF4 QLoRA quantisation, while subsequent experiments and deployment were performed on NVIDIA L40S GPUs with 48 GB of memory.

For the Storyteller, QLoRA was applied to attention and feed-forward modules ($r=16$, $\alpha=32$, dropout=0.05). Packing was disabled to guarantee one example per batch, simplifying alignment of labels with outputs. ReduceLROnPlateau was used with factor 0.5, patience 2, and minimum learning rate of $3e-5$.

Loss was computed only on the JSON output to prevent contamination from instructional prefixes. Generation was constrained with deterministic decoding (temperature=0.0) to enforce reproducibility.

Parameter	Splitter (Qwen2.5–7B)	Storyteller (Qwen2.5–32B)
Training method	Full fine-tuning	QLoRA (4-bit NF4)
Max input length	4,096 tokens	8,192 tokens
Max output length	512 tokens (practical limit, not enforced)	2,048 tokens
Effective batch size	8	32 (per-device=1, grad_acc=32)
Learning rate	$2e-4$	$1.5e-4$ (cosine scheduler)
Precision	bf16	bf16
Epochs limit/stopped	10/8	30/13
Early stopping	ROUGE-L + cosine sim	StoryScore + ReduceLROnPlateau
Weight decay	0.00	0.05
Gradient checkpointing	Enabled	Enabled
Training time	~1.5h	~49.5h

Table 5.1: Training hyperparameters for the Splitter and Storyteller.

Validation, Monitoring, and Early Stopping

Validation followed different criteria for the two models. For the Splitter, performance was evaluated using ROUGE-L and SBERT cosine similarity between predicted and target outlines, resulting in a composite metric guiding early stopping (triggered at epoch 8). Outputs were also checked for JSON validity.

For the Storyteller, a dedicated metric named *StoryScore* was used. It combines: (i) JSON validity, (ii) exact title matching, (iii) semantic faithfulness via BERTScore, (iv) lexical recall, (v) absence of repetition (measured via n-gram loops), and (vi) absence of hallucinated entities (capitalised tokens not present in the paper). Training stopped at epoch 8.5, achieving average StoryScore 0.65 and BERTScore 0.806 on the validation set.

This comprehensive setup ensures modularity, structural reliability, and factual consistency, providing a robust foundation for the experiments described in the following sections.

5.3 Experimental Setup

The experimental setup defines the protocol used to evaluate the Splitter-Storyteller pipeline under controlled and reproducible conditions. All experiments were conducted using the held-out test splits defined during dataset construction. The Splitter was evaluated on its 51-example test set, while the Storyteller was evaluated on its 82-example test set, each derived from the 818 persona-specific story instances described in Chapter 4. Together, these splits cover all eight personas and ensure a consistent and fair evaluation across both planning and narrative generation stages. For each test instance, the system generated a complete five-section narrative, enabling evaluation both at the level of individual sections and at the level of the assembled full story. Each persona was tested across the entire test set, resulting in a comprehensive assessment of the system’s ability to adapt tone, terminology, and explanatory depth to different reader profiles.

The experiments follow a two-stage generation workflow. In the first stage, the Splitter receives the full paper text and a target persona and must produce an outline consisting of four to six section titles accompanied by short, factual descriptions. The output is required to be a strictly formatted JSON object with no additional commentary. In the second stage, the Storyteller receives the persona, the outline, and a retrieved subset of the paper content, and must generate exactly five narrative sections that reproduce the outline titles verbatim and expand them into coherent persona-adapted prose. This modular architecture ensures explicit structural control and allows failures to be attributed either to the planning or the storytelling component.

Inference is performed under deterministic decoding conditions unless otherwise specified. Temperature is fixed to 0.0 and nucleus sampling is disabled to guarantee reproducibility and eliminate variance introduced by stochasticity. Additional sampling-based runs were conducted for robustness analysis, using temperature 0.4 and top- p 0.9, but the primary results reported in this work use deterministic decoding. All outputs must comply with strict JSON formatting, and the models are instructed to avoid hallucinations, fabricated entities, URLs, citations, and unverifiable numerical claims. Prompts enforce faithful paraphrasing rather than extraction, ensuring that narrative flow and persona alignment are preserved without compromising factual grounding.

Evaluation is performed at two granularities. At the section level, each of the five generated sections is assessed for structural correctness, adherence to the outline, semantic relevance to the retrieved paper context, and consistency with persona constraints. At the full-story level, the five sections are concatenated to analyse global coherence, narrative continuity, factual alignment with the original scientific paper, and stylistic consistency across the entire output. This dual-level evaluation provides a detailed view of the behaviour of the system and enables the identification of failure modes related to long-context reasoning, persona drift, and cross-section coherence.

Overall, this experimental setup ensures that the system is tested under realistic, reproducible, and well-controlled conditions, capturing both the strengths and limitations of the two-stage architecture across a diverse range of scientific papers and target personas.

5.4 Preliminary Experiments

A sequence of exploratory experiments was conducted to assess the suitability of different large language model architectures for scientific storytelling and to understand the challenges associated with long-context narrative generation, persona adaptation, and structural control. The preliminary phase served as a systematic search across decoder-only and encoder-decoder families to evaluate their capacity for handling the complexity of scientific papers and producing coherent multi-section narratives aligned with diverse personas. Each family of models was tested under realistic conditions, using the token limits, training strategies, and computational constraints described in the previous sections.

The initial experiments focused on the Flan-T5 family [3], testing the *small*, *base*, *large*, and *XL* variants. These models demonstrated strong instruction-following abilities but were fundamentally limited by their restricted 512–1024 token context windows. Chunking strategies were required to process long papers, but this introduced narrative discontinuities and hindered long-range coherence. Outputs often suffered from brevity, incomplete coverage of scientific content, and weak persona conditioning, suggesting that encoder-decoder architectures with short contexts were not suitable for full-length scientific storytelling.

To mitigate these constraints, LongT5 [7] was evaluated as a long-context alternative. Its global-local attention mechanism theoretically supports extended input sequences, making it a strong candidate for document-level tasks. However, training LongT5 under QLoRA [5] 4-bit quantisation proved unstable, yielding degenerate outputs and a complete collapse of evaluation metrics. The model failed to converge even after extensive training, indicating architectural fragility and a mismatch between LongT5’s requirements and the scale of the available supervised dataset.

The next phase of experimentation examined decoder-only architectures, starting with Mistral 7B [11]. These models showed improved fluency, natural phrasing, and stronger persona responsiveness compared to encoder-decoder alternatives. Nevertheless, section-based generation produced short or underdeveloped narratives with weak semantic grounding, while full-story generation led to content drift, factual inconsistencies, and repetition across sections. Even with distributed fine-tuning and improved optimisation stability in Mistral 7B v0.3, structural deviations and inconsistent JSON formatting remained common, limiting the model’s reliability for a tightly controlled storytelling pipeline.

Given these limitations, a unified approach using Qwen2.5–32B [18] was explored, in which a single model was prompted to perform both outline generation and storytelling within one forward pass. This approach produced moderate results, benefiting from the model’s extended context window and strong instruction compliance. However, frequent failures in JSON formatting, hallucinated section titles, misaligned structure, and persona drift indicated significant task interference. The unified model struggled to balance planning and narrative synthesis simultaneously, highlighting the need for architectural separation between these distinct cognitive functions.

Collectively, these preliminary results provided critical insights into the behaviours, strengths, and failure modes of the tested architectures. They revealed that no single-pass or single-model approach was capable of reliably producing structured, grounded, and persona-adapted stories from long scientific papers. The observed issues (context collapse, hallucinations, structural drift, and weak long-range reasoning) motivated the transition to a modular two-stage architecture. The Splitter-Storyteller framework emerged as a principled solution, enabling explicit structural planning and controlled narrative generation, and addressing many of the weaknesses identified during this exploratory phase.

5.4.1 Experiments with Flan-T5

The Flan-T5 family represented the first group of models evaluated for scientific storytelling. Four variants were tested (*small*, *base*, *large*, and *XL*) with the objective of assessing whether a pre-trained encoder-decoder architecture with strong instruction-following capabilities could generate coherent multi-paragraph stories from long scientific papers. These experiments were particularly relevant because Flan-T5 is widely used for text-to-text tasks and provides a clean baseline for understanding the limits of encoder-decoder models under the constraints of long documents, structured outputs, and persona adaptation.

Despite their strengths, all model variants were fundamentally constrained by their limited context windows, ranging from 512 tokens in Flan-T5-Small to 1024 tokens in Flan-T5-Large and Flan-T5-XL. Scientific papers and long narrative stories substantially exceed these limits, making truncation or chunk-based processing unavoidable. When fed truncated inputs, the models frequently ignored large parts of the paper, producing incomplete summaries and narratives lacking essential details. When a chunking strategy was adopted, each chunk was processed independently, which led to fragmented outputs, loss of global coherence, and discontinuities between consecutive sections.

Model Variant	Parameters	Max Input Tokens	Observed Limitations
Flan-T5-Small	60M	512	Severe truncation; minimal persona adaptation
Flan-T5-Base	220M	512–1024	Fragmented content; weak grounding
Flan-T5-Large	770M	1024	Short outputs; incomplete coverage
Flan-T5-XL	3B	1024	Instability; high memory usage; brevity bias

Table 5.2: Flan-T5 variants evaluated and their observed limitations.

A consistent issue across all variants was the strong brevity bias inherent in T5 models. Even when prompted to generate longer narratives, outputs were often significantly shorter than the reference stories, typically producing only a few sentences instead of multi-paragraph narrative text. This was further reinforced by the brevity penalty used by BLEU during evaluation, which highlighted systematic underproduction of content across all model sizes. Attempts to mitigate this behaviour through increased `max_length`, beam search, or adjusted decoding parameters showed limited improvements; the models continued to compress complex scientific sections into brief, generic statements lacking nuance and detail.

Persona conditioning also revealed structural constraints. While Flan-T5 responded consistently to short prompts or isolated instructions, persona information degraded rapidly in long outputs. Models tended to include persona-specific cues in the opening sentences but then reverted to a neutral tone, lacking sustained adaptation of vocabulary, complexity, or narrative focus. This effect was exacerbated in the chunk-based approach, where each chunk was processed independently and thus lacked awareness of previous persona cues or narrative choices.

These challenges led to an initial exploration of section-wise generation as a potential workaround. In this approach, the scientific paper was divided into semantic units (e.g., Abstract, Introduction, Methods, Results, Conclusion), and the model was asked to generate story segments corresponding to each unit. Although this partly alleviated the context-limit problem and improved persona consistency within short segments, it did not solve the global coherence issue. Stories produced through concatenation of independently generated sections exhibited abrupt transitions, redundant explanations, and inconsistencies in style and content focus. Moreover, the absence of a structured outline meant that

Flan-T5 lacked the ability to impose a coherent discourse structure across multiple sections.

To summarize what has been said about Flan-T5, these experiments demonstrated that encoder–decoder architectures with short or moderate context lengths are unsuitable for full-length scientific storytelling. Their limited ability to integrate information across long passages, combined with brevity bias and weak discourse-level persona conditioning, prevented them from producing complete, coherent, and persona-adapted stories. These findings provided the first empirical motivation to explore long-context models and to investigate modular generation strategies capable of preserving global narrative structure.

5.4.2 Experiments with LongT5

LongT5 was evaluated as a potential solution to the context-length limitations observed in the Flan-T5 experiments. Its global–local attention mechanism is designed to process substantially longer input sequences, making it an appealing candidate for document-level tasks such as scientific storytelling. The expectation was that LongT5 would retain the strong instruction-following behaviour of the T5 family while alleviating the fragmentation and truncation issues caused by the short context windows of earlier models. This made LongT5 a natural next step in the exploration of encoder–decoder architectures for this task.

In practice, however, the model proved extremely difficult to fine-tune under realistic computational constraints. Training LongT5 with QLoRA 4-bit quantisation resulted in severe instability throughout optimisation. Despite more than sixty hours of uninterrupted training on NVIDIA A100 hardware, the model failed to converge, exhibiting high and fluctuating loss values across epochs. The generated outputs consisted primarily of repetitive tokens, incomplete phrases, or sequences of non-words, indicating a breakdown in the model’s ability to produce coherent text. The collapse was reflected across all evaluation metrics, which remained near zero for the entire duration of training.

The instability appeared to stem from several interacting factors. First, LongT5’s multi-head global attention component introduces sensitivity to quantisation; applying NF4 4-bit quantisation to these layers likely disrupts the internal representations required for long-context reasoning. Second, the architecture is heavily optimised for large-scale pretraining corpora and may require significantly more supervised data than available in this project to adapt to specialised narrative tasks. Third, the combination of long context, QLoRA low-rank updates, and the relatively small batch sizes imposed by GPU memory limits further contributed to gradient noise and optimisation failure.

Observed Behaviour	Description	Effect on Training
Training divergence	High, unstable loss values	No convergence over 60+ hours
Degenerate outputs	Repetition, broken tokens, incomplete sequences	Incoherent text generation
Quantisation fragility	Sensitivity to NF4 QLoRA	Disrupted attention mechanisms
Data mismatch	Model expects large-scale supervision	Insufficient signal for adaptation

Table 5.3: Failure modes observed in LongT5 experiments.

Therefore, the experiments with LongT5 demonstrated that long-input encoder–decoder architectures can be fragile under low-bit quantisation and may require substantially larger datasets and computational budgets than those available in this project. The complete collapse of the model across all qualitative and quantitative indicators confirmed that LongT5 was not a viable option for scientific storytelling under the constraints of this work. These findings marked a turning point in the experimental phase, motivating a transition toward decoder-only architectures with more stable fine-tuning behaviour and stronger support

for extended context windows.

5.4.3 Experiments with Mistral 7B

Mistral 7B was investigated as the first decoder-only alternative to the encoder-decoder models previously evaluated. Its strong instruction-following capabilities, competitive performance on reasoning tasks, and efficient attention mechanism made it a promising candidate for generating coherent multi-paragraph narratives. Furthermore, its extended context window offered a potential advantage over Flan-T5 and LongT5, enabling experiments that more closely resembled the full scientific storytelling task without aggressive input truncation. The experiments with Mistral 7B aimed to determine whether a decoder-only architecture could maintain narrative coherence, factual grounding, and persona adaptation across long outputs.

The initial experiments followed a section-based approach in which the model was fine-tuned to produce one narrative section at a time from selected paper segments. Although Mistral generated fluent sentences and displayed a clear understanding of narrative tone, the outputs were consistently short and lacked substantive content. Semantic grounding was weak, and the generated paragraphs often summarised only superficial aspects of the paper while omitting essential scientific details. Repetition was also a recurring issue, with the model frequently rephrasing the same idea across multiple sentences, indicating insufficient integration of context and limited ability to develop multi-sentence arguments.

Subsequent experiments explored full-story generation, leveraging the model’s larger context window to process entire papers and produce complete narratives in a single pass. This configuration improved local fluency and reduced abrupt transitions between paragraphs, but significant problems remained. Faithfulness to the source content was low, with the model frequently introducing generic statements unrelated to the paper or omitting key methodological or experimental information. Persona conditioning was inconsistent: although the model occasionally adopted the appropriate tone in the opening sentences, it tended to drift toward a neutral, undifferentiated style as the narrative progressed.

A more advanced attempt was carried out using Mistral 7B v0.3, fine-tuned in a distributed setup with LoRA adapters applied to both attention and feed-forward layers. This configuration improved optimisation stability and enhanced persona conditioning, particularly in the initial sections of each story. The model demonstrated clearer control over vocabulary and tone, producing openings that more closely resembled the target personas. However, the underlying structural issues persisted. Narrative drift remained frequent, repetition continued to degrade content quality, and the model struggled to follow the outline consistently, sometimes merging sections or deviating from the prescribed structure.

Observed Behaviour	Description	Impact on Story Quality
Short outputs	Underdeveloped paragraphs	Insufficient content coverage
Weak grounding	Limited alignment with scientific text	Loss of factual detail
Repetition	Recurring n-gram loops	Reduced narrative diversity
Persona drift	Tone not maintained across sections	Inconsistent adaptation
Structural deviations	Ignored or altered section structure	Unreliable story format
Improved fluency (v0.3)	Smoother local transitions	Faithfulness issues remain

Table 5.4: Summary of behaviours observed in Mistral 7B experiments.

Finally to sum up, experiments with Mistral 7B experiments demonstrated that decoder-only architectures provide strong local fluency and stylistic flexibility but struggle with

long-range coherence, faithful summarization of scientific content, and adherence to persona constraints when trained on limited supervised datasets. Although the distributed fine-tuning using Mistral 7B v0.3 offered incremental improvements, the model’s tendency toward repetition, content drift, and structural inconsistency made it unsuitable as a standalone solution. These findings began to raise the idea of the need for a modular architecture that could explicitly separate structural planning from narrative generation.

5.4.4 Experiments with Qwen 32B (Unified Model)

Qwen2.5–32B was evaluated as a unified model designed to perform both outline generation and narrative storytelling within a single forward pass. The motivation for this experiment was to explore whether a sufficiently large decoder-only model with extended context capacity could internalise both the structural planning and narrative synthesis components of the task, eliminating the need for a modular architecture. The unified approach offered an appealing simplification: one model, one inference step, and a single training objective covering structure, content selection, persona adaptation, and long-form generation.

During experimentation, the unified model demonstrated moderate but inconsistent performance. It was capable of producing stories with fluent local phrasing and, in some cases, plausible outlines that resembled the target structure. However, the model struggled to maintain structural alignment between the outline and the generated narrative. Section titles were often invented, omitted, or replaced by generic headings unrelated to the paper. Several outputs contained hallucinated section names or topic shifts that did not appear in the original document, indicating unreliable internal planning. JSON validity also proved problematic: although the model frequently attempted to produce the correct schema, outputs often contained extraneous text, missing fields, nested structures, or formatting inconsistencies that made post-processing difficult.

Narrative grounding exhibited similar instability. While the model occasionally produced segments that were well aligned with the paper, it frequently introduced unsupported claims, speculative content, or factually incorrect details. Persona conditioning suffered from drift across sections: although the opening sentences sometimes matched the target audience, the tone rapidly degraded into a neutral style with inconsistent vocabulary and depth, (as Mistral). Semantic alignment across long outputs remained weak, and the narrative often shifted focus mid-story, reflecting insufficient separation between high-level planning and sentence-level generation.

Observed Issue	Description	Consequence
Section misalignment	Invented, missing, or altered titles	Loss of structural reliability
JSON invalidity	Malformed structures, missing keys	Outputs unusable without repair
Hallucinations	Unsupported claims and invented details	Reduced factual grounding
Persona drift	Inconsistent tone across sections	Weak persona conditioning
Task interference	Planning and storytelling conflated	Lower coherence and faithfulness

Table 5.5: Failure modes observed in the unified Qwen2.5–32B experiments.

The unified approach produced a composite evaluation score (*StoryScore*) of approximately 0.41, reflecting this mixture of partial successes and structural failures. Crucially, analysis of the outputs suggested that the two task components—outline generation and narrative storytelling—compete for attention during decoding. The model must simultaneously infer document structure, decide on content selection, adapt its tone to a persona, and produce multi-paragraph text faithful to the paper. This cognitive overload manifests as task interference, as improvements in one dimension (e.g., fluency) often degrade

another (e.g., structure, grounding), and long outputs amplify drift and inconsistencies.

These observations provided clear empirical evidence that a single-model, single-pass solution is not sufficiently robust for scientific storytelling. The inability of the unified model to maintain a stable structure, produce valid JSON, and adhere to the outline motivated a decisive shift toward a modular architecture. Separating the planning and storytelling tasks into two independently trained models emerged as a principled solution, reducing task interference and enabling explicit control over document structure and persona adaptation. This insight laid the foundation for the two-stage pipeline adopted in the final system.

5.4.5 Emergence of the Two-Stage Pipeline

The transition to a two-stage architecture emerged directly from the empirical observations collected throughout the preliminary experiments, exactly as described in these previous sub-chapters. These observations collectively indicated that a single model attempting to perform both planning and storytelling was being asked to solve two fundamentally different tasks. Outline generation requires abstraction, global reasoning, content selection, and structural organisation. Storytelling, by contrast, requires controlling tone, expanding content into multi-paragraph narratives, maintaining factual grounding, and following constraints imposed by persona and outline. Combining these tasks forces the model to alternate between high-level planning and low-level realisation within the same decoding stream, amplifying errors and increasing the likelihood of drift, hallucinations, and structural failure.

This insight motivated the design of a modular two-stage pipeline in which the Splitter and Storyteller are trained as independent models with clearly separated responsibilities. The Splitter focuses exclusively on producing a stable, human-interpretable outline that captures the essential structure of the paper and adapts it to the target persona. This explicit planning phase provides a controllable scaffold for the narrative and eliminates the need for the second model to infer document structure on its own. The Storyteller then operates under well-defined constraints: it must remain faithful to the paper content through retrieval-supported inputs, reproduce the outline titles verbatim, and expand each section into a coherent persona-adapted narrative. By decoupling the tasks, the system reduces cognitive load on the model and prevents the interference observed in unified configurations.

Early tests of the two-stage approach demonstrated immediate improvements compared to all prior experiments. Generated stories exhibited clearer structure, with sections consistently aligned to the outline and significantly fewer hallucinated titles or content shifts. Retrieval-supported inputs ensured that the Storyteller grounded its narratives in relevant portions of the paper rather than relying on global context alone, leading to better factual alignment and reduced hallucination rates. Persona adaptation also improved: with structure stabilised by the Splitter, the Storyteller was able to maintain tone and vocabulary more consistently across sections. JSON validity increased substantially, reflecting the benefits of having a dedicated model responsible for structured generation.

Overall, the emergence of the two-stage pipeline marked a decisive turning point in the development of the system. The modular design addressed the core failure modes observed in unified and single-pass models, offered greater controllability and interpretability, and provided a scalable foundation for subsequent fine-tuning and evaluation. These early successes confirmed that separating planning and storytelling was not only beneficial but essential for achieving reliable scientific storytelling under realistic constraints.

5.5 LLM Selection

The selection of the final models was driven by empirical evidence gathered throughout the exploratory experiments. Each model family tested during the preliminary phase exposed specific strengths and limitations, and the cumulative observations indicated that no single architecture could reliably perform both structural planning and narrative generation. As a result, two separate models were selected, each optimally suited to one of the two tasks in the pipeline. Qwen2.5-7B-Instruct was chosen as the Splitter due to its strong instruction-following behaviour, reliable JSON formatting, and consistent performance in producing coherent section titles and descriptions. Qwen2.5-32B-Instruct was selected as the Storyteller for its superior long-context reasoning, improved semantic alignment with scientific papers, and increased ability to sustain persona conditioning across multi-paragraph outputs.

The decision to adopt Qwen2.5-7B for the Splitter was informed by comparative analyses with other models evaluated earlier. Encoder-decoder models, despite their strong instruction-following capabilities, were fundamentally constrained by limited context lengths and demonstrated poor reliability in producing multi-section and JSON structures. Decoder-only alternatives such as Mistral 7B displayed high fluency but frequently deviated from the requested structure, introduced hallucinated sections, or returned malformed JSON outputs. In contrast, Qwen2.5-7B consistently produced well-formed outlines with accurate title-description pairs and showed strong adherence to constraints. Validation using ROUGE-L and SBERT cosine similarity confirmed that Qwen2.5-7B provided the highest structural accuracy among models tested in this role.

Criterion	Flan-T5 (S/B/L/XL)	LongT5
Context	Very limited (512–1024)	Long (theoretical)
Structure	Weak	Unstable
Persona	Weak	Weak
Hallucinations	Moderate	Severe under QLoRA
JSON	Moderate	Invalid
Task Suitability	Unsuitable	Unsuitable

Table 5.6: Comparison of encoder-decoder LLMs (Flan-T5 and LongT5) across key criteria relevant to scientific storytelling.

Criterion	Mistral 7B	Qwen 32B (Unified)	Qwen2.5-7B (Splitter)	Qwen2.5-32B (Storyteller)
Context	Medium	Long	Long	Long
Structure	Inconsistent	Poor (section drift)	Strong	Strong
Persona	Partial	Inconsistent	Strong	Strong
Hallucinations	Moderate	Moderate-High	Low	Low
JSON	Inconsistent	Frequent errors	Reliable	Reliable
Task Suitability	Partially suitable	Not suitable	Best for structure	Best for storytelling

Table 5.7: Comparison of decoder-only LLMs across key criteria relevant to scientific storytelling.

For the Storyteller, the choice of Qwen2.5-32B emerged from the need for a model capable of sustained long-context reasoning, faithful paraphrasing, and persona-adapted narrative generation. Smaller models lacked the capacity to maintain semantic alignment across entire stories, and single-pass generation with unified architectures produced inconsistent personas, hallucinated details, and structural drift. Mistral 7B exhibited strong local fluency but failed to handle extended context and often repeated or omitted key con-

tent. Experiments with unified models showed that combining planning and storytelling in one model led to task interference and reduced reliability. When fine-tuned exclusively as a storyteller, however, Qwen2.5-32B demonstrated significant improvements in faithfulness, JSON validity, and cross-section consistency. Early tests showed reduced hallucinations, greater lexical richness, and more stable persona adherence across the five narrative sections.

Several criteria guided the final model selection. Long-context handling was essential for grounding narratives in scientific papers without excessive truncation or reliance solely on retrieved excerpts. JSON reliability was required to maintain a deterministic pipeline and minimise post-processing complexity. Low hallucination rates were critical to ensuring factual faithfulness. Instruction-following strength was necessary for adhering to strict persona prompts and outline constraints. Finally, consistency across sections was a key indicator of the system’s ability to maintain tone, structure, and narrative continuity across long outputs. Qwen2.5-7B and Qwen2.5-32B jointly satisfied these requirements more effectively than all other models tested.

In summary, the selection of Qwen models for the Splitter and Storyteller was the result of a systematic empirical process. The limitations observed in earlier models highlighted the importance of modularity, strong constraint adherence, and long-context capabilities. The Qwen2.5 family provided the best balance of controllability, structural fidelity, narrative quality, and persona adaptation, making it the most suitable choice for the final two-stage architecture.

5.6 Summary and Lessons Learned

Throughout the development of the system, several consistent failure modes emerged, each shaping the final modular design of the Splitter and Storyteller models. One of the earliest insights was that persona adaptation does not reliably emerge from zero-shot prompting. Across all tested models, persona markers were typically respected only in the opening sentences, with the narrative progressively drifting toward neutral phrasing. This demonstrated that persona control is a *long-range objective* requiring explicit supervision. As a result, persona-specific sections were incorporated into the dataset, and the Storyteller was fine-tuned to maintain tone and vocabulary consistently across multiple paragraphs.

Attempts to generate entire stories in a single pass revealed fundamental limitations in long-form organisation. Models routinely invented section titles, merged unrelated concepts, or lost the narrative thread when processing full papers. These behaviours showed that high-level planning and low-level narrative generation correspond to different reasoning processes, motivating the introduction of the Splitter and the Storyteller.

Feeding entire papers to long-context models demonstrated another failure mode: increasing the context window does not solve grounding issues. Even models with extended attention ranges hallucinated details or ignored important content when presented with excessively broad input. This indicated that grounding depends not on *quantity* of context but on *relevance*. Consequently, the final system employs retrieval-based context.

A recurring weakness among unified and decoder-only baselines was their difficulty in following strict JSON schemas. Malformed or incomplete outputs hindered pipeline stability, underscoring that structural constrainability must be a key criterion for model selection. This requirement contributed to the decision to adopt the Qwen family, whose formatting reliability and adherence to constraints consistently exceeded that of alternative models.

Traditional text-generation metrics such as ROUGE or BLEU proved inadequate for

evaluating scientific storytelling. Often, outputs with high scores frequently contained structural errors, hallucinations, repetitions, or invalid JSON. These limitations motivated the development of *StoryScore*, a composite metric aligned with the practical failure modes observed during experimentation, better explained in the following chapter.

Across all preliminary experiments, the clearest lesson was that no single model can reliably solve planning and storytelling simultaneously. Even 32B-scale unified models with long contexts exhibited complications. The final conclusion is that **modularity is not merely an architectural preference but a necessity**: separating the Splitter and Storyteller yields a system that is more interpretable, controllable, and robust.

Taken together, these insights justify the two-stage architecture adopted in the final system. Qwen2.5-7B provides reliable and constraint-following outline generation, while Qwen2.5-32B delivers coherent, grounded, persona-consistent narratives. These findings establish the foundation for the evaluation presented in the next chapter.

Chapter 6

Evaluation

6.1 Evaluation Metrics

In order to evaluate the quality of the generated stories, this thesis adopts a set of complementary metrics that jointly assess semantic faithfulness to the source article, structural alignment with the outline produced by the Splitter, fluency and coherence of the narrative, and persona-dependent adaptation. These measures reflect the core requirements of scientific storytelling: factual grounding, communicative clarity, structural consistency, and adaptability across audiences.

StoryScore (Composite Metric). StoryScore is the main evaluation metric used throughout this project. It aggregates semantic similarity, lexical grounding, structural fidelity, fluency, and hallucination control into a single score in $[0, 1]$. Following extensive experimentation, including a multi-stage analysis of hallucination detection explained later, the final version of StoryScore is:

$$\begin{aligned} \text{StoryScore} = & (0.3 \text{ ContextRecall}) + \\ & (0.2 \text{ BERTScore}) + \\ & (0.2 \text{ PromptCleanliness}) + \\ & (0.1 \text{ TitleCoverage}) + \\ & (0.1 \text{ NoRedundancy}) + \\ & (0.1 \text{ NoHallucination}). \end{aligned} \tag{6.1}$$

This formulation captures the most stable and reliable indicators, while maintaining a balanced representation of narrative quality dimensions.

Context recall / Article coverage. Context recall quantifies the amount of content from the original article that is reflected in the generated story, i.e., the proportion of tokens from the article that also appear in the narrative (article-centered coverage). Unlike metrics based solely on lexical overlap, contextual recall is used here as a proxy for content coverage. Although calculated at the token level, contextual recall emphasizes coverage: it measures whether the narrative includes a sufficient proportion of the salient concepts, terminology, and tokens present in the article. Formally, it is defined as:

$$\text{ContextRecall} = \frac{|T_{\text{story}} \cap T_{\text{paper}}|}{|T_{\text{paper}}|}, \tag{6.2}$$

where T_{paper} and T_{story} denote the sets of tokens extracted from the paper and the generated story. Higher values indicate stronger grounding in the source material, reflecting a narrative that retains a substantial portion of the paper’s vocabulary and key content elements.

BERTScore (Semantic Faithfulness). *BERTScore* [25] measures semantic faithfulness by comparing contextual embeddings of the generated story with those of the source article. Instead of relying on surface lexical overlap, BERTScore aligns tokens from the hypothesis to the most semantically similar tokens in the reference using cosine similarity in the embedding space. For each token, precision and recall are computed by aggregating the highest-scoring matches, and the final score is obtained via the harmonic mean of these two signals. Formally, let H and R denote the sets of contextual embeddings for the hypothesis (story) and reference (paper); BERTScore is defined as:

$$\text{BERTScore} = F_1(\text{sim}(H, R), \text{sim}(R, H)), \quad (6.3)$$

where $\text{sim}(A, B)$ denotes the cosine-similarity-based alignment between tokens in A and their best matches in B . Here, F_1 denotes the harmonic mean of the precision and recall components, ensuring that the score is high only when both semantic coverage and semantic faithfulness are simultaneously satisfied. Higher values indicate stronger semantic grounding and better preservation of meaning, which is particularly important for technical personas and sections requiring precise conceptual fidelity.

PromptCleanliness (Instruction Leakage Control). PromptCleanliness measures the extent to which the generated story is free of prompt-related artifacts and instruction leaks, such as residual system directives, role indicators, formatting constraints, or meta-level indications. These artifacts indicate a failure of narrative abstraction, where the model collapses from storytelling to instruction-following behavior, severely degrading readability and audience engagement. This metric was added following qualitative analysis, explained later, where it was noticed that some outputs contained pieces of instructions in the final output.

The metric is implemented as a structural contamination detector rather than a keyword-based filter. The generated text is analyzed line by line and sentence by sentence using a set of high-precision regular expression patterns anchored to sentence or line boundaries. These patterns capture strong indicators of prompt leakage, including role prefixes in chat, (often spotted in final outputs) blocks of imperative instructions (e.g., repeated occurrences of “do not”), echoed JSON structures, and markdown code fences. Each detected pattern contributes to a contamination score proportional to its severity and frequency.

Formally, let C denote the contamination score that quantifies the degree of prompt leakage present in the generated story. It is defined as a normalized weighted sum of explicit structural indicators of instruction-level artifacts. Let L denote the set of non-empty lines in the generated text. The contamination score is computed as:

$$C = \frac{1.0 \cdot N_{\text{line}} + 0.75 \cdot N_{\text{sent}} + 1.25 \cdot N_{\text{json}} + 0.75 \cdot N_{\text{fence}} + 2.5 \cdot N_{\text{block}}}{|L|} \quad (6.4)$$

where:

- N_{line} is the number of lines beginning with explicit chat-role or instruction markers (e.g., `Human:`, `Assistant:`, `Rules:`);

- N_{sent} counts sentences that exhibit imperative instruction patterns at sentence boundaries;
- N_{json} denotes the number of lines consisting solely of structured JSON-like content;
- N_{fence} counts occurrences of markdown code fences;
- N_{block} counts dense instruction blocks characterized by multiple repeated occurrences of imperative constraints (e.g., three or more instances of “do not” within a single sentence or paragraph).

The weights associated with each term were selected empirically to prioritize high-precision detection of severe leakage patterns over recall, assigning stronger penalties to artifacts that indicate a near-complete collapse into instruction-following mode. Normalization by $|L|$ ensures that C reflects the density of contamination rather than the absolute length of the text. The score is clipped to the unit interval, as outputs exceeding this threshold are considered equally dominated by prompt artifacts. PromptCleanliness is then defined as:

$$\text{PromptCleanliness} = 1 - \min(1, C). \quad (6.5)$$

By explicitly penalizing prompt leakage, this metric enforces a strict separation between internal model control signals and the final user-facing story, which is essential for producing fluent, persona-appropriate scientific narratives.

Title Coverage / Outline Fidelity. This metric evaluates whether the Storyteller preserves the section structure proposed by the Splitter, ensuring that the generated narrative adheres to the planned outline. Let $\mathcal{O} = \{o_1, \dots, o_5\}$ denote the target section titles and $\mathcal{G} = \{g_1, \dots, g_5\}$ the corresponding generated titles. It therefore measures the fidelity of the Storyteller to the structural plan produced by the Splitter. Title Coverage is defined as:

$$\text{TitleCoverage} = \begin{cases} 1 & \text{if } \forall i, \text{norm}(g_i) = \text{norm}(o_i), \\ 0 & \text{otherwise,} \end{cases} \quad (6.6)$$

where $\text{norm}(\cdot)$ denotes a normalization function that removes differences in case, whitespace, and punctuation before comparison. A full score indicates perfect structural fidelity to the outline and correct section alignment without hallucinated outlines.

No Redundancy (Fluency Indicator). This metric penalizes degenerate loops and excessive reuse of the same textual fragments, which are common artifacts in long-form generation. This metric was introduced following the numerous results that emerged from the experiments previously explained, in which each section always repeated the same concepts, thus making a global summary of the paper for each section. Repetition is quantified by computing the frequency of n -grams in the generated story. Let $\mathcal{G}_n$ denote the multiset of all n -grams extracted from the narrative; the repetition rate is defined as:

$$\text{RedundancyRate} = \frac{\max_{g \in \mathcal{G}_n} \text{freq}(g)}{|\mathcal{G}_n|}, \quad \text{NoRedundancy} = 1 - \text{RedundancyRate}. \quad (6.7)$$

High values of *NoRedundancy* indicate fluent, varied text with minimal looping or redundant phrasing.

Hallucination Indicators. This metric captures the extent to which the generated story introduces entities that are not supported by the scientific article. In the current system, hallucination detection is implemented through a NER-based comparison: the Storyteller’s output is analyzed using SpaCy [8] to extract PERSON and ORG entities, which are then matched against the entities detected in the paper. Any entity appearing in the story but absent from the article is treated as a hallucination. A broader discussion of alternative hallucination detectors, along with their limitations and the decision of why this is the approach used, is provided in the dedicated hallucination analysis subchapter (Section 6.4). Formally, let *GeneratedEntities* be the set of PERSON/ORG entities extracted from the story, and *HallucinatedEntities* the subset of those not found in the paper; the score is defined as:

$$NoHallucination = 1 - \frac{|HallucinatedEntities|}{|GeneratedEntities|}. \quad (6.8)$$

A higher value indicates stronger factual alignment with the source document.

Metric	Type	Objective	Range
Context Recall	Lexical	Proportion of paper tokens covered by the story	[0, 1]
BERTScore	Semantic	Faithfulness of the story to the original content	[0, 1]
Prompt Cleanliness	Structural	Absence of instruction leakage and prompt-related artifacts	[0, 1]
Title Coverage	Structural	Similarity between generated titles and the Splitter outline	[0, 1]
NoRedundancy	Fluency	Avoidance of repeated n -gram loops and redundant phrasing patterns	[0, 1]
NoHallucination	Factuality	Entity consistency (SpaCy PERSON/ORG) with the paper	[0, 1]
StoryScore	Composite	Global weighted quality score	[0, 1]

Table 6.1: Summary of evaluation metrics used in this thesis.

Rationale for Metric Selection. These metrics were selected based on three requirements: (i) sensitivity to semantic and factual grounding in the source article, (ii) robustness to paraphrasing and persona variation, and (iii) ability to capture structural, narrative, and stylistic correctness. Together, they provide a reliable and multidimensional evaluation framework for persona-adaptive scientific storytelling.

6.2 Quantitative Results

This section presents the quantitative evaluation of the system and its report

1. The performance of the final Splitter–Storyteller pipeline,
2. The results obtained from early baseline experiments (Flan-T5, Mistral, LongT5),
3. The progression across models leading to the final architecture.

These results are based on experiments conducted during the project, explained in the previous chapter.

6.2.1 Performance of the Final Splitter–Storyteller Pipeline

The final system was evaluated on a held-out test set of 82 papers. Table 6.2 reports the scores for the main evaluation metrics introduced in Section 6.1.

Metric	Mean Score
StoryScore	0.7873
Context Recall	0.4725
BERTScore	0.8150
Prompt Cleanliness	1.0000
Title Coverage	0.9982
No Redundancy	0.9025
No Hallucination	0.9248

Table 6.2: Quantitative results of the final Splitter–Storyteller pipeline on the test set.

The system achieves strong semantic coherence ($BERTScore = 0.8150$), while maintaining good structural robustness, with more than 99% of outputs matching the Splitter outline exactly. Fluency is particularly high ($No\ Redundancy = 0.9025$) as the $No\ Hallucination$ score (0.9248). Grounding is moderate, reflecting the system’s design to avoid extractive summarisation. The perfect PromptCleanliness score reflects the fact that the final system was evaluated after stabilizing prompt templates and eliminating instruction leakage during development.

6.2.2 Results of the Baseline and Intermediate Models

Before converging on the final architecture, several baseline models were evaluated. Only experiments producing non-empty and interpretable outputs are reported. For early models such as Flan-T5, outputs consisted of extremely short summaries (1–2 sentences), making metrics like BERTScore or story-structure measures non-informative. The following table therefore reports only ROUGE and BLEU, the only meaningful metrics given the nature of the outputs.

Model	ROUGE-1 F	ROUGE-2 F	ROUGE-L F	BLEU	BERTScore F1
Flan-T5-base	0.230	0.047	0.078	0.34	—
Flan-T5-xl	0.1239	0.0246	0.0817	0.0039	—
Flan-T5-large (best)	0.1967	0.0349	0.1262	1.90	−0.189

Table 6.3: Quantitative results for the main Flan-T5 experiments. Missing values indicate metrics that were not meaningful due to extremely short outputs.

Interpretation. Despite increasing model size, the Flan-T5 family did not produce structured narratives. Outputs were often one or two sentences long, resulting in:

- ROUGE improvements too small to matter,
- BLEU scores dominated by short-output bias,
- BERTScore either undefined or misleading (comparing a full story with a single sentence).

These models demonstrated that instruction-following LLMs cannot handle multi-section storytelling without explicit architectural support.

6.2.3 Progression Across Models

To illustrate the semantic improvement obtained throughout the project, Figure 6.1 reports the progression of *BERTScore F1* across the only models for which this metric is meaningful: the non-finetuned Flan-T5-Large baseline, the finetuned Mistral 7B model, the unified Qwen-32B model, and the final two-stage Splitter-Storyteller system. Earlier Flan models (base, xl) are omitted because they produced extremely short outputs (one or two sentences), making semantic similarity metrics unreliable.

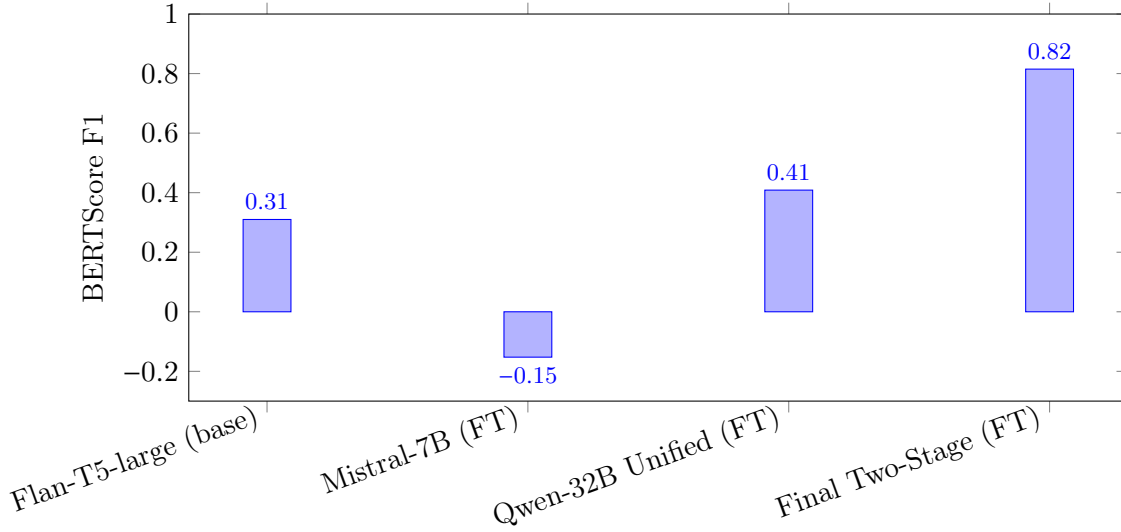


Figure 6.1: Progression of BERTScore F1 across models capable of generating multi-sentence outputs. Flan-T5-large baseline achieves 0.31; Mistral drops to negative alignment; Qwen unified reaches 0.41; and the two-stage system achieves 0.815.

Interpretation.

- **Flan-T5-Large baseline** achieves a moderate BERTScore (0.31). Although the model generates only short summaries, the content is loosely related to the paper, leading to non-trivial semantic similarity.
- **Mistral-7B finetuned** performs significantly worse (-0.1524). This negative score indicates severe semantic misalignment: the model often produces generic or off-topic sentences unrelated to the scientific article.
- **Qwen-32B unified** reaches a substantially higher score (0.41), confirming that larger pretrained models begin to capture the paper’s semantic space, although they still lack consistent structure and persona control.
- **Final two-stage Splitter-Storyteller** achieves a dramatic improvement (0.815), becoming the first configuration capable of producing long, coherent, multi-section narratives with high semantic fidelity. This validates the architectural intuition that

separating high-level planning (Splitter) from persona-conditioned storytelling (Storyteller) is essential for controlled scientific narrative generation.

Compared to ROUGE, which is poorly aligned with long-form storytelling, BERTScore provides a far more informative measure of semantic quality. These results highlight that architectural design—not model size alone—is the primary factor driving performance improvements in this task.

6.3 Hallucination Evaluation

A recurring challenge in this project was determining whether this project’s narratives remained faithful to the scientific articles they were based on, without invent new entities and facts (hallucination). Unlike summarization models, this project must simplify, contextualize, narrate and adapt explanations to different personas. As a result, some creative divergence is not only acceptable but also desirable. The central difficulty, therefore, is to distinguish *intended pedagogical creativity* from *true factual hallucination*. Over the course of the project, seven increasingly sophisticated hallucination detectors were explored. Each revealed different strengths and limitations, ultimately shaping the final evaluation design.

Before detailing each method, Figure 6.2 provides a visual overview of the evolution of hallucination detection throughout the project.

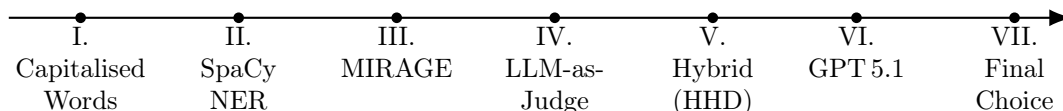


Figure 6.2: Timeline of hallucination detection methods explored throughout the project.

This evolution reflects an increasing desire for both precision and interpretability, balanced against the constraints of persona-driven storytelling.

6.3.1 Capitalised Words as Entity Proxies

The earliest approach was deliberately simplistic: any word starting with a capital letter was treated as a potential factual entity. If such a token appeared in the story but not in the article, it was marked as hallucinated.

This heuristic provided a transparent mechanism for catching blatantly invented institutions or proper names and served as an initial sanity check during early system debugging.

Yet it quickly became apparent that its rigidity clashed with persona-adapted storytelling:

- abbreviations (e.g. “AI”) were flagged if the paper only contained the expanded form (e.g. “Artificial Intelligence”),
- metaphors or narrative constructs capitalised for emphasis were misclassified as entities,
- minor morphological variants (pluralisation, genitives) produced false positives.

Although rudimentary, this method laid the foundation for understanding the structure of the problem: hallucination detection must separate surface-form noise from true semantic divergence.

6.3.2 NER-Based Detection (SpaCy PERSON/ORG)

To reduce noise, the next iteration used SpaCy’s PERSON and ORG recognisers to extract only “true” named entities. This significantly reduced the number of irrelevant candidates and detected genuinely invented organisations or people introduced by the model.

After incorporating regex normalisation (e.g. lowercasing, removal of possessive suffixes), many false positives were corrected. Nevertheless, NER remained a shallow signal: it could detect incorrect affiliations, but failed to identify deeper factual inconsistencies (e.g. wrong scientific claims, invented datasets, or unsupported causal statements). NER was therefore retained as a component, but clearly insufficient on its own.

6.3.3 MIRAGE Rewrite-Consistency Scoring

MIRAGE [24] attempts to detect hallucinations by rewriting the same passage multiple times and measuring the stability of the concepts that appear. If an idea disappears or mutates across rewrites, MIRAGE treats it as hallucinated.

In summarisation, this is often effective. In persona-oriented storytelling, it was not. Experimentally, MIRAGE consistently and incorrectly flagged:

- explanatory metaphors,
- didactic analogies,
- rephrasings intended for non-expert audiences.

This revealed a profound discrepancy between the assumptions of MIRAGE (strict fidelity, literal foundation) and the ambition of a narrative system aimed at simplifying scientific ideas.

6.3.4 LLM-as-a-Judge (Qwen-7B)

A more flexible idea was to use an LLM (Qwen-7B) to judge hallucinations directly. The judge received:

- the full scientific article as **CONTEXT**,
- the story as **ANSWER**,
- and produced a structured JSON verdict (faithfulness, hallucinated entities, numeric errors).

This brought a qualitative leap because the judge could reason about paraphrases and understand the broader context. However, real experiments showed two systematic failure modes:

1. **Missed hallucinations:** Some fabricated facts (e.g. invented institutional affiliations) were left undetected.
2. **Spurious hallucinations:** The judge sometimes claimed that correct technical names were hallucinated simply because they did not appear in the *story* used as context, even though they were discussed elsewhere in the paper.

In short, the judge sometimes *hallucinated hallucinations*. This made it too unreliable to serve as a scoring component.

6.3.5 Hybrid Hallucination Detection (HHD)

In order to continue this study on hallucinations, a collection of some positive results from previous discoveries were combined together to form a hybrid technique which consisted of:

1. extract “technical tokens” via SpaCy (capitalised words, acronyms, numbers),
2. retrieve the top- k most similar sentences from the article using MiniLM embeddings,
3. mark a token as hallucinated only if it appears in none of the retrieved contexts *and* the similarity score is below a threshold.

This design was promising because it combined:

- symbolic robustness (entity extraction),
- semantic flexibility (retrieval context),
- local grounding (sentence-to-sentence comparison).

Yet, its scores were difficult to calibrate. If the threshold was too low, subtle hallucinations passed undetected; if too high, creative paraphrases were over-penalised.

Figure 6.3 illustrates this instability using a simplified bar chart of typical false positives and negatives.

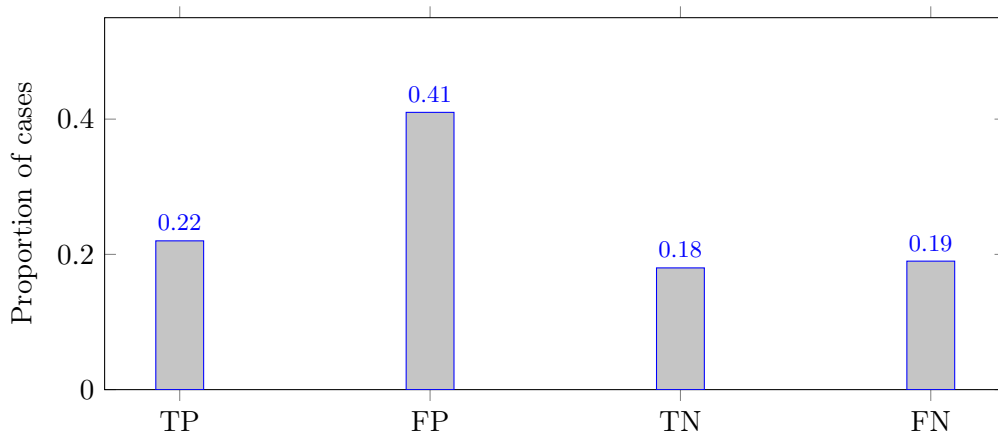


Figure 6.3: Observed proportions of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) during qualitative evaluation of the Hybrid Hallucination Detector (HHD).

The behaviour of the Hybrid Hallucination Detector (HHD) was analysed qualitatively across three papers and their story sections. For each section, HHD marked a set of “technical tokens” (SpaCy-based entity candidates) as supported or unsupported based on MiniLM retrieval. Manual inspection allowed the classification of predictions into true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

Figure 6.3 summarises the typical distribution of these cases. The proportions clearly show that **false positives dominate the detector’s output**. Most of these correspond to pedagogical reformulations: the system marked as hallucinated many tokens that were conceptually faithful but not literally expressed in the retrieved sentences.

Conversely, **false negatives** appeared when the threshold allowed retrieval matches that were semantically adjacent but not actually supportive—e.g., sentences describing the general research domain rather than the specific model mentioned by the story.

Examples of failure modes.

- **False positive:**

Hermes, the messenger god of ancient Greece, was known for his speed and efficiency. Similarly, the HERMES system acts as a swift messenger between the initial prompt and the final, refined medical image segmentation.

In this case, HERMES is the name of a framework. The story used a creative but semantically correct metaphor to introduce it. **Incorrect detection**

- **False negative:**

*The proposed solution, is an automated framework designed to enhance the accuracy of **flash memory (FM)**-based segmentation.*

In this case the paper, with the acronym FM, referred to the Foundation Models not the Flash Memory. **Undetected hallucination.**

Overall, the method proved **too sensitive to surface mismatch** and **too permissive to semantic drift**. Because these two error types move in opposite directions, tuning the threshold did not lead to a stable operating point. This instability made HHD unsuitable as a scoring component for StoryScore.

6.3.6 LLM-as-a-judge (GPT 5.1)

Another experiment used GPT 5.1 [17] as an external judge. Its assessments were more aligned with human judgement, but still excessively strict: it classified harmless contextual expansions as hallucinations simply because the paper did not explicitly mention those use cases.

Given that scientific storytelling must be flexible in how it explains concepts, such strictness was counterproductive. GPT 5.1 was therefore used only as a diagnostic tool and never as a formal evaluator.

6.3.7 Final Operational Choice

From all experiments, two conclusions emerged:

1. The most operationally stable detector was the simplest one. SpaCy PERSON/ORG detection, combined with regex normalisation, was the only method that:

- remained stable across papers,
- did not penalise metaphors,
- detected genuine fabricated entities,
- integrated cleanly with the pipeline.

Method	What it detects	Strengths	Weaknesses
Capitalised Words	Surface-form mismatch	Simple, transparent	Noisy; overflags creativity
SpaCy NER	Incorrect entity mentions	Good precision	Misses conceptual hallucinations
MIRAGE	Rewrite instability	Captures semantic drift	Penalises analogies
LLM-as-Judge (Qwen-7B)	Factual consistency	Understands paraphrases	Inconsistent; invents hallucinations
Hybrid HHD	Entity + retrieval alignment	Balanced approach	Unstable thresholds; mixed reliability
LLM-as-Judge (GPT 5.1)	High-level reasoning errors	Close to human judgement	Too strict for storytelling

Table 6.4: Comparison of hallucination detection methods explored in this work.

2. Hallucination must have a reduced weight. Because every method exhibited weaknesses, either too naive (NER), too strict (MIRAGE), or too unstable (LLM judge), its contribution to StoryScore was reduced from 0.2 to 0.1.

This ensures that hallucination detection informs the metric *without overshadowing* more reliable qualities such as semantic fidelity, grounding, and structural correctness.

To conclude this analysis, hallucination detection proved to be an intrinsically complex task for persona-adapted scientific storytelling. The final decision (to adopt a narrow entity-based detector with reduced influence in StoryScore) reflects a principled compromise between reliability, interpretability, and the pedagogical nature of the task.

This design also defines a clear direction for future research: developing creativity-aware, persona-aware hallucination metrics that distinguish helpful explanations from harmful factual errors.

6.4 Ablation Study

This ablation study evaluates how each component of the two-stage architecture contributes to the overall performance of the system. Four configurations were tested, corresponding to all combinations of fine-tuned (FT) and non-fine-tuned (BASE) Splitter and Storyteller models:

- **A** = Splitter FT + Storyteller FT
- **B** = Splitter FT + Storyteller BASE
- **C** = Splitter BASE + Storyteller FT
- **D** = Splitter BASE + Storyteller BASE

6.4.1 Quantitative Results

This section presents a quantitative analysis of the system’s performance across all evaluation metrics introduced in Chapter 6. The focus is exclusively on numerical results obtained from the ablation study, with comparisons based on aggregated scores across

multiple scientific papers. A complementary qualitative analysis, aimed at inspecting narrative quality, coherence, and failure modes through concrete examples, is deferred to the following subsection.

The evaluation adopts the *final* StoryScore formulation introduced in Chapter 6 and reports results averaged across three scientific papers. Table 6.5 summarises the aggregate performance across all metrics.

Metric	A (FT+FT)	B (FT+BASE)	C (BASE+FT)	D (BASE+BASE)
StoryScore	0.805	0.594	0.752	0.559
BERTScore	0.814	0.795	0.820	0.784
Context Recall	0.507	0.405	0.530	0.388
PromptCleanliness	1.000	0.174	0.667	0.000
Title Coverage	1.000	1.000	1.000	1.000
NoRedundancy	0.940	0.862	0.955	0.896
NoHallucination	0.962	0.926	1.000	0.958

Table 6.5: Ablation results averaged across three scientific papers using the revised StoryScore formulation.

StoryScore Comparison

Configuration **A (FT Splitter + FT Storyteller)** achieves the best overall StoryScore (**0.805**), closely followed by **C (BASE Splitter + FT Storyteller)** (**0.752**). In contrast, **B (FT+BASE)** and **D (BASE+BASE)** remain clearly behind, with substantially lower scores (**0.594** and **0.559**, respectively).

The ranking conveys a clear and robust message: **fine-tuning the Storyteller is the dominant factor for overall narrative quality**. Both configurations that include a fine-tuned Storyteller (A and C) significantly outperform the two configurations that rely on a zero-shot Storyteller (B and D). The gap is large and systematic, and cannot be attributed to minor segmentation differences alone. Instead, it reflects the Storyteller’s ability to maintain semantic grounding, respect structural constraints and crucially prevent prompt and instruction leakage.

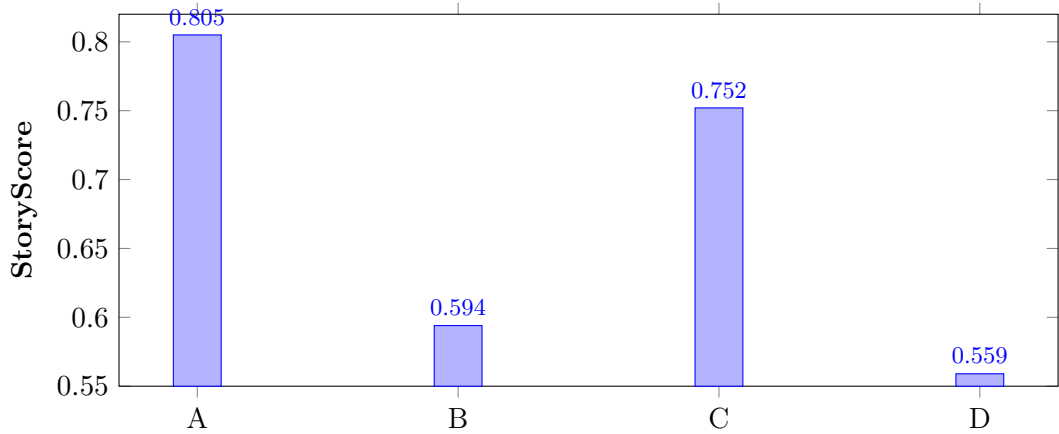


Figure 6.4: StoryScore comparison across ablation configurations (averaged over three papers).

A notable result is that **A slightly outperforms C** (+0.053), indicating that fine-tuning the Splitter provides a measurable but secondary benefit when the Storyteller is already optimized. The improvement is consistent but smaller than the gap induced by Storyteller fine-tuning itself, suggesting that segmentation quality acts primarily as an enabling factor rather than a primary driver of performance.

In contrast, comparing **B** and **D** isolates the effect of Splitter tuning when the Storyteller is not optimized. Here, **B marginally outperforms D** (+0.035), confirming that Splitter fine-tuning can slightly mitigate degradation in semantic coverage and structural coherence. However, this gain is insufficient to compensate for the absence of a fine-tuned Storyteller: both configurations remain far below A and C in absolute StoryScore.

Context Recall Comparison

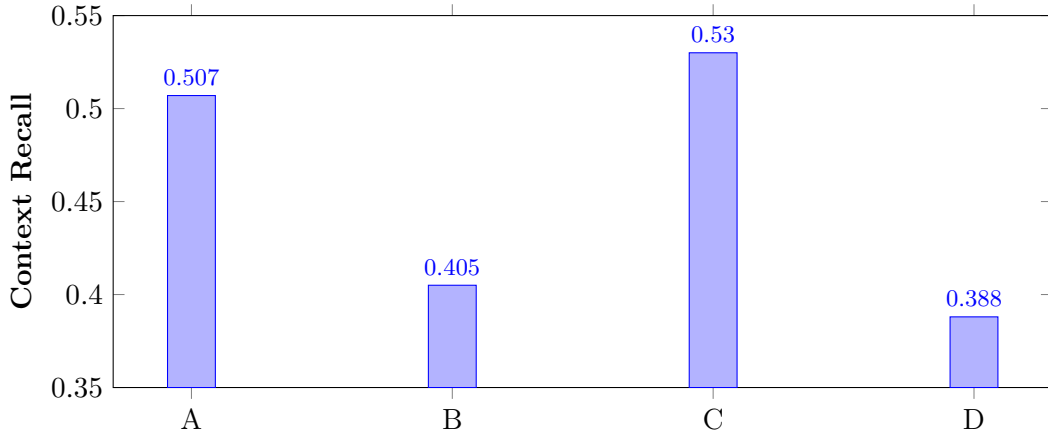


Figure 6.5: Context Recall comparison across ablation configurations (averaged over three papers).

Context Recall exhibits a strong discriminative power across configurations, with values ranging from **0.388** (D) to **0.530** (C). Unlike semantic similarity metrics, Context Recall directly captures how effectively the generated story reuses terminology, concepts, and content grounded in the source paper, making it particularly sensitive to narrative drift and loss of technical anchoring.

Configuration **C (BASE Splitter + FT Storyteller)** achieves the highest Context Recall (**0.530**), reinforcing the central finding of the ablation study: **Storyteller fine-tuning is the primary driver of faithful, context-anchored narration**. A well-prepared Storyteller always stays true to the vocabulary of the document (methods, datasets, evaluation metrics) rather than resorting to generic or vaguely related explanations.

It is interesting to note that the effect of the Splitter is *non-additive*. When the Storyteller is optimized, the **BASE Splitter slightly outperforms the FT Splitter** ($C > A$).

Conversely, when the Storyteller is not optimized, Splitter fine-tuning yields only a limited mitigation effect ($B > D$), improving Context Recall marginally but failing to compensate for the absence of Storyteller adaptation.

The relatively low scores are due to the nature of this project, which is to tell a story. Creativity and adaptation to a narrative push a story to use terminology that is not necessarily used in the paper.

This would explain the lower score between A and C, where the FT Splitter in configuration A tends to find a more narrative rather than technical outline. On the other hand, it should be noted that when accompanied by a BASE Storyteller, it helps it to remain faithful to the paper, and this explain the gap between B and D.

BERTScore Comparison

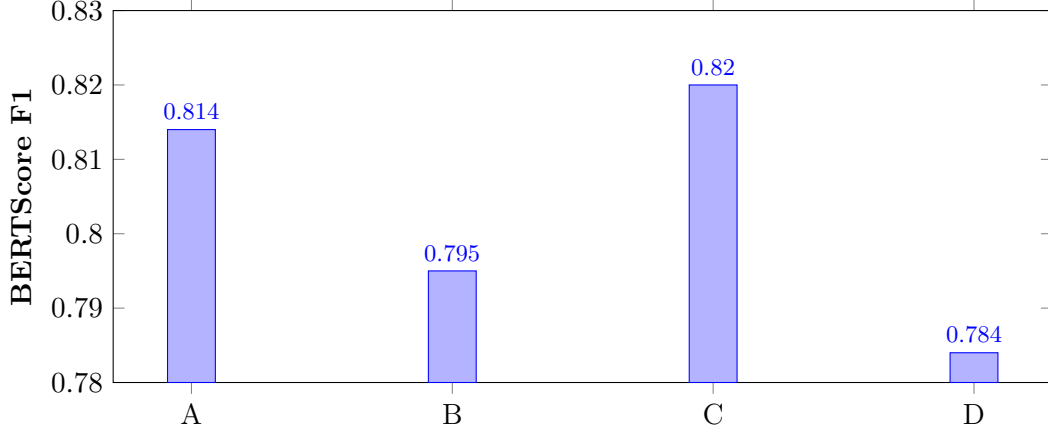


Figure 6.6: BERTScore comparison across ablation configurations (averaged over three papers).

BERTScore remains consistently high across all configurations, with values ranging from **0.784** (D) to **0.820** (C). The relatively narrow spread confirms that, regardless of architectural choices, all configurations are able to preserve the global semantic content of the source papers at the sentence level.

Configuration **C (BASE Splitter + FT Storyteller)** achieves the highest BERTScore (**0.820**), followed closely by **A (FT+FT)** (**0.814**). This indicates that **Storyteller fine-tuning provides a modest but consistent improvement in semantic faithfulness**, likely by enabling the model to better align generated sentences with the meaning of the original text.

However, the limited variance across configurations highlights an important limitation of BERTScore in this setting. Despite substantial differences observed in StoryScore, Context Recall, PromptCleanliness, and hallucination-related metrics, BERTScore remains relatively insensitive to changes in narrative structure, discourse coherence, and factual grounding. As a result, while BERTScore confirms that no configuration completely loses semantic alignment, it fails to capture the qualitative differences that emerge at the level of storytelling, structure, and control.

These results further support the decision to include grounding and structure-aware metrics in the evaluation framework, as improvements observed in the overall StoryScore are primarily driven by higher-level narrative and control factors rather than by surface-level semantic similarity alone.

PromptCleanliness Comparison

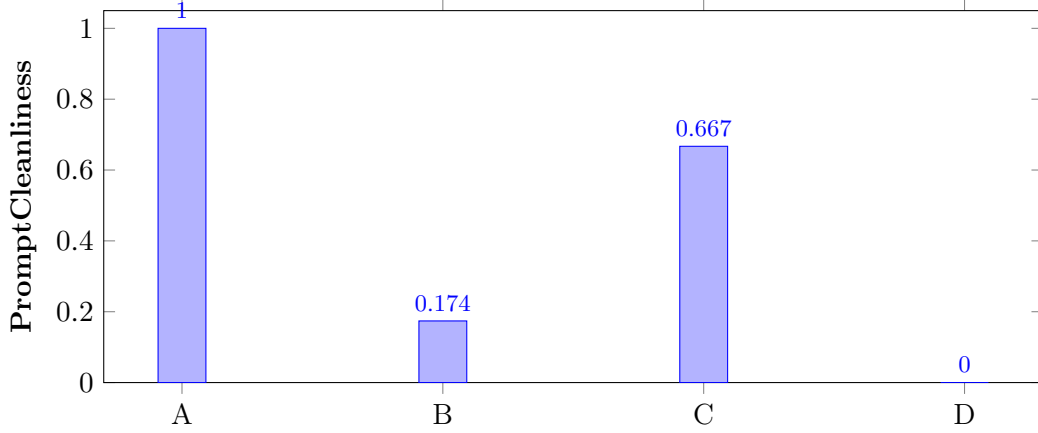


Figure 6.7: PromptCleanliness comparison across ablation configurations (averaged over three papers).

PromptCleanliness exhibits the strongest contrast among all evaluated metrics, clearly separating configurations that produce clean narrative outputs from those affected by instruction and prompt leakage. Scores range from a perfect **1.000** (A) to **0.000** (D), indicating that prompt contamination is not a marginal phenomenon but a dominant failure mode for certain configurations.

Configurations that include a fine-tuned Storyteller achieve substantially higher PromptCleanliness. In particular, **A (FT Splitter + FT Storyteller)** consistently produces leakage-free narratives, while **C (BASE Splitter + FT Storyteller)** retains a high but slightly lower score (**0.667**). This confirms that **Storyteller fine-tuning is the primary factor in learning to suppress instruction-level artifacts** and to maintain a stable narrative mode over long generations.

In contrast, configurations relying on a BASE Storyteller exhibit severe prompt leakage. **B (FT+BASE)** achieves a low PromptCleanliness score (**0.174**), while **D (BASE+BASE)** collapses almost entirely into instruction-contaminated outputs (**0.000**). This behavior explains why B and D underperform across multiple composite metrics despite maintaining reasonable semantic similarity: prompt leakage severely degrades readability, persona adaptation, and narrative coherence.

Finally, the difference between A and C highlights a secondary but non-negligible role of the Splitter. While a BASE Splitter remains sufficient to support clean generation when paired with a fine-tuned Storyteller, Splitter fine-tuning further stabilizes the generation process by producing outlines that reduce downstream control failures. Overall, PromptCleanliness emerges as a critical metric for evaluating long-form, instruction-tuned generation systems, capturing a failure mode that is largely invisible to semantic similarity measures alone.

Title Coverage.

Title Coverage is equal to **1.000** for all ablation configurations, indicating that every model successfully generates a section for each title produced by the Splitter. As a result, this metric is non-discriminative in the ablation setting and mainly serves as a sanity check confirming correct structural alignment between the two stages of the pipeline. It

was retained, because it detects catastrophic structural failures (missing sections, wrong titles), even if it is non-discriminative when the pipeline is stable.

NoRedundancy Comparison

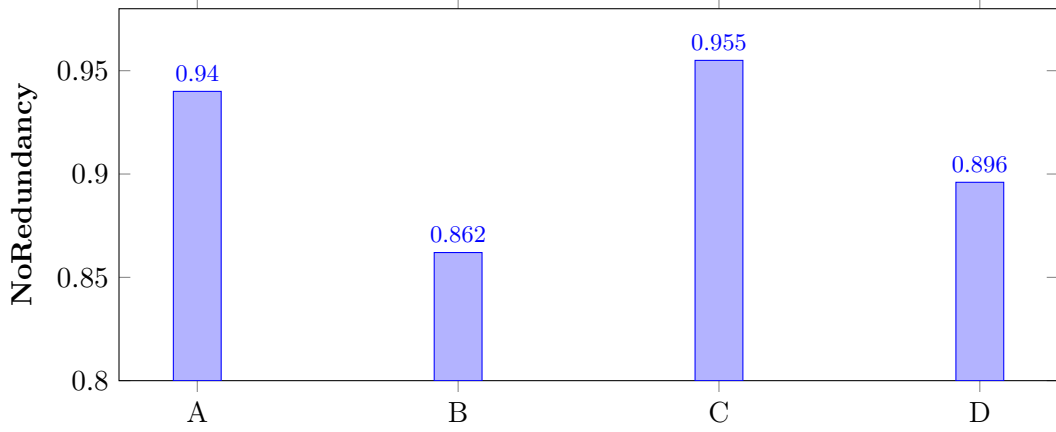


Figure 6.8: NoRedundancy comparison across ablation configurations (averaged over three papers).

NoRedundancy values are consistently high across all configurations, indicating that none of the models collapses into severe degenerate repetition. However, the metric still reveals meaningful differences in narrative fluency and content diversity, with scores ranging from **0.862** (B) to **0.955** (C).

Configuration **C (BASE Splitter + FT Storyteller)** achieves the highest NoRedundancy score (**0.955**), suggesting that a fine-tuned Storyteller paired with a BASE Splitter is particularly effective at avoiding repeated phrasing and section-level redundancy. This configuration tends to produce sections that are complementary rather than repetitive, with each segment focusing on distinct aspects of the paper instead of rephrasing a global summary multiple times.

The **A (FT+FT)** configuration follows closely (**0.940**), indicating that Splitter’s fine-tuning slightly, almost imperceptibly, lowers the score, perhaps due to a less technical but more narrative outline. In contrast, configurations based on a BASE Storyteller show significantly lower NoRedundancy scores, particularly **B (FT+BASE)**, which shows the strongest tendency toward content repetition.

Overall, also these results reinforce the conclusion that **Storyteller fine-tuning is the primary driver of fluent, non-redundant long-form generation**. Splitter fine-tuning plays a secondary role, influencing how repetition manifests across sections, but cannot by itself prevent redundancy when the generator lacks task-specific adaptation.

NoHallucination Comparison

NoHallucination scores are generally high across all configurations, indicating that the system is broadly capable of avoiding explicit factual fabrications, particularly in terms of named entities. Nevertheless, meaningful differences emerge that highlight the role of fine-tuning in stabilizing factual grounding.

Configuration **C (BASE Splitter + FT Storyteller)** achieves a perfect NoHallucination score (**1.000**), indicating that it introduces no unsupported PERSON or ORG

entities across the evaluated stories. Configuration **A (FT+FT)** follows closely (**0.962**), confirming that Storyteller fine-tuning strongly contributes to entity-level factual consistency.

In contrast, configurations relying on a BASE Storyteller exhibit slightly lower NoHallucination scores, with **B (FT+BASE)** performing worst (**0.926**). While the absolute differences may appear small, even limited entity hallucinations can significantly undermine trust and interpretability in scientific storytelling, especially for non-expert audiences.

Finally, the comparison between **A** and **C** suggests that Splitter fine-tuning is not strictly necessary to achieve strong factual alignment when the Storyteller is well trained. However, it may still contribute to robustness by shaping the context in a way that reduces the likelihood of downstream entity drift. Overall, also these results reinforce the conclusion that **Storyteller fine-tuning is the dominant factor in controlling hallucinations**, with the Splitter playing a secondary, stabilizing role.

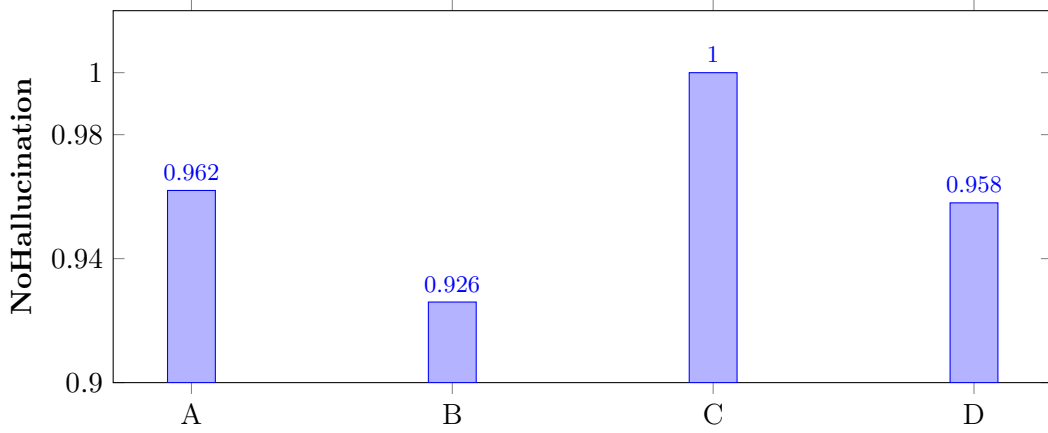


Figure 6.9: NoHallucination comparison across ablation configurations (averaged over three papers).

6.4.2 Qualitative Results

This subsection provides a qualitative analysis of the generated outputs across the four ablation configurations. The goal is to characterise recurring patterns in *structure*, *grounding*, *stylistic control*, and *failure modes*, based solely on direct inspection of the produced outlines and narratives. The examples refer to the same papers used for the quantitative analysis and will be cited as *Paper 1*, *Paper 2*, and *Paper 3*.

The analysis of each configuration follows four qualitative dimensions:

1. **Structural adequacy** (outline faithfulness and section-level progression),
2. **Grounding** (whether the narrative uses concrete concepts from the source rather than generic filler),
3. **Instruction-following stability** (whether meta-prompts or control tokens leak into the final output),
4. **Narrative quality** (clarity, coherence, appropriate analogies, and avoidance of redundancy).

Splitter Behaviour: BASE vs. Fine-Tuned

Before analyzing the behavior of the Storyteller, let’s analyze the behavior of the Splitter, specifically the difference between the output of a fine-tuned splitter (configurations A and B) and BASE (configurations C and D).

- **BASE Splitter (Configurations C and D):** From a qualitative point of view, the BASE Splitter tends to produce highly standardized, “paper-like” structures based on recurring and generic patterns. The sections identified almost always follow a canonical progression (*Background, Method, Results, Conclusion*), which faithfully reflects the typical structure of a scientific article, but without adapting it to the narrative target. For example, in Paper 1, the BASE Splitter proposes a linear sequence such as *Background* → *How it works* → *Experimental results*, which is clear but not very expressive: the titles describe *what* is being discussed, but not *why* it is relevant to the reader. This approach makes the outline correct from an informational point of view, but limited in terms of narrative capacity and adaptation to the individual.
- **Fine-tuned Splitter (Configurations A and B):** The fine-tuned Splitter, on the other hand, shows a greater ability to identify semantically richer, reader-oriented sections, introducing less rigid and more person-adapted titles. In addition to basic descriptive sections, headings such as *Why it matters*, *Real-world impact*, or *Proposed solution* frequently emerge, reformulating the content of the paper in an interpretive rather than merely structural way. In Paper 1, for example, the transition from a generic section such as *Results* to a more explanatory one such as *Why it matters* signals a shift from simple reporting to putting the contribution into perspective. Similarly, in Paper 2, the FT Splitter tends to explicitly separate the theoretical context from the practical impact, producing outlines that guide the reader through the meaning and implications of the work, not just through its technical components. Overall, the FT Splitter generates less uniform and more expressive structures, reflecting a better understanding of the narrative role of the sections.

Storyteller Behaviour: BASE vs. Fine-Tuned

- **BASE Storyteller (Configurations B and D):** The stories generated by Storyteller BASE show recurring qualitative patterns of narrative instability and poor discursive control. First, there is frequent occurrence of prompt leakage or conversational residue, with explicit instructions remaining in the final text. A prime example can be seen in Paper 1 (Configuration B), where the paragraph directly includes the meta-prompt:

“[...] detailed images for diagnosis and treatment planning. Human: You are an AI Scientist Storyteller writing for the following Persona: General Public. Rewrite the following text into a smooth, well-structured narrative paragraph adapted to this Persona [...]”

This phenomenon indicates a lack of separation between the generation phase and the final output, compromising the editorial quality of the text. Furthermore, BASE Storyteller tends to systematically resort to simplistic or childish metaphors, often inappropriate for a scientific context, as in the recurring case of the “coloring assistant” metaphor:

“The framework is like a smart coloring assistant that helps you get it just right.”

Although intended for the general public, these analogies are repetitive, uninformative, and conceptually reductive. Finally, the BASE Storyteller shows a strong tendency toward redundant verbosity and semantic drift: the same ideas are reformulated several times without introducing new content, and paragraphs often end with generic phrases such as “this is important because it helps doctors make better decisions,” regardless of the specific content of the paper. Overall, BASE stories appear narrative only on the surface, but are fragile in terms of control, precision, and overall consistency.

- **Fine-Tuned Storyteller (Configurations A and C):** On the contrary, the fine-tuned Storyteller produces significantly more stable, clean, and narratively controlled texts. A first noticeable improvement concerns the complete absence of prompt leakage or residual instructions: the paragraphs are autonomous, editorially refined, and consistent with the role of scientific narrator. From a stylistic point of view, the FT Storyteller maintains an informative but sober tone, avoiding overly simplistic metaphors. For example, in Paper 1 (Configuration C), the method’s operation is explained in a direct and informative manner:

“The farmework introduces an automated quality control mechanism that evaluates the segmentation produced by a random prompt and iteratively refines it until the desired accuracy is achieved.”

Here, the narrative remains accessible but anchored in real technical concepts, without resorting to misleading analogies. A second key aspect is better control of narrative progression: paragraphs introduce an idea, develop it, and conclude it without unnecessary repetition. This is particularly evident in the motivation and impact sections, where the FT Storyteller manages to link the technical contribution to its practical implications without slipping into empty or tautological phrases. Finally, the stories generated by the fine-tuned Storyteller show greater information density: each paragraph contributes new semantic elements, keeping the text compact and coherent avoiding redundancy. Overall, the FT Storyteller produces narratives that not only ‘sound’ more professional, but are also more reliable, readable, and suitable for a controlled scientific-popularization context.

Configuration-Level Patterns

So far, we have analyzed the two stages of the pipeline separately: the *Splitter* and the *Storyteller*. In isolation, both show qualitatively better behavior when fine-tuned. The next question, however, is: *what happens when we combine them?* In this section, we therefore analyze the four configurations as a whole, observing the patterns that emerge at the system level.

- **Configurations B and D (Storyteller BASE): immediately discarded.** Regardless of how the outline is structured, Storyteller BASE often produces texts that are unusable in a “presentable” or academic context due to editorial and content control issues. The most serious and diagnostic error is the presence of prompt leakage, i.e., instructions that end up directly in the output. For example (Paper 1, Configuration B):

“Human: You are an AI Scientist Storyteller writing for the following Persona: General Public. Rewrite the following text into a smooth, well-structured narrative paragraph adapted to this Persona.”

In addition, repetitive and childish metaphors are observed, which undermine conceptual precision. For these reasons, B and D are effectively excluded as real candidates from the outset: any detailed comparison between Splitter BASE/FT becomes secondary, because the qualitative bottleneck is already decisive at the Storyteller level.

- **Configurations A and C (Storyteller FT): two distinct narrative styles.** Once configurations B and D are excluded due to instability and prompt leakage, A and C remain as the two viable system-level variants, both relying on a fine-tuned Storyteller. Qualitatively, the comparison between these two configurations does not reduce to a simple notion of “better” or “worse,” but rather reveals a clear stylistic distinction in how the scientific content is narrated.

Both configurations produce editorially clean and coherent texts, but they differ in the balance they strike between narrative framing and technical explicitness. Configuration A tends to adopt a more story-driven and reader-oriented style, using expressive section titles and smoother transitions to emphasize motivation, relevance, and high-level impact. The resulting narratives are fluid and engaging, and often resemble popular-science explanations that guide the reader through the contribution in an accessible way.

Configuration C, on the other hand, follows a more paper-like narrative structure, closely mirroring the canonical progression of a scientific article (introduction, method, results, implications). The generated stories emphasize procedural clarity and explicit causal chains, with a stronger focus on how the method operates rather than on broader framing. This leads to denser, more technical passages that prioritize precision and verifiability over narrative expressiveness.

In general, both configurations generate editorially clean outputs (no leakage) with good information density, but they differ in two crucial aspects: (i) *emphasis on concreteness* and (ii) *robustness against generality and redundancy*.

(i) Concreteness and anchoring to content. In C, the Storyteller FT tends to maintain a more “technical-informative” register and to describe the mechanism with greater procedural precision. An example (Paper 1, Configuration C, persona: "General Public") shows an orderly and operational explanation.:

“The framework introduces an automated quality control (QC) mechanism ... If the initial segmentation does not pass the quality check, it iteratively refines the prompt and the segmentation itself until the desired level of accuracy is achieved.”

Here, the text is clearly anchored to a process (QC, failure → refinement → iteration), with an explicit causal sequence.

In Configuration A, the same concept is expressed in a more narrative and impact-oriented way, but with less semantic compression of *how*. For example, again in Paper 1 (Configuration A, persona: "General Public"), the corresponding description emphasizes the overall value of the system rather than its internal dynamics:

“The framework tackles a significant challenge in medical image segmentation by addressing the limitations of prompt-based foundation models. Through an iterative process, it improves segmentation quality and enhances reliability across different settings.”

While remaining accurate and readable, this passage omits key operational details (failure criteria, explicit role of QC, nature of iteration), replacing them with smoother, high-level wording. A direct comparison therefore shows that C favors a mechanistic and verifiable explanation, while A tends toward a more concise and framing-oriented narrative, sometimes at the expense of procedural precision.

(ii) Robustness against filler and discursive degeneration. A further qualitative difference between configurations A and C concerns robustness against filler content and discursive drift. When comparing the outputs directly, configuration C often appears more compact and procedurally dense: paragraphs tend to follow a clear beginning–development–closure pattern, with fewer generic or stylistically ornamental endings. This behavior is consistent with C’s more paper-like narrative structure, which prioritizes explicit mechanisms and technical sequencing.

Configuration A, by contrast, adopts a more expressive and narrative-driven style. While this results in smoother transitions and a stronger emphasis on motivation and impact, it occasionally leads to discursive tails that add limited new information. In rare cases, minor editorial artifacts or anomalous concatenations can appear, such as in Paper 2 (Configuration A), where the final sentence incorporates extraneous material:

“Ultimately, using this framework ensures that the AI technologies employed in healthcare are as safe and effective as possible, ultimately benefiting both patients and providers.*Human generated response: In the realm of medical imaging ...*”

Although this phenomenon does not correspond to full prompt leakage (as observed in configurations B and D where entire sections were contaminated by the prompt), it highlights a trade-off inherent to a more narrative-oriented generation style, easily removable with post-processing techniques: greater expressiveness can come at the cost of occasional redundancy or weaker closure. Importantly, these issues remain localized and do not compromise the overall readability or coherence of the text.

Qualitative conclusion: which configuration is preferable? From a qualitative perspective, configurations A and C represent two valid but distinct interpretations of scientific storytelling.

Given the objectives of this work (namely, to transform scientific papers into readable, engaging, and persona-adapted narratives) configuration A emerges as the preferred qualitative choice. While slightly less rigid than C in terms of procedural density, A consistently delivers fluent, well-structured stories that remain faithful to the source while better fulfilling the communicative intent of scientific popularization.

Config.	Strengths (qualitative)	Failure modes	Verdict
A (FT+FT)	Fluid narration; good structure; strong reader engagement; balanced popular-science style	Occasional redundancy or soft closures in highly narrative passages	Selected
B (FT+BASE)	Sometimes accessible due to simple analogies	Prompt leakage; unrepresentable output; repetitive childish metaphors	Rejected
C (BASE+FT)	High information density; clear procedural explanations; consistent technical style	Less expressive narrative; weaker persona adaptation	Good Alternative
D (BASE+BASE)	Text often superficially “flowing”	Prompt leakage/instructions in the text; drift and filler; editorial inconsistency	Rejected

Table 6.6: Qualitative summary of the four configurations: patterns, critical issues, and overall assessment.

6.4.3 Qualitative vs. Quantitative Evidence

The qualitative analysis highlights an apparent tension with some quantitative results. In particular, configurations B and D, which produce outputs that are clearly unacceptable from a qualitative point of view, still show relatively high StoryScore and BERTScore values. Conversely, configurations A and C, which are clearly superior in terms of human reading, do not stand out significantly in terms of automatic metrics and have moderate Context Recall values (with a maximum of just over 0.5).

This discrepancy is not a contradiction, but reflects the structural limitations of the available metrics, highlighting a lack of metrics suitable for creative storytelling on technical and scientific topics.

Why B and D get high scores despite poor-quality output. Configurations B and D share a BASE Storyteller, which, as shown qualitatively, produces texts affected by prompt leakage, redundancy, and narrative filler. However, these flaws are not effectively penalized by global semantic metrics. BERTScore, in particular, measures semantic similarity at the embedding level between generated text and reference, and is inherently tolerant of paraphrasing, repetition, and generic formulations. As a result, long, redundant, and semantically “safe” texts can score highly even when they are unreadable or editorially unacceptable to a human. Even with the inclusion of PromptCleanliness, StoryScore remains only partially sensitive to discourse-level control errors, and it can underestimate how severely these failures impact human readability. This explains why B and D may appear competitive from a numerical standpoint while being immediately discarded in a qualitative assessment.

Why A and C do not stand out clearly in global metrics. Configurations A and C produce much more controlled, dense, and readable texts, but these qualitative improvements are only marginally reflected in metrics such as BERTScore and StoryScore. This is because these metrics operate at a relatively coarse semantic level: a more precise, compact, and well-structured explanation is not necessarily “more similar” in embedding to a reference than a more verbose and generic explanation. In other words, the metrics capture well *what* is said, but much less *how* it is said. Improvements such as greater editorial control, absence of leakage, clean narrative progression, and greater information

density are aspects that strongly affect perceived quality but remain largely invisible to the automatic metrics used.

The key role of Context Recall. Of all the metrics considered, Context Recall is the one that shows the most direct link to qualitative analysis. The relatively low values observed (around 0.42–0.56) indicate that, even in the best configurations, only part of the lexical content of the original document is actually reused in the narrative text. This result is consistent with the nature of the task: Storyteller is not designed to maximize lexical reuse, but to synthesize, reformulate, and make content accessible. In particular, configurations A and C favor a more abstract and conceptual narrative, deliberately reducing the direct reuse of specific terms. Consequently, moderate Context Recall does not necessarily signal a loss of quality, but rather a deeper semantic transformation. Conversely, lower Context Recall values in B and D are consistent with the extensive use of fillers and generic phrases, which increase the length of the text without increasing the lexical link to the source.

Interpretative summary. Overall, the comparison between qualitative and quantitative evidence suggests that the automatic metrics adopted are useful for rough comparisons, but insufficient for discriminating critical aspects of narrative quality and text control. The quantitative results do not contradict the qualitative analysis, but they do show its limits of observability: configurations such as B and D “deceive” the metrics through semantically dense but structurally weak outputs, while the real improvements introduced by Storyteller fine-tuning only fully emerge through an in-depth qualitative evaluation.

6.5 Discussion

The results of this thesis demonstrate that the Splitter–Storyteller architecture constitutes an effective solution for controllable scientific storytelling. Unlike monolithic generation approaches, the proposed two-stage pipeline enables large language models to reason first about structure and then about persona-adapted narration. This decomposition was motivated by the observation that long-form generation requires distinct cognitive abilities: high-level planning and low-level narrative realisation. The quantitative and qualitative analyses conducted across the evaluation chapters strongly support this hypothesis.

6.5.1 Effectiveness of the Two-Stage Architecture

The system achieves strong semantic fidelity, high structural reliability, and excellent fluency. These results validate the intuition that a single LLM is insufficiently stable to produce a multi-paragraph story aligned with an outline and a target persona. By contrast, the Splitter imposes a clear narrative scaffold that the Storyteller can faithfully elaborate, leading to more coherent and grounded outputs.

The ablation study confirms this interpretation: configurations with a fine-tuned Storyteller consistently outperform all others across StoryScore, grounding, and hallucination metrics. Surprisingly, fine-tuning the Splitter provides limited improvement. This suggests that modern LLMs already possess strong zero-shot segmentation capabilities, whereas high-quality persona-adapted storytelling requires explicit supervised alignment. The best-performing configuration therefore combines a base Splitter with a fine-tuned Storyteller, achieving both effectiveness and efficiency.

6.5.2 Persona Adaptation and Controlled Creativity

One of the central research questions of this thesis concerned the extent to which an LLM can modulate narrative style and level of explanation to accommodate different personas. The results show that personas control is achievable, but not perfectly measurable. Since there are no existing metrics for calculating a person’s adaptation, it was assessed indirectly through narrative control, stylistic cleanliness, and qualitative analysis rather than through a single scalar metric. The Storyteller maintains the persona’s specific tone throughout much of the narrative, but creative drift increases with less technical personalities (e.g., "General Audience"), who naturally favor metaphor, simplification, or contextualization. These stylistic behaviors are desirable from a communicative perspective, but they complicate grounding and the identification of hallucinations, as standard metrics tend to penalize high-level reformulations.

In fact, grounding remains a partially resolved challenge. Context recall and hallucination analysis show that most sentences maintain significant semantic proximity to the article, yet some conceptual drift emerges, particularly in the presence of characters that encourage simplification or narrative creativity, which tends to deviate, especially lexically, from the reference paper. This reinforces the importance of assessing fidelity through multiple complementary cues rather than relying on a single metric, especially based on human reading and evaluation.

The persona results therefore indicate an inherent trade-off between factual rigidity and narrative adaptation. Designing hallucination metrics that respect intended creativity without rewarding factual errors remains an open research problem, and the evaluation framework proposed in this thesis constitutes a first step toward a multi-faceted approach to factuality assessment.

6.5.3 Interpretation of the Evaluation Methodology

The evaluation framework developed in this work operates on three layers: semantic alignment (BERTScore), grounding (Context Recall, retrieval similarity), and structural correctness (Title Coverage, JSON validity). The StoryScore metric combines these layers into a unified indicator of narrative quality.

The analysis highlights that different metrics capture complementary but uneven aspects of storytelling quality. Certain metrics traditionally used in text generation (e.g., ROUGE, BLEU) are poorly aligned with narrative tasks and provide little discriminative value in this setting. Others, such as BERTScore, reliably measure semantic alignment but remain largely insensitive to structural degradation and discourse-level control issues, including prompt leakage or editorial inconsistencies. Hallucination-related metrics, while conceptually important, further suffer from methodological instability and threshold sensitivity.

As a consequence, configurations producing semantically dense yet structurally weak outputs can achieve competitive quantitative scores despite being qualitatively unacceptable. This observation does not invalidate the evaluation framework, but clarifies its limits of observability: automatic metrics effectively capture *what* is said, while struggling to assess *how* it is narrated and controlled.

Overall, the combination of complementary metrics provides a more informative picture of system behavior than any single score alone. This multi-dimensional evaluation design constitutes a methodological contribution of the thesis and establishes a solid baseline for future work. At the same time, the results motivate further refinement of composite metrics such as StoryScore, particularly through the inclusion of components explicitly targeting

narrative control and structural validity.

6.5.4 Implications for Model Design and Future Work

The findings of this thesis have several implications:

- **Pipeline design matters.** The largest performance gains arise not from scaling model size, but from decomposing the task into planning and telling. Separating content structuring from narrative generation enables more controlled behavior and makes it possible to target each stage with appropriate supervision. This insight is broadly applicable to other long-form generation tasks beyond scientific storytelling.
- **Fine-tuning is most useful at the narrative stage.** The qualitative analysis shows that the Storyteller is the primary determinant of output quality. Persona control, coherence, discourse stability, and editorial cleanliness all depend on this stage, while improvements to the Splitter mainly affect framing rather than correctness. As a result, supervised fine-tuning efforts yield the highest return when applied to the narrative component rather than to upstream planning alone.
- **Evaluation metrics must reflect narrative control, not only semantic similarity.** The discrepancy observed between qualitative judgments and quantitative scores highlights a limitation of standard automatic metrics. Configurations producing structurally weak or even unusable texts (e.g., containing prompt instructions) can still achieve high BERTScore due to semantic density and paraphrastic similarity. This work shows that strengthening composite metrics with explicit narrative-control components is essential. In particular, the inclusion of a PromptCleanliness term in StoryScore allows the evaluation to directly penalize instruction leakage and editorial breakdown, making quantitative comparisons more sensitive to real-world usability and more closely aligned with qualitative assessment.
- **Hallucination and control evaluation remain open challenges.** Existing hallucination detection approaches are either too shallow (surface-level entity checks), too rigid (retrieval-based matching), or too unstable (LLM-as-a-judge). The results of this work suggest that future evaluation frameworks should explicitly account for persona adaptation and narrative transformation, distinguishing acceptable abstraction from factual distortion.
- **Evaluation connects naturally to user studies.** While automatic metrics capture structural and semantic properties, they fail to assess readability, pedagogical effectiveness, and perceived quality. The strong divergence between metric-based rankings and qualitative judgments motivates the inclusion of human-centered evaluation. This directly supports the user study presented in Part II, which assesses the system from the perspective of its intended audience.

Part II

Web Platform Development: Application Architecture and User Interaction

Chapter 7

Design

Part I introduced the core methodological framework of this thesis: a two-stage pipeline that transforms long scientific papers into persona-adapted narratives through explicit structure extraction and grounded generation. This framework demonstrates that large language models can be guided to produce structured, faithful, and audience-aware scientific stories.

However, a model pipeline alone is not sufficient to improve scientific communication in practice. For such a framework to be useful beyond controlled experiments, it must be made accessible to real users: readers, students, journalists, policymakers, and researchers who wish to understand scientific work without engaging directly with raw papers.

The next question is therefore system-oriented: *how can this framework be exposed through an interactive platform that is usable, transparent, and controllable?* Answering this question requires examining what tools and platforms already exist for transforming scientific papers into explanations or stories, what interaction paradigms they support, and which limitations remain unresolved.

This motivates a comparative analysis of existing AI-based systems for scientific understanding and communication, which in turn justifies the design of the platform developed in this work: *SciTeller*.

7.1 Related Systems: A Benchmarking Perspective

Rather than comparing individual models in isolation, the goal is to identify which parts of the end-to-end scientific communication workflow are currently supported and which remain unresolved.

Although a general overview of related research directions has already been discussed in Chapter 2, the focus here is explicitly system-oriented. We consider concrete tools and platforms available to users and evaluate the extent to which they support accessibility, person adaptation, narrative structure, interaction, and trust.

7.1.1 Landscape of AI Systems for Scientific Content

Current AI systems that support the understanding of scientific content can be divided into four main categories:

- **Question answering and information retrieval systems**, which allow users to search scientific articles and obtain well-founded answers (e.g., ScholarQA¹, Semantic

¹<https://allenai.org/blog/ai2-scholarqa>

Scholar tools²).

- **Summary and explanation assistants**, which generate condensed or paraphrased versions of articles, often through generic LLM interfaces (e.g., Semantic Scholar + GPT, ExplainPaper³, SciSpace⁴).
- **Autonomous or agent-based research systems**, designed to assist or automate parts of the scientific discovery process rather than scientific communication (e.g., Sakana AI Scientist⁵, Google AI Co-Scientist⁶).
- **Data narration and storytelling tools**, which emphasize narration or visualization but are not typically based on full scientific articles or explicit persona adaptation (e.g., Flourish⁷).

Although these systems demonstrate impressive capabilities within their respective scopes, they tend to address isolated aspects of the problem. Most tools focus on retrieval, summarization, or reasoning, without integrating these components into a coherent, user-facing workflow for audience-adapted scientific storytelling.

7.1.2 Benchmarking Criteria

To compare heterogeneous systems in a principled way, we define a set of qualitative benchmarking criteria inspired by human-computer interaction principles and by the requirements identified throughout this project:

- **Accessibility and configuration:** How easy is it for a novice user to start using the system without technical skills?
- **Persona selection:** Does the system allow explanations to be explicitly tailored to different audiences (e.g., students, experts, policymakers)?
- **Output structure and clarity:** Are the generated outputs well structured, readable, and organized into coherent sections?
- **Interactivity:** Can users interact iteratively with the output (e.g., refine sections, ask follow-up questions, control tone or complexity)?
- **Feedback and trust:** Does the system disclose sources, background information, or alignment with the original scientific document?
- **Overall user experience:** A qualitative summary (1-5) capturing usability, clarity, and engagement.

These criteria deliberately go beyond raw model performance, focusing instead on the practical usability and communicative effectiveness of each system.

7.1.3 Comparative Analysis of Existing Systems

Table 7.1 summarizes the qualitative comparison across representative systems.

²<https://www.semanticscholar.org>

³<https://www.explainpaper.com>

⁴<https://typeset.io>

⁵<https://sakana.ai/ai-scientist/>

⁶<https://research.google/blog/accelerating-scientific-breakthroughs-with-an-ai-co-scientist/>

⁷<https://flourish.studio/>

System	Access	Personas	Structure	Interactivity	Sources	UX
ScholarQA (AllenAI)	~	–	–	✓	✓	3
Semantic Scholar + GPT	✓	–	~	–	✓	3
ExplainPaper / SciSpace	✓	–	–	~	~	3
Sakana AI Scientist	–	–	~	~	–	2
Google AI Co-Scientist	–	–	–	~	–	2
Flourish	✓	–	–	✓	✓	3
SciTeller	✓	✓	✓	✓	✓	5

Table 7.1: Qualitative benchmarking of representative AI systems for scientific content. Symbols indicate full support (✓), partial support (~), or absence (–).

Several patterns emerge from this comparison. Question-answering systems excel in factual accuracy and reliability, but provide little narrative structure and no adaptation to personas. Generic summarization tools offer easy access, but reduce complex scientific content to a single, undifferentiated explanation. Autonomous research agents focus on scientific reasoning rather than communication and are largely inaccessible to non-expert users. Data narration and storytelling tools emphasize engagement, but typically lack an explicit foundation in primary scientific sources.

7.1.4 Synthesis and Positioning of SciTeller

Taken together, the benchmark suggests a consistent limitation: existing systems address isolated components of the scientific communication workflow, but do not provide an integrated interaction loop where users can select a target audience persona, obtain a structured narrative, verify provenance against the original paper, and iteratively refine the output at multiple levels of granularity.

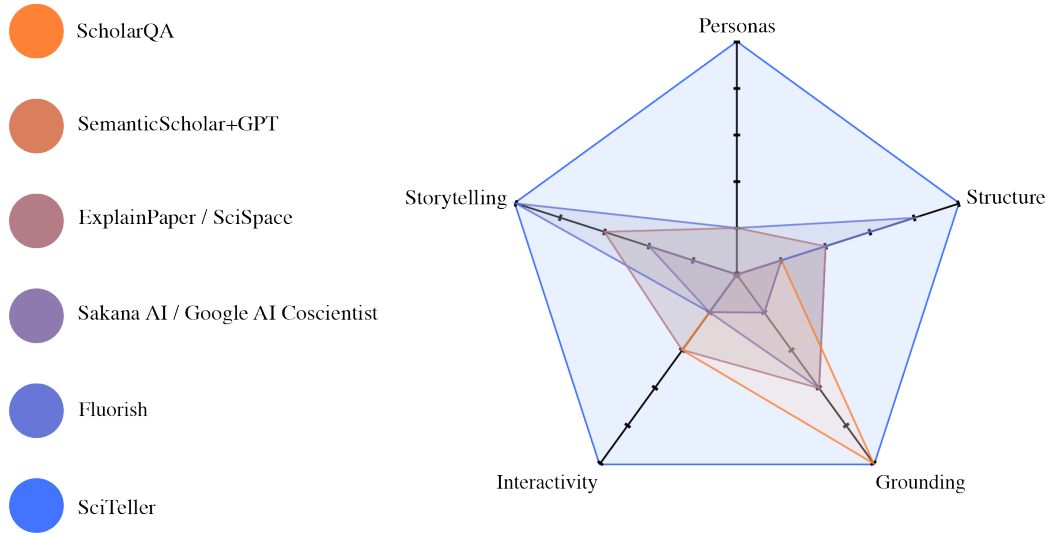


Figure 7.1: Conceptual radar plot comparing SciTeller with existing AI systems across key dimensions of scientific communication.

The platform presented in this work, *SciTeller*, is designed specifically to fill this gap.

7.2 Design Goals

The design of the platform, henceforth called *SciTeller*, was driven by a series of needs regarding how to make this system quickly, easily, and conveniently accessible to users who wish to transform complex scientific articles into structured and accessible narratives. These goals permeated every level of the system, from the user interface and interaction flow to backend orchestration, model serving, and database design.

Usability for non-expert users. A fundamental requirement is that the platform remains usable even by people without prior computer science, natural language processing, or machine learning skills. Uploading a document, selecting a person, and obtaining a first draft narrative must be simple. All the complexity of the system, including PDF parsing, prompt construction, model orchestration, and caching, is intentionally hidden behind a simplified interface. Usability also requires predictable interactions: the interface must always communicate what the system is doing, why a particular step is necessary, and when a process is complete.

Transparency and traceability. Because scientific storytelling involves the interpretation and reformulation of factual content, the system must make the relationship between the generated story and the original document explicit. Transparency is achieved through a double-view architecture that displays the story alongside the PDF, with mechanisms to highlight the origin of a narrative segment within the article. Thanks to text extraction with DocumentParsing, which exposes layout-based metadata (page indexes, bounding boxes, text ranges), this alignment has been easily made possible. Users are therefore able to verify claims, examine evidence, and develop trust in the generation pipeline.

Iterative review and refinement. Scientific narratives rarely converge in a single generation. The platform is therefore designed to support multi-level refinement workflows: users can regenerate an entire story, rewrite specific sections, or experiment with alternative paragraphs written by the model. With each revision, the system retains a complete version of the story and its metadata, enabling branched timelines and reversible edits. Granular review not only enhances the creative process, but also provides a methodological framework for comparing alternative interpretations of the same scientific content and gives the user a sense of control.

Real-time feedback and system responsiveness. Story generation involves computationally expensive operations such as PDF parsing, embedding processing, retrieval, and multi-stage inference on a remote VM where GPUs are located. Without adequate feedback, users may perceive the system as slow or unresponsive. To address this, the platform integrates real-time progress reporting via Server-Sent Events (SSE). The backend normalizes low-level VM tracking messages into readable progress states, while the frontend presents them as structured stages (e.g., "Text Extraction...", "Outline Preparation...", "Section 1 Generation..."). Responsiveness is therefore not just a performance metric, but an explicit design principle aimed at maintaining user engagement and trust.

7.3 Core Features

The *SciTeller* platform provides a set of tightly integrated features that collectively transform a scientific paper into an interpretable, editable, and traceable narrative. These features can be grouped into seven functional pillars: paper ingestion, persona-aware generation, double-view story-PDF visualization, fine-grained refinement, temporal versioning, story analytics and export utilities.

Upload and Automatic Analysis of Scientific Papers

Users can submit a paper via PDF upload or an external link. The system then automatically analyzes the document, including page segmentation, text extraction, layout reconstruction, and removal of non-informative elements such as tables, headings, or references. For text extraction, DocumentParsing is used, as described in section 4.2.1 *Parsing of Scientific Papers (PDF)*, then the text is cleaned and refined, ready to be sent to the VM for the inference phase.

Persona Selection and Persona-Aware Story Generation

Story generation is influenced by a user-selected persona. This persona modifies not only the stylistic features, but also the granularity of the explanations, the density of the technical language, and the narrative structure. The user can choose the persona that best suits them, and each persona is accompanied by comments about tone and context, that allow the user to make a more informed choice and understand the story’s final narrative objective beforehand.

Persona	Tone	Context
General Public	Plain-language	General audience
Investor	Concise & impact-focused	Business / ROI
Student	Simplified academic	Educational use
Journalist	Clear & news-oriented	Media coverage
Policy Maker	Brief & evidence-based	Policy implications
Professor	Didactic & structured	Classroom explanation
Researchers & Engineers	Technical & rigorous	Advanced research & implementation

Table 7.2: Effect of persona on narrative style and informational density.

Dual Viewer: Story–Paper Alignment

One feature of the platform that could be considered distinctive is its dual-view interface, which displays the generated article alongside the original scientific article. The alignment of the two views ensures transparency and verifiability. It works at the section level; for each section, the user can verify where it appears in the article. Once the desired section has been identified, a cosine similarity mechanism identifies the sections of PDF text that best match that section, sorted by highest score.

Story-Level, Section-Level and Paragraph-Level Refinement

A multi-level refinement workflow allows users to regenerate content with different creative settings and lengths.

- **Story-level regeneration:** rewriting the entire story, modifying the character if necessary and controlling creativity and length.
- **Section-level regeneration:** rewriting entire sections using a prompt with user-defined settings.
- **Paragraph-level regeneration:** generation of 1-3 alternatives (number chosen by the user), choosing creativity and length as parameters.

A refinement session does not overwrite previous versions but instead creates a new revision.

Versioning and Branching History

Each modification produces a new *story revision* persisted in the database. Revisions form a branching temporal graph rather than a linear history: in fact, every regenerated section or paragraph is attached to the revision from which it originated.

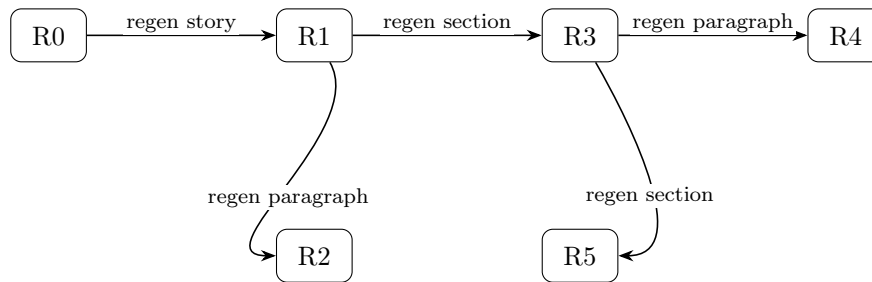


Figure 7.2: Conceptual illustration of revision branching in SciTeller. Nodes represent story revisions; edges represent refinement operations that create new versions without overwriting previous ones.

This example illustrates a possible use of the versioning system explained as follows: start with a newly created story (R0). The user can then regenerate the story with new parameters, which would create version R1. The user could then regenerate a single paragraph, creating R2. At this point, the user may prefer the previous version and therefore return to version R1 to regenerate a single section, creating R3. The user could then choose to regenerate a single paragraph, creating R4, and have one version, or go back to R3 to regenerate the entire section, creating R5, and have a final choice of exporting two stories, R4 and R5.

With this small example, you can better understand the behavior of version control and branch history in a possible use case in Sciteller.

Story Analytics

In addition to generation and refinement functions, the platform offers a dedicated *Info* view that allows users to examine quantitative indicators describing the current status and quality of a specific generated story version. The goal of this component is twofold: to

increase transparency about how the narrative relates to the original article and to support informed refinement decisions during iterative review.

Story analytics are computed on demand for the currently active revision and presented in a read-only dashboard. This avoids continuous recomputation during editing while giving users explicit control over when analytical feedback is required.

The analytics panel first exposes high-level descriptors summarizing the current revision, including:

- the selected persona,
- the number and the names of sections in the story,
- the target length preset (or a *mixed* label when heterogeneous overrides are present),
- the average creativity level aggregated across sections and paragraphs.

These values are derived from a combination of global generation parameters and local overrides applied during section or paragraph-level regeneration.

The analytics view explicitly accounts for partial regeneration events. When a subset of sections or paragraphs has been regenerated, the system records the operation metadata (affected sections, creativity, length preset, timestamp) and exposes a summarized trace in the analytics panel.

A central element of the Info panel is the **StoryScore**, already discussed in Part I. The StoryScore can be calculated on the fly, and after a few seconds its value is displayed along with all its metrics (each provided with its own specific description), allowing users to have quantitative metrics to understand the numerical quality of a specific version and compare it with other versions.

In addition to story-level metrics, the analytics view provides a section-level overview. For each section, the system reports the effective creativity and length after accounting for paragraph-level overrides and recent regeneration events. Sections that deviate from the base story configuration or were recently modified are visually marked, allowing users to quickly identify heterogeneous regions of the narrative.

Story Analytics is not designed as an automatic evaluation gate, but as a decision-support tool integrated into the refinement workflow. By combining revision awareness, aggregate descriptors, and interpretable quality metrics, the analytics panel encourages users to iteratively improve stories while maintaining alignment with the source paper and the intended persona.

This analytical layer complements the qualitative inspection enabled by the dual viewer and versioning system, contributing to a transparent and controllable scientific storytelling process.

Export Options utilities

To support downstream use, from teaching materials to presentations or formal documentation, the platform offers various export formats.

- **PDF:** narrative in printable form.
- **HTML:** embeddable in external web pages.
- **Markdown:** editable, interoperable with scientific writing tools.

Users can export either the current version or any other revision in the history, and choose if to include header and metadata in the export file.

7.4 User Workflow & Interface Components

The *SciTeller* platform is designed around a linear but iterative workflow that guides the user from a raw scientific paper to a refined, persona-targeted narrative. This section summarizes the main steps a typical user follows in the web application and describes the main user-facing components of the platform. Each component is tightly coupled with the backend logic, and together they form a coherent workflow that supports reading, editing, and refining AI-generated scientific narratives.

1. Upload of the Scientific Paper and Persona Selection

The user starts on the home view by uploading a scientific article and selecting a target persona (e.g., *General Public*, *Student*, *Researcher*, *Journalist*) from the interface. The file and the chosen persona are sent to the backend, which performs deduplication based on a SHA-256 hash for storing the PDF files (needed later for the double-side view). The PDF is then parsed into a structured markdown representation using the DocumentParsing tool already discussed, which also preserves layout information (pages and bounding boxes).

The entry point of the application provides a clean, minimal interface that guides the user generating a story, with an easy step-by-step process that starts uploading a scientific paper. The component supports both drag-and-drop and traditional file-selection methods or insert via URL. the next step is the chose of the persona through a menu a tendina that shown seven target-persona (Reserachers & engineers are jointed into a single category due to their very similar request of tone and context).

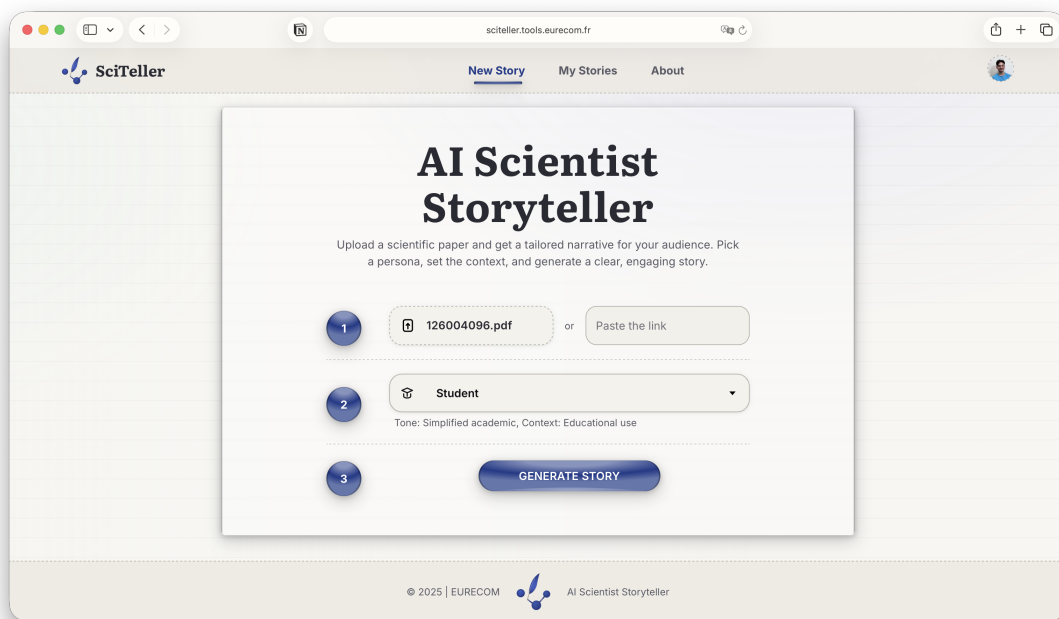


Figure 7.3: Screenshot of the SciTeller home page, which guides the user in generating a new story from the beginning.

2. Waiting Story Generation through the AI Pipeline

Once the PDF has been analyzed and the persona has been set up, they are sent to a remote VM that has a direct connection to the GPUs for the inference phase and becomes part of the prompt configuration for both the Splitter and Storyteller models. The VM has a two-stage pipeline code that orchestrates: first, the Splitter model, which segments the document into an outline of sections adapted to the persona, then the Storyteller model, which generates a narrative section for each element of the schema, also conditioned by the persona. During this process, the frontend displays a loading screen with streaming progress events (e.g., start/end of analysis, schema generation, section generation), providing the user with real-time feedback.

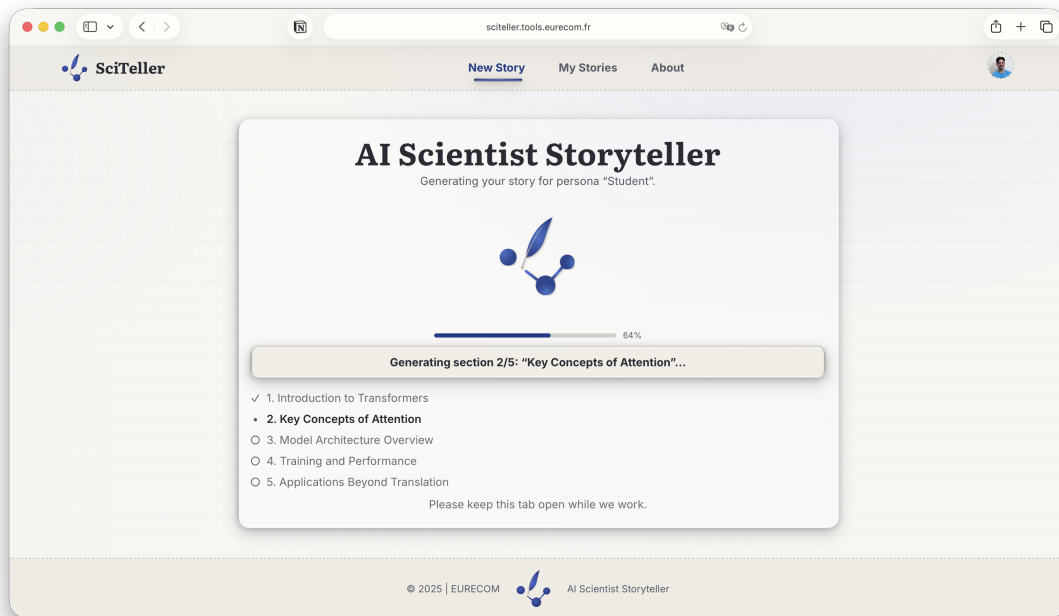


Figure 7.4: Screenshot of the SciTeller Loading page, which shows the user the generation process in real time.

3. Story view and Inspection in the Double-side Viewer

Once the pipeline is complete, the user is redirected to the main reading interface. Here, the generated story is displayed divided into sections, each with its own subtitle. In the reading interface, the user can see a timeline of all the stories generated on the left, the text of the selected story in the center, and the control panel for interacting with the story in question on the right.

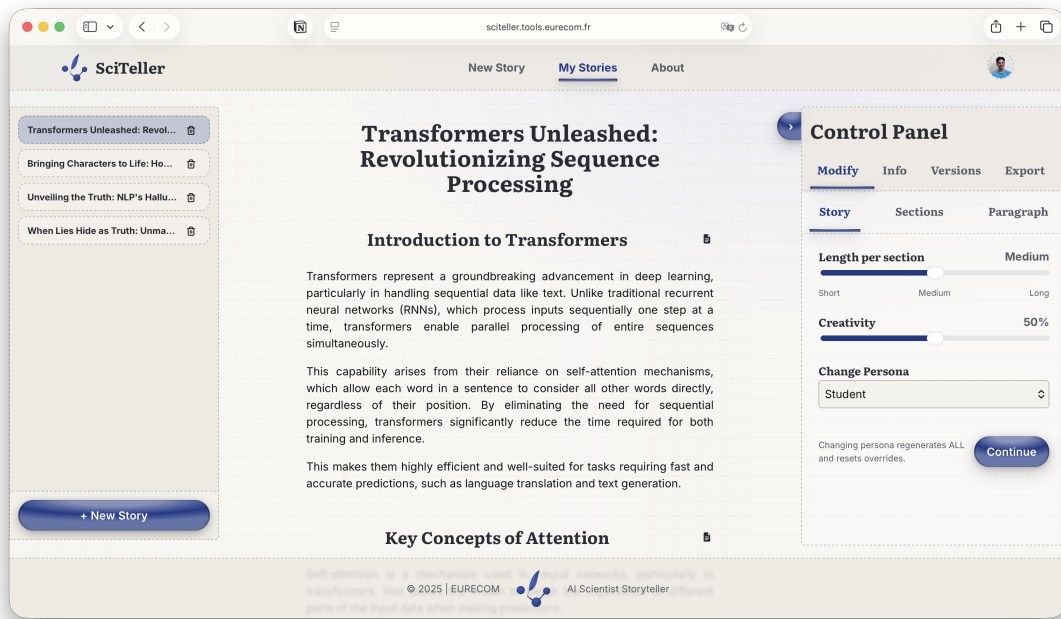


Figure 7.5: Screenshot of the SciTeller reading page, which shows all stories generated and the Control Panel.

Each section also features a paper icon that takes the user to the double-side story-PDF view showing that specific section where it is located in the paper.



Figure 7.6: Screenshot of the SciTeller reading page, which shows all stories generated and the Control Panel.

4. Story-Level, Section-Level, and Paragraph-Level Refinement

From the reader view, the user can refine the generated content at different levels of granularity:

- **Story-level:** regenerating the whole story from the same paper text, possibly with different global parameters.
- **Section-level:** selecting one or more sections and requesting a partial regeneration via the remote GPU service, with updated length or creativity knobs.
- **Paragraph-level:** choosing a specific paragraph, generating multiple alternatives, and interactively selecting the preferred variant.

Each refinement request is sent to dedicated backend endpoints that call specialized VM routes, and the resulting sections or paragraphs are merged back into a new coherent story state.

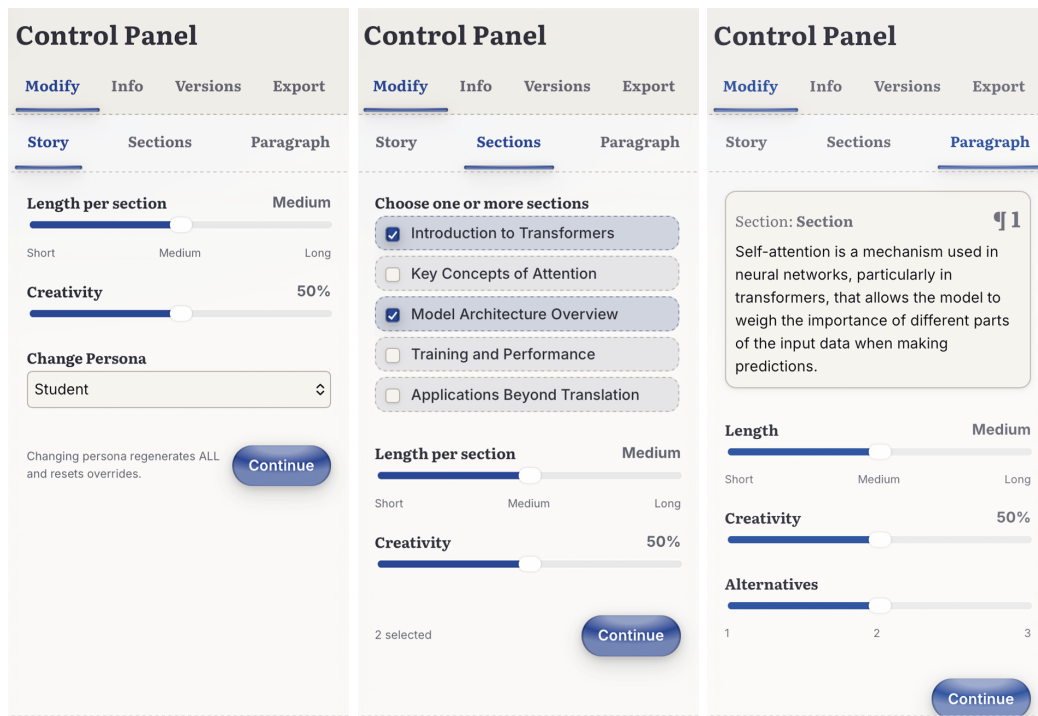


Figure 7.7: Screenshot of the SciTeller Modify tabs in Control Panel

Paragraph-level regeneration, in particular, supports a comparative refinement workflow by allowing the model to generate multiple alternative rewrites for a selected paragraph in a single operation. When the user triggers paragraph regeneration, the system produces a small set of candidate variants (one to three, number chosen by the user), each reflecting the same semantic content but differing in phrasing, level of detail, or stylistic choices according to the selected creativity and length parameters.

These alternatives are presented to the user through a dedicated carousel-style interface, which enables sequential inspection of each candidate without immediately modifying the story. The user can read all proposed variants, compare them side by side with the original paragraph, and explicitly select the preferred version to apply to the narrative. Only the chosen and selected alternative (via the check mark icon) is then integrated into the

story, creating a new revision and implicitly preserving all unselected variants as discarded candidates, which can always be revisualized, reread, and reselected by clicking on the paragraph and reopening the carousel with the alternatives.

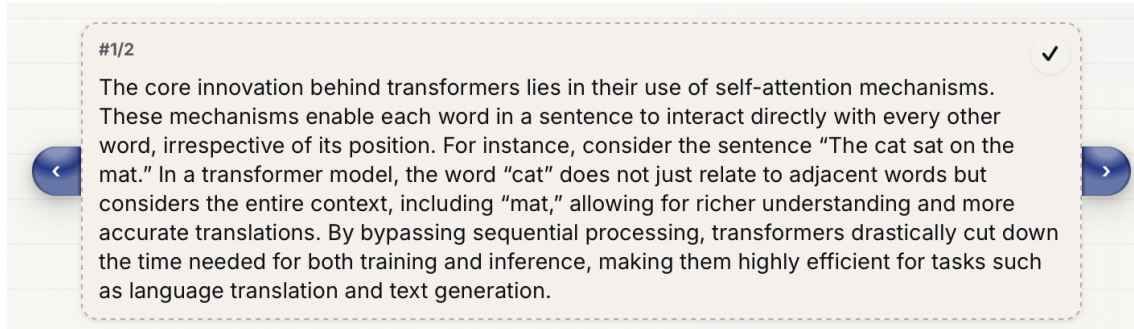


Figure 7.8: Paragraph-level refinement interface. After regeneration, multiple alternative rewrites are presented in a carousel, allowing the user to inspect each variant and explicitly select the preferred paragraph to apply to the story.

5. Info Story (Analytics and Inspection)

The Info panel presents a structured summary of the active revision, including the story title, selected persona, number of sections, effective length configuration, and aggregated creativity level. These values reflect the actual state of the story after accounting for section (and paragraph) level overrides, providing users with a concise overview of how the narrative has evolved relative to its initial generation parameters.

A key component of the Info view is the on-demand computation of the **StoryScore**. When triggered by the user, the backend evaluates the current revision and returns a composite quality indicator together with its individual components, such as semantic alignment with the source paper, contextual recall, title consistency, redundancy penalties, and hallucination checks. Each metric is displayed alongside a short explanation, allowing users to interpret numerical values without requiring expertise in automatic evaluation methods.

In addition to story-level indicators, the Info view offers a section-level inspection mode. For each section, the system reports the effective creativity and length, also after propagating paragraph-level settings and recent regeneration operations. Sections that diverge from the base configuration or were modified during recent partial regenerations are visually marked, enabling users to quickly identify heterogeneous regions of the narrative.

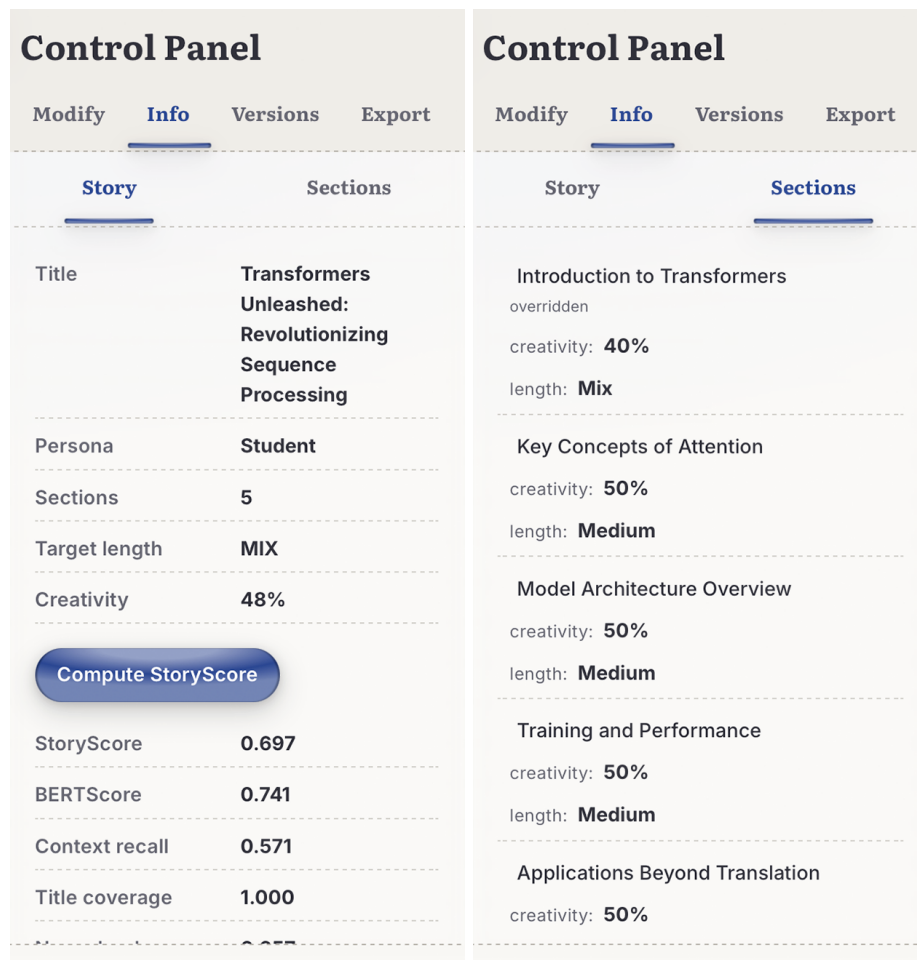


Figure 7.9: Screenshot of the SciTeller Info view, showing story-level descriptors, StoryScore metrics, and section-level analytics for the active revision.

6. History Review and Revision Comparison

SciTeller incorporates an explicit revision management system that records every modification applied to a story. Rather than overwriting previous content, each regeneration operation (whether at the story, section, or paragraph level) produces a new *story revision* stored persistently in the database.

The History view exposes this revision structure to the user through a graphical interface that represents revisions as nodes in a branching graph. Each node corresponds to a distinct version of the story and is annotated with metadata such as creation time, applied operation, and associated parameters. Edges represent refinement actions, making it possible to visually trace how different versions originate from a common ancestor.

This non-linear revision model supports exploratory workflows. Users may branch from earlier revisions, compare alternative refinements, revert to a previous state, or designate a preferred version as the default without losing intermediate results. The History view thus encourages controlled experimentation while preserving full traceability of editorial decisions.

By making version history explicit and navigable, the platform bridges the gap between creative writing workflows and systematic version control principles, adapted to the needs of non-technical users.

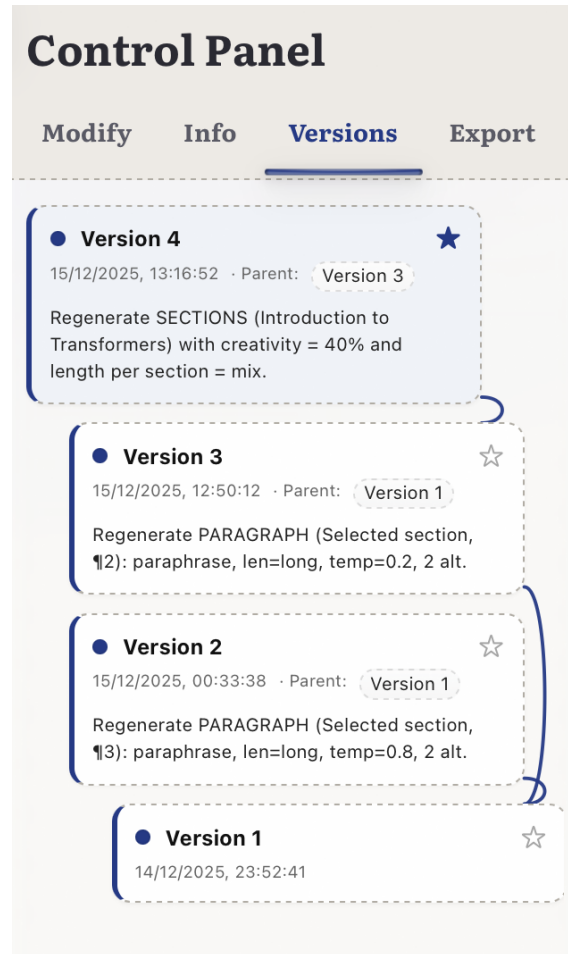


Figure 7.10: Screenshot of the History view, showing the revision graph and metadata associated with each story version.

7. Export of the Final Story

Once a satisfactory version of the story has been reached, SciTeller provides export functionalities that transform the active revision into a portable artifact suitable for downstream use. Supported formats include **HTML**, **PDF**, and **Markdown**, covering common scenarios such as web publication, printable documents, and further editing in external tools.

The export interface allows users to configure several options prior to generation. These include selecting the output format, choosing which sections of the story to include, and deciding whether to embed auxiliary information such as headers (title, persona, version) or metadata (identifiers and timestamps). Export operations are performed client-side and do not modify the underlying story or its revision history.

The exported artifact always reflects the currently active revision, ensuring consistency between the on-screen narrative and the final output. This design guarantees that persona adaptation, refinement choices, and accepted paragraph variants are faithfully preserved in the exported document.

By supporting flexible export configurations while maintaining revision integrity, the platform enables SciTeller-generated stories to be reused in educational, professional, and communicative contexts without additional post-processing.

Control Panel

[Modify](#)
[Info](#)
[Versions](#)
[Export](#)

OPTIONS

Format HTML

HTML (.html) ▾

Header Included

☒ Include title, persona, date, version

Metadata Included

☒ Include id, persona, version, timestamps

Export is client-side only and does not modify your story.

SECTIONS TO EXPORT

☒ Introduction to Transformers

☒ Key Concepts of Attention

☒ Model Architecture Overview

☒ Training and Performance

☒ Applications Beyond Translation

Figure 7.11: Screenshot of the Export view, allowing users to select format options, included sections, and metadata before generating the final output.

7.5 Design Rationale

This section explains the reasons behind SciTeller’s key design decisions. While the previous sections described *what* the platform offers (features and workflow), here the focus is on *why* these choices were made, what alternatives were considered, and how the final design addresses recurring failure modes in LLM-based scientific communication: instability between regenerations, loss of document structure, limited traceability, and difficulty for non-expert users to manage the output.

From one-shot generation to an iterative editorial workflow

A natural baseline for scientific explanation is a one-shot generation interface where the user upload a paper, ask for a summary and obtain a single output. In practice, this interaction pattern is poorly suited to long scientific papers and persona adaptation. Even

when an initial output is acceptable, small prompt or parameter changes often lead to global rewrites that overwrite good content, making improvements non-monotonic and discouraging exploration.

SciTeller therefore treats storytelling as an *editorial process*. Instead of forcing users to restart the entire generation every time they want to make a small change, the platform supports refinement at multiple levels of granularity (story, section, paragraph) and introduces explicit versioning. This redefines the model’s output as a draft that can be reviewed and improved iteratively, aligning the system with real-world writing workflows.

Why explicit structure extraction is a control mechanism

Long documents pose a structural challenge: relevant information is spread across many pages, and LLMs can drift, compress sections, or overemphasize salient but secondary details. A purely generative approach tends to intertwine two tasks: *deciding what the story should cover* and *writing it*, which makes the result more difficult to control and correct.

SciTeller separates these roles through a two-stage design. Thanks to the Splitter, which produces an explicit outline, it provides a stable reference for downstream refinement and evaluation. In terms of the system, this outline functions as a controllable intermediate representation because it improves reproducibility between executions, enables targeted regeneration (only the affected schema element), and makes it easier to locate faults (e.g., an omitted contribution becomes a schema issue rather than a vague “bad story”).

Persona as a first-class configuration, not a stylistic afterthought

Many existing tools implicitly approximate adaptation to the audience through generic instructions (“keep it simple,” “explain as if you were a student”). In practice, adaptation to the person affects not only style, but it changes which concepts need to be introduced, what background can be assumed, and what framing is appropriate (e.g., policy implications vs. technical details for engineers).

For this reason, SciTeller treats person selection as a first-class configuration propagated throughout the pipeline and stored in revision metadata. This ensures that persona consistency is maintained during refinement and that users can attribute changes in tone or content to explicit persona choice or subsequent regeneration decisions.

Paragraph refinement without destructive edits

A key risk to usability in LLM-assisted writing is destructive editing: content regeneration can remove correct statements or introduce new errors, leaving the user uncertain about what has changed and why. SciTeller addresses this issue by allowing regeneration at the smallest meaningful unit (paragraph) and avoiding immediate overwriting, as written before.

In paragraph-level refinement, the system generates a small set of alternatives and presents them in a carousel. This design supports comparative decision-making: users can review candidates before applying any changes. It also encourages “human selection” rather than blind acceptance, which is critical for scientific content where fidelity is important.

At the system level, this choice reduces regret: users can restore, branch, and compare, rather than losing the previous text. The goal is not to maximize automation, but to maximize controllability.

Why branching revision history is preferable to linear undo/redo

Traditional interfaces are based on a linear undo/redo system, which assumes that changes form a single timeline. Scientific storytelling, however, often involves exploring alternative frameworks that are all plausible: the “best” story depends on the target audience, the level of detail, and the emphasis (methods vs. impact). Linear chronology therefore forces users to choose prematurely between alternatives. SciTeller models revisions as a branching graph. Each regeneration produces a new node connected to its parent revision, allowing multiple exploration paths to coexist. This mirrors the ideas of lightweight version control, while remaining usable for non-technical users: the interface visually exposes versions and allows you to switch between them or set a default version without requiring advanced concepts such as merges or conflict resolution.

Traceability through alignment rather than blind trust

A persistent problem in scientific texts generated by LLMs is the inability to verify them: users may receive fluent explanations without knowing which parts are based on the article and which are inventions or excessive generalizations. SciTeller addresses this problem through an explicit story-article alignment interface that allows users to map narrative sections to candidate regions in the PDF.

This choice is motivated by the principle that trust must be verifiable. Instead of asking the user to trust a black-box generation, the platform provides a practical mechanism for validating claims and navigating between the story and the source. The alignment view is therefore not just a visualization feature, but a fundamental component for risk mitigation in scientific communication.

Analytics as decision support, not as an automatic gate

Automatic metrics can provide useful signals, but they are not reliable enough to serve as strict acceptance/rejection criteria, especially when the goal is a personalized narrative rather than a rigorous summary. SciTeller therefore calculates the StoryScore on demand and presents the user with both the overall score and its components. In short, the analyses are integrated as diagnostic tools that support editorial control rather than replace it.

System responsiveness and perceived reliability

Finally, an end-to-end pipeline for long documents includes latency resulting from parsing, structuring, and multi-stage generation on remote GPUs. Without progress feedback, users may interpret the wait time as an error and abandon the process. SciTeller mitigates this issue through real-time progress reporting (SSE), displaying pipeline stages in a user-readable manner. This design choice is not only an improvement to the user experience: it increases perceived reliability and reduces confusion in cases where generation is slow but proceeding correctly. It also supports debugging and reporting by the user by making the system state explicit during execution.

Overall, the platform design is driven by a consistent principle: *scientific storytelling is not a single-shot generation task, but an iterative communication process*. SciTeller operationalizes this principle through explicit structure, persona-aware control, non-destructive refinement, revision branching, inspectable traceability, and analytics that guide rather than constrain user decisions.

Chapter 8

UI and Implementation

8.1 Overall Architecture

The platform is designed as a multi-tier web architecture that clearly separates user interaction, application logic, complex AI computations, and persistent storage. This separation improves modularity, scalability, and maintainability, while allowing computationally intensive workloads to be isolated on dedicated GPU infrastructure.

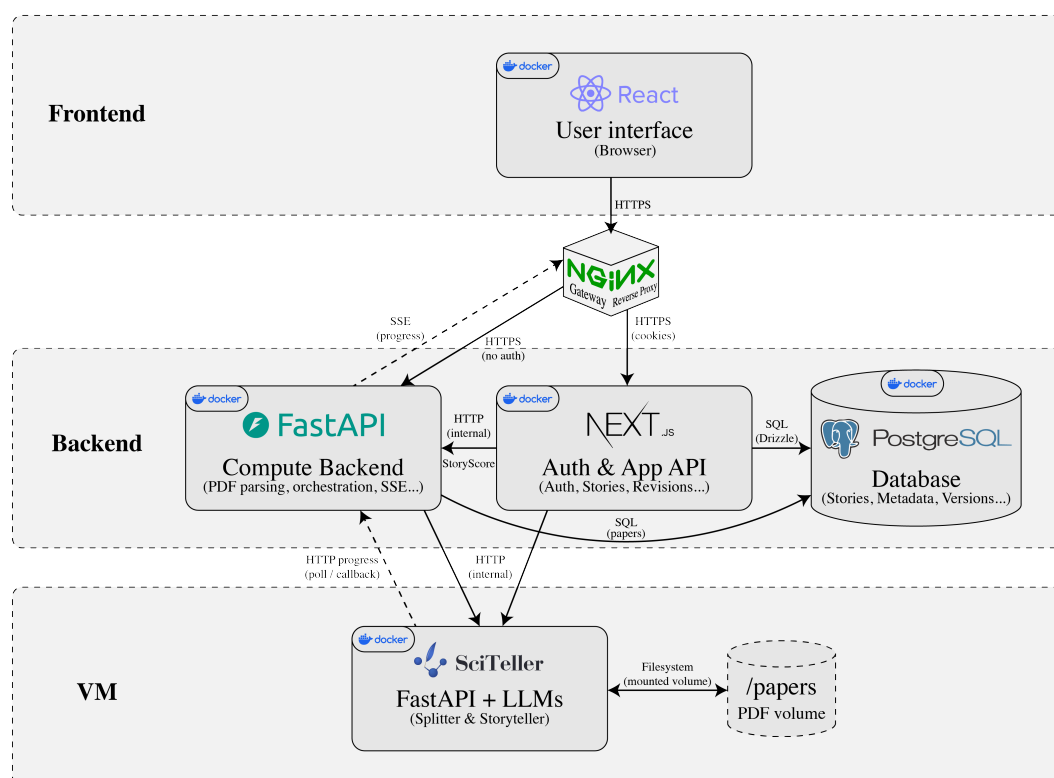


Figure 8.1: SciTeller overall architecture

Figure 8.1 provides an overview of the system and its main components. At the top

level, the frontend consists of a web-based user interface implemented as a single-page application, which runs in the user’s browser and is responsible for interaction, visualization, and user-driven control of the storytelling process.

All external requests from the frontend are routed through a gateway and reverse proxy component, which acts as the single point of access to backend services. This layer is responsible for request routing and protocol handling, and enables a clear separation between public endpoints and internal services.

The backend layer is divided into two complementary services. The first is an application API that handles authentication, user management, and high-level application logic related to stories, revisions, and metadata. The second is a computation-oriented backend that focuses on orchestration tasks, including PDF capture, document preprocessing, progress streaming, and coordination of long-running AI jobs. Both services interact with a shared relational database, which stores users, stories, revisions, and associated metadata.

To support the inference of computationally intensive language models, the architecture includes an isolated virtual machine equipped with GPU resources. This VM hosts a dedicated inference service that runs the main AI pipeline (Splitter and Storyteller) as well as any regenerations (of stories, sections, and paragraphs). By isolating this component, the system prevents heavy inference workloads from interfering with the responsiveness and stability of the main application backend.

Uploaded PDF files are stored as regular files on persistent host storage and mounted into the services that need them (inference container and internal backends) via bind mounts / Docker volumes. This avoids storing large binaries in PostgreSQL while keeping document access efficient for parsing and “read on paper” workflows. Communication between backend services and the inference VM occurs via internal HTTP interfaces, while progress is streamed to the frontend via SSE.

Taken together, this architecture enables a clear separation of responsibilities: the frontend focuses on interaction, the backend services handle orchestration and state, while the GPU-based inference layer handles the artificial intelligence calculations. This design allows each component to evolve independently and supports future extensions such as horizontal scaling, model replacement, or deployment in distributed environments.

8.2 Frontend Implementation

The frontend is a client-side web application built with React and React Router. Its main role is to guide the user through the loading of documents and the selection of personas for story generation, then display the generated stories and their revision history, and also provide detailed editing/regeneration controls, as well as supporting a “read on paper” workflow where the story content is aligned with the original PDF.

8.2.1 Routing and main pages

The application is organized around three primary routes:

Home (/). The home page (`Home.jsx`) is the access point for creating stories. The user interface follows a simple three-step flow:

1. provide a document (local PDF upload or link),
2. select a persona,
3. start generation.

Link-based inputs are downloaded via a backend proxy endpoint to avoid browser-side CORS constraints; the resulting blob is wrapped in a `File` object so that the subsequent pipeline is identical to local uploads. When generation begins, the page creates a backend job ID and attaches an SSE stream that emits structured progress events. These events drive a dedicated loading screen that separates the extraction/analysis phase from the story generation phase, also showing section-level progress as titles become available.

Stories workspace (/stories). The stories page (`Stories.jsx`) is the main workspace. It displays a left sidebar with the list of stories (create/delete/select), a central area for reading and interacting with the story, and a control panel on the right side for regeneration and editing. The page maintains the active story ID and the active revision ID. This allows the user interface to support multiple editing scopes: regeneration of the entire story, regeneration of the selected section, and paragraph-level rewriting with candidate variants. To keep the experience responsive, the page uses explicit “busy” flags for sections/paragraphs and temporary loading screens for long operations.

Reader (/reader). The reader page (`Reader.jsx`) is opened from the stories workspace via a navigation state payload. It displays the section of the story chosen by the user on the right and the PDF with the corresponding highlighted sections on the left. The navigation state includes the PDF URL, a starting page number, and one or more highlighted rectangles. The highlighted elements represent the best match from the backend (multiple optional alternatives) for the section of the story selected in the original document.

8.2.2 Progress feedback via SSE (generation pipeline)

While creating the story, the frontend provides real-time feedback on progress using server-sent events (SSE). The home page opens an SSE connection to a backend log endpoint using the job ID created at the beginning of the request. The user interface listens for events such as `parsing_start`, `parsing_done`, `splitter_progress` (structure titles as they are produced), `story_progress` (current section being generated), and `all_done`. These events update:

- the current phase,
- a history of section titles as they become known,
- a progress bar calculated not as elapsed time but as a fraction of milestones completed.

This approach provides stable feedback to the user even when generation latency varies depending on documents.

8.2.3 Stories workspace: state model and interactions

The stories page is implemented as a workspace with a status that combines reading and editing. The main status includes: the list of stories, the ID of the selected story, the ID of the active revision, the optional selection of the current section, and an optional selection of the paragraph (section ID + paragraph index + text). When the user clicks on a paragraph in `StoryView`, the page sets `selectedParagraph` and moves the interface into paragraph editing mode. When the user changes revisions, the page updates the active revision ID and resets the transient UI elements (selection and variant carousel) to avoid mixing content between revisions.

The page supports three main regeneration flows:

- **Complete regeneration of a story from source text:** invoked when the user requests a complete rewrite (persona and generation checks are applied, using the already stored `paperText`).
- **Partial regeneration of selected sections:** called by passing a list of section identifiers to the backend along with the current revision ID (used as the basis for versioning).
- **Paragraph-level regeneration with variants:** called for a single paragraph, returns multiple candidate rewrites. The variants are shown to the user and can be adopted either as a local preview (overwriting the user interface) or saved in the backend as a new revision.

In all these flows, the user interface explicitly tracks “busy” elements (sections/paragraphs) to show local spinners and prevent conflicting changes.

8.2.4 ControlPanel and paragraph variants

A key component of the frontend is the `ControlPanel`, which concentrates all editing controls in a dedicated side panel rather than distributing actions throughout the body of the article. The panel operates in internal stages (tracked by `cpStage`), allowing the same panel to support article-level controls, section-focused operations, and paragraph-level rewriting. When a paragraph regeneration request is made, the frontend updates the history of variants for the selected paragraph and revision, then displays the latest batch as a carousel of candidate texts. The user can either preview the candidates inline without confirming the changes, or commit a chosen candidate via a dedicated API call that updates the article and advances the current revision.

In addition, the control panel combines information processed by the backend with lightweight summaries on the client side. On the one hand, complex calculations (e.g., `StoryScore` and its sub-metrics) are performed on the backend and exposed via a dedicated API endpoint, which the panel activates on demand and then displays. On the other hand, the dashboard also derives “presentation-level” indicators directly in the frontend, such as actual creativity and length labels shown for the story or individual sections, aggregating parameters stored in the story payload (e.g., `meta.upstreamParams`, section substitutions, and paragraph-level knobs, when available). When aggregates provided by the backend are present (e.g., `meta.currentAggregates`), they are displayed directly; otherwise, the frontend recalculates equivalent lightweight aggregates as a fallback, ensuring that the user interface remains consistent even if pre-calculated summaries are missing.

This design provides a clear separation between *reading* (central canvas) and *actions* (side panel), keeping the state of the interaction explicit and minimizing accidental changes.

8.2.5 “Read on paper” workflow

The frontend supports an alignment workflow that links story sections back to the original paper. From the stories page, the user can request to “read on paper” for a selected section. The frontend sends the section title and text (plus a paper identifier) to a backend endpoint that returns a best match location in the PDF (page and normalized bounding box), optionally with alternative matches. The application then navigates to `/reader`, passing the PDF URL and highlights through navigation state. In the reader view, the PDF renderer highlights the matched region(s) and displays the corresponding story section

alongside the document, enabling a quick grounding check for the narrative against the source.

8.3 Backend Implementation (FastAPI)

The platform’s backend, which handles orchestration, is implemented as a Python-based service using the FastAPI framework. Its main role is to act as an orchestration layer that coordinates document processing, story generation workflows, and evaluation activities, while remaining independent of authentication and user interface logic issues, which are handled by the second backend component (the JavaScript component explained below).

It is important to note that this backend does not directly orchestrate the internal AI generation pipeline. The detailed coordination between the Splitter and Storyteller models is performed inside a dedicated GPU-based inference service. The FastAPI backend instead acts as an intermediary coordination layer, responsible for preparing inputs, forwarding requests to the GPU service, and collecting the generated outputs.

This separation allows the backend to focus on system components that require high computing power and are specific to Python.

This backend block, which will henceforth be referred to simply as the backend, is responsible for performing tasks that require direct access to Python libraries and machine learning tools, such as document analysis, text preprocessing, and evaluation metric calculation. Unlike the *authentication and API* layer, which is implemented in JavaScript, the backend leverages the Python ecosystem to integrate natural language processing utilities, embedding models, and evaluation frameworks. It functions as an internal service and is not directly exposed to end users.

The backend communicates with the authentication and API layer via HTTP requests and operates behind the same network boundary.

Document parsing and VM calls Uploaded scientific articles are processed by the backend through a dedicated document parsing pipeline, described in previous chapters. PDF files are retrieved via an internal proxy from the GPU VM file store. This includes cleaning up artifacts introduced by PDF extraction, normalizing formatting, and producing a consistent representation of the article similar to markdown. By centralizing document analysis in the backend, the system ensures consistent pre-processing in all subsequent stages of the pipeline.

It then sends the cleaned text to the remote GPU-based inference service. It interacts with the GPU inference service through well-defined API calls, without maintaining persistent state about users or sessions. Long-running operations are decoupled from the user interface and monitored asynchronously through progress streams, enabling the system to remain responsive without blocking frontend interactions. In this interaction, the backend remains independent of the internal execution logic of the GPU service.

Evaluation and StoryScore computation. In addition to generation workflows, the backend implements the evaluation logic described in Chapter 6. The evaluation endpoints are invoked on demand by the frontend through the authentication and API layer, allowing quality metrics to be computed independently from the generation process and without requiring access to GPU resources.

The calculation of the composite metric StoryScore is based on Python libraries, such as sentence embedding models and lexical similarity measures. For this reason, this functionality is encapsulated in a dedicated module and exposed through a backend endpoint

that can be called from the authentication and API layer described below. The backend communicates with the authentication and API layer via HTTP requests and operates behind the same network boundary.

Placing this component in the backend, rather than in JavaScript-based services, allows for efficient reuse of existing NLP tools and simplifies maintenance.

Design considerations. The choice to use FastAPI provides a lightweight but expressive framework for defining typed endpoints, managing concurrency, and integrating with Python-based machine learning components. By isolating the orchestration and evaluation logic in a dedicated backend service, the system achieves a clear separation of responsibilities: user interactions and access control are handled elsewhere, while computational workflows remain centralized and extensible. This design improves scalability and facilitates future extensions of the storytelling pipeline and evaluation methodology.

This separation also makes the system easier to reason about: the FastAPI backend coordinates data and services, while the GPU-based service encapsulates the complexity of AI model orchestration. As a result, changes to the generation pipeline can be deployed independently from the core backend logic.

8.3.1 Reverse proxy and request routing

All external HTTP traffic to the platform is handled by an Nginx reverse proxy, which acts as the single entry point for the web application. This component is responsible for request routing, payload size control, timeout management, and isolation of internal services.

Incoming requests are routed to different backend components based on URL prefixes. Requests under the `/api` path are forwarded to the Next.js API layer, which handles authentication, session management, and application-level logic. Requests under the `/svc` path are forwarded to the FastAPI orchestration backend, which manages document processing, progress streaming, and coordination of long-running AI jobs. Static frontend assets are served directly by Nginx, while client-side routing is supported through a fallback mechanism that always returns the main application entry point.

The reverse proxy is explicitly configured to support Server-Sent Events (SSE), which are used to stream progress updates for long-running operations such as document parsing and story generation. For these endpoints, response buffering is disabled and extended read timeouts are enabled to allow continuous event streaming over a single HTTP connection.

Given the need to upload large PDF files and handle long-running inference tasks, the proxy is configured with an increased maximum request body size and extended timeout values. By centralizing request routing in this layer, the platform avoids exposing internal service addresses directly to the browser and maintains a clear separation between public endpoints and internal services.

8.4 Backend Implementation (Next.js API Layer)

The platform includes a second backend component implemented in JavaScript using Next.js. While the FastAPI service focuses on orchestration, document processing, and metric calculation, the Next.js layer acts as an application-facing API gateway: it exposes the HTTP endpoints used by the React frontend, enforces authentication and access control, and provides a stable interface for CRUD operations on stories, revisions, and document metadata. In practice, this layer is the “entry point” for the web application,

while the internal services (FastAPI and the GPU VM) remain behind the same network boundary.

The Next.js API layer handles all issues naturally related to the backend of a web application:

- session management and authentication (login/logout, token- or cookie-based sessions),
- authorization checks before any operation on articles, revisions, or documents,
- database access for persistent entities (articles, revisions, document records, and related metadata),
- request validation, error normalization, and response modeling for the frontend.

This design keeps the Python backend independent of user identity and web issues and avoids exposing internal services directly to the browser.

From the frontend perspective, the Next.js backend offers a set of REST-like endpoints grouped by domain:

- **Stories and revisions.** Endpoints implement CRUD for stories (create, list, update, delete) and handle version control by creating new revisions for operations that modify generated content (e.g., full regeneration, partial regeneration, adoption of paragraph variants). The API returns consistent story objects, including revision identifiers and timestamps, allowing the user interface to maintain consistent state and render the version graph.
- **Document management.** Endpoints register uploaded PDFs or external links, store document-level metadata, and provide controlled access to the PDF resource (e.g., a secure URL or proxy path). In this way, the “paperId $\rightarrow$ pdfUrl” layer is resolved securely and authenticated.
- **Generation orchestration (proxy).** Requests that trigger expensive processing are forwarded to internal services. The Next.js layer does not execute the pipeline itself: it validates the request, checks permissions, then calls the FastAPI orchestration backend (which in turn interacts with the GPU VM where the two-stage inference code is executed). Responses are remapped to the frontend schema so that the UI state can be updated immediately.

Authentication, authorization, and policy enforcement Authentication on the platform is implemented using the NextAuth framework and is handled entirely by the Next.js API layer. The system supports OAuth-based access via external identity providers and does not require local password management.

Currently, two authentication providers are enabled: Google and GitHub. Both providers are configured using standard OAuth 2.0 flows and are exposed to the frontend via NextAuth API endpoints.

The login process is initiated from a dedicated frontend page, which presents the user with the available identity providers. When a provider is selected, the frontend sends a POST request to the corresponding NextAuth login endpoint, including a CSRF token and a callback URL. The user is then redirected to the external provider for authentication.

After successful authentication, the provider redirects the user back to the application, where the Next.js backend validates the response, creates or updates the user record if

necessary, and establishes an authenticated session. Upon completion of the process, the user is redirected to the main stories workspace.

Session management is handled using JSON Web Tokens (JWT). After successfully logging in, the session token is stored on the client side and automatically attached to subsequent requests. The backend extracts the user's identity from the session token and uses it to enforce access control on all protected endpoints.

The user identifier derived from the authentication token is propagated to the application layer and used consistently to associate stories, reviews, and other user-generated resources with their respective owners.

User and session data is stored in PostgreSQL using the Drizzle adapter provided by NextAuth. This adapter maps authentication-related entities, such as users, accounts, sessions, and verification tokens, to relational tables in the database.

Although these tables are logically linked, referential integrity is enforced at the application level rather than through explicit foreign key constraints in the database schema. This design choice simplifies schema evolution and keeps authentication concerns isolated within the Next.js API layer.

Authorization checks are centralized in the Next.js backend. Before performing any operation that accesses or modifies application data, the API verifies that the authenticated user is authorized to act on the requested resource. For example, story ownership is validated before allowing read, update, or delete operations.

By consolidating authentication and authorization logic into a single backend component, the system avoids duplication of security logic and prevents direct exposure of internal services to unauthenticated clients.

Database access and transactional updates The Next.js layer is responsible for reading and writing persistent state to PostgreSQL. Operations that modify story content typically follow a transactional pattern:

1. read the current state of the story/revision,
2. trigger a generation/evaluation process in the internal backend when necessary,
3. write a new revision (or update metadata) and set the appropriate pointers (e.g., `current_revision_id`, `defaultVersionId` semantics used by the user interface).

This approach ensures that the frontend can treat each update as an atomic state transition and that the version history remains verifiable.

File and PDF access patterns PDF management passes through the Next.js backend and is not done directly by the browser. This serves three very practical purposes:

- **Protection** The backend provides a controlled path for downloading or viewing PDFs stored as files on the GPU VM file store, instead of exposing them directly.
- **No CORS issues** If the PDF comes from an external site (such as arXiv), the backend acts as an intermediary (proxy) and retrieves it, avoiding browser blocks.
- **Internal consistency** The backend assigns and maintains consistent identifiers for each document (`paperId`, `paperUrl`, extracted text), so that all parts of the system refer to the same PDF in the same way.

This is also where practical deployment concerns are handled, such as stable URLs behind a reverse proxy and consistent path prefixes used by the frontend (`/svc`).

Error handling and API normalization Finally, the Next.js API layer provides a consistent error contract for the frontend. Internal errors from orchestration services are detected and normalized into predictable HTTP status codes and messages. This is essential for the user experience, allowing the React application to show meaningful feedback (e.g., PDF rendering/proxy issues, regeneration errors, unavailable resources) without exposing internal stack traces or service topology.

Overall, the implementation of this layer in Next.js provides a cohesive “web backend” component: it integrates naturally with the frontend routing model, simplifies authentication and database integration, and separates user-facing APIs from internal Python and GPU-based services that perform heavy computation.

8.5 Database Architecture

The persistent state of the application is stored in a PostgreSQL relational database, which is accessed exclusively by the Next.js API layer. The database is not directly accessible from the FastAPI orchestration backend or the GPU inference service; instead, all read and write operations are mediated by authenticated APIs at the application level.

PostgreSQL stores only metadata and logical references (e.g., `paperId`, `file_path`, `url`); PDF binaries are stored as files on the GPU VM filesystem (bind-mounted into the inference container).

The schema is designed to support document management, story versioning, granular editing operations, and controlled sharing, while keeping the database layer independent from long-running AI computation workflows.

Although the database schema encodes clear relationships between entities (e.g., stories, revisions, users, and documents), these relationships are enforced at the application level rather than through explicit foreign key constraints in PostgreSQL.

The `users` table 8.5.1 stores authenticated users created via OAuth-based access providers. Each user is uniquely identified and linked to the content generated via identifiers managed by the application.

Scientific papers are represented in the `papers` table 8.5.1. Each record corresponds to an uploaded PDF file or an external link. In the case of file uploads, the physical PDF is stored on the GPU virtual machine’s file system, while the database stores the logical identifier, source type, and file path or URL. This separation avoids the presence of large binary objects in the database, while preserving a consistent reference level.

The generated narratives are modeled using a two-level structure consisting of `stories` 8.5.1 and `story_revisions` 8.5.1. A story represents a logical narrative entity owned by a user, while each revision captures a concrete version of the generated content.

Each revision stores the complete structured output (sections, paragraphs, metadata) as JSON, along with the persona, generation parameters, and optional change summaries. Revisions form a version graph through an optional parent revision identifier, enabling full regeneration, partial regeneration, and rollback-like behaviors without overwriting previous content.

Detailed modification operations are supported through paragraph-level variant tracking. When a user requests the regeneration of a paragraph, a new entry is created in the `paragraph_variant_batches` table 8.5.1, which records the target story, revision, section, paragraph index, and applied transformation operations.

Individual candidate rewrites are stored in the `paragraph_variants` table 8.5.1. Each variant is classified and can be optionally applied to produce a new revision of the story.

This design allows multiple alternatives to be explored and compared before confirming changes to the persistent revision history.

Story sharing is implemented through the `story_shares` table 8.5.1, which enables token-based access to stories. Each share record defines a unique access token, optional edit permissions, and an expiration timestamp.

This mechanism allows stories to be shared externally without exposing user credentials or internal identifiers, while still enforcing access policies through the application layer.

8.5.1 Database schema overview

For completeness, here there are the main database tables used by the application, together with their columns.

Users

Column	Type
id	text
name	text
email	text
email_verified	timestamp with time zone
image	text
created_at	timestamp with time zone
updated_at	timestamp with time zone

Papers

Column	Type
id	text
source_type	text
url	text
file_path	text
sha256	text
doi	text
title	text
first_author	text
title_first_norm	text
created_at	timestamp with time zone

Stories

Column	Type
id	text
user_id	text
title	text
visibility	text
current_revision_id	text
created_at	timestamp with time zone
updated_at	timestamp with time zone

Story revisions

Column	Type
id	text
story_id	text
parent_revision_id	text
persona	text
meta	jsonb
content	jsonb
model	text
params	jsonb
change_summary	text
created_at	timestamp with time zone

Paragraph variant batches

Column	Type
id	text
story_id	text
base_revision_id	text
section_id	text
section_index	integer
paragraph_index	integer
ops_json	jsonb
created_at	timestamp with time zone

Paragraph variants

Column	Type
id	text
batch_id	text
text	text
rank	integer
applied_revision_id	text
created_at	timestamp with time zone

Story shares

Column	Type
id	text
story_id	text
share_token	uuid
can_edit	boolean
expires_at	timestamp with time zone
created_by_user_id	text
created_at	timestamp with time zone

8.6 GPU Inference Service (VM)

The components of the platform that require the most computing resources are distributed across a dedicated virtual machine with GPUs, isolated from the main web infrastructure.

This service is responsible for running large language models used in the two-stage generation pipeline and handling all long-running inference workloads. By offloading these tasks to a separate virtual machine, the system avoids resource conflicts with the frontend and API layers and enables scalable, GPU-accelerated execution.

It runs on a virtual machine equipped with an NVIDIA L40S GPU with 46 GB of VRAM. The inference container is explicitly bound to a single device at startup using Docker GPU constraints.

The GPU inference service implements the platform’s core generative capabilities: structural decomposition of scientific articles (Splitter) and personas-adapted narrative generation (Storyteller). It exposes a FastAPI-based HTTP interface that is called by the backend layer whenever generation or regeneration is requested. This design allows the main application to remain responsive while heavy inference is performed asynchronously on specialized hardware.

The main inference service executes a two-stage pipeline. In the first stage, the Splitter model processes the cleaned text of a scientific paper and produces a structured outline composed of semantically coherent sections, explicitly tailored to the target reader persona. In the second stage, the Storyteller model generates natural language content for each section, grounding the output in retrieved passages from the original paper and adapting tone, length, and complexity to the selected persona. Everything that has already been described in the chapters on part 1.

Both Splitter and Storyteller support optional LoRA adapters, allowing the system to switch from basic models to optimized variants without changing the surrounding infrastructure (currently both set to the fine-tuned version, exactly as indicated by the ablation study conducted). The configuration is controlled via environment variables passed at container startup, allowing for rapid experimentation with different model checkpoints, adapter paths, and generation parameters.

Regeneration In addition to complete story generation, the GPU service also supports different types of regeneration. For the regeneration of the entire story, the same two-stage structure used for the initial generation is used, with any new parameters set by the user in the frontend, passed to the backend, and finally reported here in VM. As for the regeneration of sections and paragraphs, these obviously skip the Splitter steps and only use the Storyteller model with specific prompts for these tasks. Everything is orchestrated by *two_stage_app*, which coordinates requests from the backend, returns responses, and launches the Splitter and Storyteller inference codes.

Data handling and persistence. The GPU container does not incorporate persistent application data. Instead, uploaded PDF files are mounted into the container via host volumes, allowing the backend to access the source documents (required for double side-view display) without duplicating storage space within the image. Model caches and Hugging Face artifacts are also mounted into external directories, reducing container size and startup time.

Isolation and scalability. By performing inference on a separate VM, the platform achieves strong isolation between user-facing services and model execution. Failures or slowdowns in GPU workloads do not directly affect authentication, database access, or front-end rendering. In addition, this architecture allows you to scale the inference service independently, either by deploying more GPU containers or by migrating to larger GPU instances as workload demands increase.

8.7 Dockerization and Deployment

The entire platform is implemented using Docker-based containerization, which ensures reproducibility, dependency isolation, and consistent execution across development and production environments. Each major subsystem is packaged as an independent container, reflecting the logical separation of responsibilities in the overall architecture.

The frontend, Next.js API layer, FastAPI orchestration backend, and GPU inference service are all deployed as separate Docker containers. This separation allows each component to use its own technology stack and runtime environment, such as Node.js for web services and Python with CUDA support for model inference, without interference or dependency conflicts.

For the sciteller-api container (the layer on the VM, the inference service), it is based on an NVIDIA CUDA runtime image and runs with explicit GPU binding. This configuration enables hardware-accelerated execution of large language models while keeping GPU drivers and CUDA libraries isolated from the rest of the system. Environment variables are used to configure model paths, adapter usage, and cache directories, enabling flexible switching between base models and optimized models.

Persistent data, such as uploaded PDF files and model caches, are not integrated into container images. Instead, they are mounted at runtime using Docker volumes. Specifically, the directory containing user-uploaded documents is shared with the GPU container via a bind mount, allowing the inference code to directly access the documents while keeping the container stateless. Database persistence is handled externally by PostgreSQL, ensuring that the application state survives container restarts.

For web-facing services, multi-stage Docker builds are used to separate dependency installation, application compilation, and runtime execution. This approach minimizes production image size and reduces the attack surface while preserving rapid iteration during development.

Containerization enables independent scaling and deployment of individual services. Web-facing components can be replicated to handle increased user traffic, while the GPU inference service can be scaled vertically or horizontally depending on available hardware. This deployment strategy also facilitates migration to managed container orchestration platforms without requiring changes to application logic.

8.8 Scalability and Performance Considerations

The platform was designed with scalability and performance as primary architectural goals, reflecting the computational asymmetry between user interactions and large-scale language model inference. The system explicitly separates latency-sensitive components from computationally intensive workloads, allowing each layer to scale according to its own constraints.

Frontend and API scalability The React frontend and Next.js API layer are stateless and lightweight, making them particularly well-suited for horizontal scaling. You can deploy multiple instances behind a load balancer to handle concurrent user sessions, authentication requests, and metadata operations without shared in-memory state. Since these components do not perform complex computations, their performance is primarily limited by network latency and database throughput rather than CPU or memory constraints.

Asynchronous orchestration and request decoupling Long-running AI tasks are handled by a dedicated FastAPI orchestration backend, which separates user requests from model execution. Requests that trigger story generation are forwarded to the GPU service and monitored asynchronously via progress identifiers and polling or callback mechanisms. This design prevents web services from blocking and allows the system to remain responsive even under high inference load.

GPU-bound inference bottlenecks The main bottleneck in platform performance lies in the inference of large language models, which requires both a dedicated GPU and a significant amount of memory. To overcome this problem, as mentioned above, inference is isolated on a dedicated virtual machine equipped with a GPU. A lightweight GPU selection and locking mechanism ensures that simultaneous requests do not overload the same device, preserving stability and predictable execution times.

Instead of trying to maximize raw productivity, the system prioritizes controlled competition and gradual degradation. GPU workloads are executed in series or with limited parallelism to avoid insufficient memory errors and uncontrolled latency spikes.

Performance is further improved through selective caching of intermediate artifacts, such as clean paper text, contours, and retrieved context segments. Model caches and tokenizer files are stored using shared volumes, reducing startup overhead and repeated downloads. These optimizations significantly reduce the amortized cost of repeated requests on similar inputs.

The architecture favors modular scalability over monolithic optimization. Web services can be scaled horizontally at minimal cost, while inference scalability depends on the availability of GPU resources. This asymmetry reflects a deliberate design choice: the system is optimized for correctness, transparency, and controllability of AI generation rather than maximum throughput at the expense of reliability.

The current design can be extended to support multi-GPU scheduling, distributed inference, or batch generation strategies. Because orchestration and inference are cleanly separated, such enhancements can be introduced without changing frontend layers or APIs, preserving architectural stability as the system evolves.

Chapter 9

User Study

The objective of this user study is to evaluate the SciTeller platform in terms of both usability and perceived quality, focusing on the effectiveness with which the system supports the generation of personalized science stories. In particular, the study aims to evaluate:

- the ease of use of the web interface and its main interaction workflows,
- the clarity, consistency, and perceived usefulness of the generated narratives,
- the extent to which the system is considered reliable in accurately representing the original scientific content,
- the effectiveness of character adaptation in terms of style, tone, and level of technicality.

In addition, the study seeks to identify usability issues, interaction bottlenecks, and limitations of the current design, as well as to gather qualitative feedback to guide future improvements to both the user interface and the underlying narrative models.

9.1 Participants

The user study involved participants selected to represent the main target audience of the SciTeller platform. In order to ensure a balanced assessment between different levels of expertise and different communication needs, participants were evenly distributed across three predefined profiles.

Specifically, the study included Bachelor/Master/PhD students, Researchers & Engineers and Professors, with an equal number of participants assigned to each group. This controlled sampling strategy was adopted to reduce bias towards a single user profile and allow meaningful comparisons between different profiles. Table 9.1 summarizes the distribution of participants by persona.

Table 9.1: Participant distribution by persona

Persona	Number of Participants
Students (Bachelor/Master/PhD)	2
Researchers & Engineers	2
Professors	2
Total	6

In addition to profile classification, participants reported their level of familiarity with artificial intelligence and language models, how often they interact with scientific literature, and their English language proficiency. These baseline characteristics were collected to contextualize user feedback and support future analyses correlating experience level with system perception.

9.2 Tasks

The user study was structured around a series of predefined tasks designed to reflect realistic interaction scenarios with the SciTeller platform. Each task targeted a specific core functionality of the system and was intended to evaluate both the usability and perceived quality of the generated science stories. The tasks were completed sequentially, and participants were asked to interact freely with the system following the instructions provided.

Table 9.2 provides an overview of the three tasks performed during the study, along with their objectives and the main evaluation dimensions associated with each task.

Table 9.2: Overview of user study tasks

Task	Description	Evaluation Focus
Task 1	Generate two complete stories from a scientific paper for two different personas	Story clarity, coherence, persona adaptation, ease of use
Task 2	Modify and regenerate selected sections by adjusting creativity and length parameters	Parameter control, consistency, partial regeneration usability, ease of use
Task 3	Retrieve a previous story version and regenerate alternative versions of a paragraph	Versioning usability, transparency, parameter adherence, ease of use

Task 1: Story Generation This task represents the main point of access to the system and simulates the scenario of a novice user. Participants were asked to generate two complete stories from the same scientific article, each aimed at a different persona. In addition to evaluating narrative quality and character adaptation, this task also served to assess the ease of learning the interface and the clarity of the initial setup process.

Task 2: Section Modification and Regeneration The second task focused on iterative refinement, one of the platform’s main use cases. Participants were asked to modify one or more sections of a previously generated story by adjusting parameters such as creativity level and length, and then regenerate the selected sections. This task was designed to evaluate how effectively users were able to exercise detailed control over the generated content, as well as the system’s ability to preserve overall consistency during partial updates.

Task 3: Version Retrieval and Paragraph Regeneration The third task concerned advanced interaction features related to version control and localized regeneration. Participants were asked to retrieve a previous version of a story and generate alternative versions of a selected paragraph using updated parameters. This task evaluated the transparency of the version control mechanism, the usability of navigation between previous revisions, and the reliability of content regeneration within an evolving narrative context.

The three tasks were intentionally ordered to increase the complexity of interaction, moving from complete story generation (simplest task) to localized regeneration and explicit version management (most complex task). This progression allowed the study to capture both early user impressions and more expert-oriented interaction patterns, providing a comprehensive evaluation of the system across different stages of user engagement.

9.3 Questionnaire Design

The evaluation questionnaire was designed to collect both quantitative and qualitative feedback on usability, perceived quality of the generated stories, interface design, and overall user experience. Participants were asked to fill it out progressively as they completed the tasks, and it was structured in several sections, each targeting a specific dimension of evaluation.

The task-related questions were formulated using Likert scale items to assess difficulty, clarity, engagement, consistency, and adherence to user-defined parameters. Depending on the nature of the interaction being evaluated, a 5-point or 7-point Likert scale was used. The more detailed 7-point scale was used for more technical or subjective aspects, such as narrative consistency, character appropriateness, and adherence to parameters, where greater granularity of responses was considered beneficial.

The usability of the system was evaluated using statements from the System Usability Scale (SUS) [13]. These standard questions assessed perceived ease of use, complexity, learnability, confidence, and overall integration of the system through a specific calculation that resulted in a ranking on the SUS scale.

Other sections focused on the interface and visual design, evaluating layout clarity, readability, visual hierarchy, and feedback mechanisms. Finally, open-ended questions were included to collect qualitative feedback on positive aspects, critical points, and suggestions for improvement. This combination of structured and open-ended feedback was designed to support scalable analysis, allowing for seamless integration of future responses as the dataset grows.

9.4 Results

Given the limited number of participants, the reported results should be interpreted as preliminary and exploratory. Quantitative scores are used to highlight trends rather than to support statistically significant claims.

9.4.1 Quantitative Results

Quantitative results are reported using descriptive statistics, mainly average scores calculated among participants for each item in the questionnaire. Given the use of 5- and 7-point Likert scales, the results are always presented together with the corresponding scale to ensure their interpretability.

Task-Based Evaluation

The first set of metrics evaluates user performance and perception in the three main tasks of the study: story generation, section regeneration, and version retrieval with paragraph regeneration. Tables 9.3, 9.4 and 9.5 summarizes the average scores obtained for the main questions related to the tasks.

Table 9.3: Task 1 — Story generation and content quality results

Metric	Mean Score	Scale
Did the generated stories help you understand the paper?	3.67	1–5
How easy or difficult was this task?	4.17	1–5
The stories were clear and engaging	4.67	1–7
The story was coherent across sections	5.33	1–7
The story style and tone fit the selected persona	5.83	1–7

Table 9.4: Task 2 — Section modification and regeneration results

Metric	Mean Score	Scale
It was easy to modify and regenerate the section(s)	4.33	1–5
The regenerated section(s) was (were) consistent with the rest of the story	5.17	1–7
The regeneration respects the parameters you chose (Modification)	6.33	1–7

Table 9.5: Task 3 — Version retrieval and paragraph regeneration results

Metric	Mean Score	Scale
It was clear how to retrieve an old version	4.33	1–5
The regeneration worked as I expected (Versioning)	4.83	1–5
The regeneration respects the parameters I chose	6.83	1–7

Figure 9.1 provides a visual overview of the average scores obtained for the three tasks. To aggregate items measured on different Likert scales, scores on a 1–5 scale were linearly rescaled to a 1–7 range before computing task-level averages.

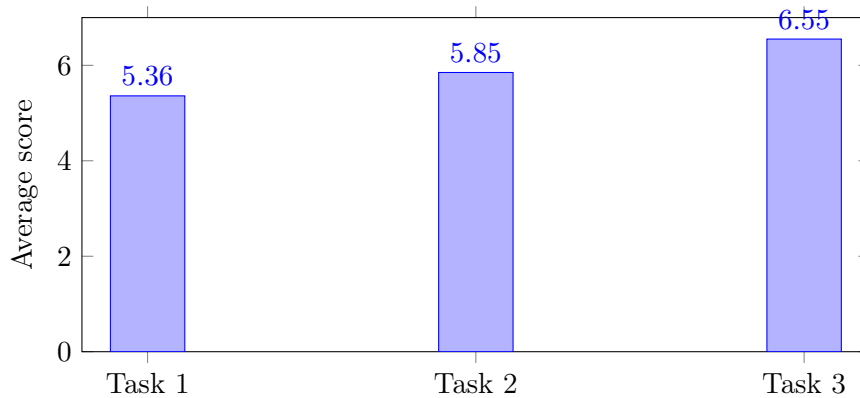


Figure 9.1: Average evaluation scores across the three main tasks

System Usability and Interface Evaluation

The usability of the system was assessed using a series of statements derived from the System Usability Scale (SUS), while aspects related to interface layout and visual design were evaluated through dedicated questions using a 5-point Likert scale. Table 9.6 reports the average scores for the SUS-related items, whereas Table 9.7 summarizes the results concerning interface and visual design.

Table 9.6: System usability (SUS-related) evaluation results

Metric	Mean Score	Scale
I think I would like to use this system frequently	4.17	1–5
I found the system unnecessarily complex	1.17	1–5
I thought the system was easy to use	4.33	1–5
I think I would need the support of a technical person to use this system	1.17	1–5
I found the various functions in this system were well integrated	4.33	1–5
I thought there was too much inconsistency in this system	1.50	1–5
I would imagine that most people would learn to use this system very quickly	4.67	1–5
I found the system very cumbersome to use	2.00	1–5
I felt very confident using the system	4.50	1–5
I needed to learn a lot of things before I could get going with this system	1.33	1–5

Table 9.7: Interface and visual design evaluation results

Metric	Mean Score	Scale
The interface’s layout and structure were clear and easy to understand	4.17	1–5
Texts and icons were easy to read and recognize	4.67	1–5
Colors and contrasts helped me focus on the important areas	4.33	1–5
Visual hierarchy made it easy to understand what to do next	4.17	1–5
The interface looked aesthetically pleasing	4.17	1–5
Feedback animations and messages were helpful	4.50	1–5
I could easily find my way through the menus or controls	4.50	1–5

To obtain an overall measure of perceived system usability, the SUS-related responses reported in Table 9.6 were aggregated following the standard SUS scoring procedure. The resulting SUS score for SciTeller is **86.25**, which corresponds to a grade **A** and is classified as *Excellent* according to established SUS interpretation guidelines. This result indicates a very high level of perceived usability and suggests that users are able to interact with the system efficiently with minimal learning effort.

Persona-Based Analysis

Given the limited number of participants, a quantitative comparison across personas would not be statistically meaningful. For this reason, persona-based differences were analyzed qualitatively based on open-ended responses and task-related comments.

Feedback indicates heterogeneous perceptions of persona adaptation. Some participants reported that stories generated for expert audiences (Researchers and Engineers) were precise and closely aligned with the original paper, while others noted issues such as hallucinations, premature conclusions, or mismatches in tone when targeting academic personas.

Conversely, several users found stories generated for less technical audiences more engaging and accessible, although occasionally overly generic or stylistically repetitive. These contrasting observations suggest that persona adaptation is perceived differently depending on user expectations and expertise, and that the current results are insufficient to draw definitive conclusions.

A larger-scale user study is required to quantitatively assess persona-specific performance and validate these preliminary trends.

9.4.2 Qualitative Insights

Qualitative feedback was analyzed through explanatory comments written by users in the questionnaire. This analysis complements the quantitative results by providing explanations for the trends observed and highlighting aspects not fully captured by numerical scores.

Participants often highlighted the system’s ease of use and the usefulness of character-based story generation as key strengths. The ability to selectively regenerate individual sections or paragraphs and navigate between multiple versions of the story was consistently described as a valuable feature that supports iterative refinement.

Several users reported issues related to system stability and generation time, particularly during PDF processing and regeneration tasks. Furthermore, feedback revealed mixed perceptions regarding the creativity parameter: low values sometimes produced nearly identical results, while higher values could lead to content that deviated from the original article. These observations suggest a need for clearer explanations of parameters and more accurate calibration.

Finally, qualitative comments revealed differing expectations among various profiles. Expert users tended to value accuracy and fidelity to the original article, while feedback indicates that further improvements are needed to better adapt narrative depth and tone to a non-expert audience. These insights provide concrete directions for future improvements and will be revisited as the number of participants increases.

9.5 Implications for the System

The results of the user study provide several actionable insights for the future evolution of the SciTeller platform, both in terms of system design and underlying storytelling models. While the overall feedback indicates a strong level of usability and user satisfaction, the findings also highlight specific areas where targeted improvements can enhance robustness, adaptability, and user control.

User Interface and Interaction Design: The consistently high usability and interface scores suggest that the current interaction paradigm is effective and should be preserved. However, qualitative feedback points to opportunities for refinement, particularly in improving navigation across story versions and in making certain interactions more explicit, such as the selection and comparison of alternative paragraph regenerations. Enhancing visual cues and affordances in these areas could further reduce cognitive load and support more efficient iterative workflows.

Control and Transparency of Generation Parameters: User feedback regarding the creativity parameter indicates a need for clearer communication of its effects on the generated content. Low creativity values were often perceived as overly deterministic, while higher values sometimes resulted in deviations from the source article. This suggests that future versions of the system should either normalize the effective range of the creativity parameter or provide real-time guidance and warnings to users about the expected trade-offs between faithfulness and expressiveness.

Persona Adaptation and Narrative Fidelity: Qualitative feedback suggests a tendency for the system to generate narratives with a tone closer to the original paper, which was often appreciated for expert-oriented personas (e.g., Researchers & Engineers). At the

same time, comments revealed heterogeneous expectations and occasional mismatches in tone or factual fidelity across personas, indicating that adaptation for low-expertise audiences (e.g., general public) still requires refinement. These observations are preliminary and should be validated on a larger participant set.

Model Robustness and Stability: Several participants reported issues related to generation time and occasional errors during generation activities (issues resolved by logging into the webapp). These observations highlight the importance of improving the robustness of the system, including better error handling and a reduction in generation time (currently around 6 minutes). From a modeling perspective, this also suggests the need for more stable decoding strategies and stricter grounding mechanisms to reduce hallucinations and ensure consistency across regenerations.

Chapter 10

Conclusions

This thesis presented *SciTeller*, a complete AI-based system for transforming scientific articles into structured, coherent, and persona-adapted narrative stories. The work addresses a growing need in scientific communication: enabling diverse audiences to access and understand complex research outputs without compromising factual accuracy.

A primary contribution of this work is the design and implementation of a two-stage storytelling pipeline that explicitly separates *content planning* from *narrative realization*. The first stage, referred to as the *Splitter*, analyzes the structure of a scientific paper and produces a high-level outline grounded in the original sections and content. The second stage, the *Storyteller*, transforms this outline into a fluent narrative adapted to a target persona. This architectural separation proved fundamental in improving coherence, interpretability, and controllability of the generation process.

From a system perspective, the thesis contributes a modular, scalable web-based architecture that integrates frontend interaction, authentication and API management, orchestration logic, GPU-based inference services, and persistent storage. The proposed design demonstrates how large language models can be embedded into real-world applications while preserving reproducibility, traceability, and extensibility.

Another key contribution lies in the explicit modeling of audience personas. Instead of treating scientific storytelling as a one-size-fits-all task, the system adapts language complexity, narrative focus, and explanatory depth to different user profiles. This persona-aware approach moves beyond generic summarization and positions storytelling as a form of adaptive scientific mediation.

On the evaluation side, this thesis introduced the *StoryScore*, a composite automatic metric that jointly captures semantic faithfulness, structural alignment, fluency, redundancy control, and hallucination avoidance. By combining multiple complementary dimensions into a single score, StoryScore enables systematic comparison across model configurations while remaining interpretable.

Finally, the thesis contributes an empirical evaluation comprising ablation studies, fine-tuning comparisons, and a user-centered evaluation conducted through an online questionnaire. Together, these experiments provide both quantitative and qualitative evidence supporting the effectiveness of the proposed approach.

10.1 Answers to Research Questions

This thesis was structured around three research questions, divided into two complementary parts. The first two research questions focused on the development and evaluation of

the AI system for personalized scientific storytelling, while the third addressed the design of an interactive web platform that would allow user control, transparency, and traceability.

RQ1: How can long scientific papers be transformed into organised textual representations that support coherent storytelling? This research examined how complex and lengthy scientific articles can be transformed into intermediate representations that enable coherent and controllable narration. The results of this thesis demonstrate that effective transformation can be achieved through an explicit content planning phase that precedes the generation of the narration.

The proposed Splitter module analyzes the structure of scientific articles and produces an organized structure that reflects the logical flow and key components of the original article. This intermediate representation acts as a bridge between the rigid structure of scientific writing and the flexible requirements of storytelling. By breaking down long documents into semantically coherent units, the system reduces the cognitive and generative load on the language model during narrative generation.

Experimental results and ablation studies demonstrate that this organized representation plays a central role in ensuring the structural coherence and thematic coverage of the final stories. Compared to direct end-to-end generation, schema-based narration leads to narratives that better preserve the progression of ideas present in the source article. These results confirm that transforming scientific articles into structured textual representations is a necessary and effective step in supporting coherent scientific narration.

RQ2: How effectively can generative models adapt scientific narratives to different audience profiles? The second research question examined the extent to which generative models can adapt scientific narratives to heterogeneous audience profiles. The results obtained in this thesis provide a positive but nuanced answer. Generative models can effectively perform persona adaptation when persona-specific signals are explicitly encoded during training and reinforced through constrained and structured prompting strategies at inference time.

Persona conditioning allows the Storyteller module to modulate language complexity, explanatory depth, and narrative focus based on the target audience. Automatic evaluation metrics indicate that person-aware generation maintains semantic alignment with the source content, while user studies highlight improvements in perceived clarity and relevance.

At the same time, this research highlights a fundamental limitation of the current state of the art: the lack of evaluation metrics capable of jointly capturing factual accuracy and narrative creativity. Existing metrics provide only partial signals and fail to fully represent the communicative effectiveness of personalized scientific storytelling. Within these limitations, the combined use of automatic metrics and human evaluation offers the most reliable evaluation strategy. Overall, the results demonstrate that personalization is feasible and beneficial, but also that its evaluation remains an open research challenge.

RQ3: How can users be given intuitive control over story generation, refinement, and version navigation while maintaining transparency and traceability? The third research question focused on the level of interaction surrounding the AI system, investigating how users can effectively control and explore the generated stories without compromising transparency and traceability. The results of this thesis demonstrate that intuitive user control can be achieved through explicit interaction mechanisms combined with systematic versioning and artifact tracking.

The developed web platform exposes key points of interaction in the storytelling workflow, allowing users to generate stories, request refinements, and navigate between multiple versions of the same narrative. Rather than treating generation as a single opaque operation, the system supports iterative exploration, where each refinement produces a new identifiable version linked to the original article and previous results.

Transparency and traceability are guaranteed by architectural and design choices that preserve the genealogy of generated content. Each version of the story is associated with its inputs, including the source document, intermediate draft, and generation parameters. This design allows users to understand how stories evolve over time and to compare alternative narrative realizations. User study results indicate that such mechanisms improve trust and usability, demonstrating that automated storytelling systems can offer both creative flexibility and responsible interaction.

10.2 Future Work

Although the results presented in this thesis demonstrate the feasibility and effectiveness of personalized scientific storytelling, several important research directions remain open. These directions encompass model design, evaluation methodology, interaction paradigms, and system scalability, reflecting the interdisciplinary nature of the problem.

A first area of future work concerns the development of more expressive and controllable personas representations. In this thesis, personas are modeled as discrete categories which, although effective, fail to capture the full spectrum of users' backgrounds and communication preferences. Future systems could explore continuous personas embeddings or learned representations derived from user behavior and feedback, allowing for more fluid adaptation and granular control of narrative style, level of detail, and explanatory focus.

Another critical aspect concerns the evaluation of scientific storytelling systems. As highlighted throughout this thesis, existing automatic metrics are insufficient to capture both factual accuracy and narrative creativity. Future work could focus on designing evaluation frameworks that explicitly model this trade-off, for example by combining fidelity metrics with measures of engagement, novelty, or explanatory effectiveness. Human-in-the-loop evaluation protocols and task-specific benchmarks would also contribute to a more reliable and interpretable assessment of narrative quality.

Detecting and mitigating hallucinations remains an open challenge for generative models in high-risk domains. Although the proposed system reduces hallucinations through structural grounding and constrained generation, future research could integrate retrieval-augmented generation, conscious decoding of citations, or external verification modules. Such mechanisms would further strengthen the reliability of generated narratives and support more transparent attribution of scientific claims.

From a system and interaction perspective, future work could expand the current web platform toward richer forms of user control and feedback. Interactive refinement mechanisms, such as section-level editing, real-time person-based adjustment, or explicit user feedback loops, could enable the system to dynamically adapt to user preferences. Longitudinal studies could then investigate how such interactions affect trust, usability, and learning outcomes over time.

The scope of scientific storytelling could also be extended beyond text. Multimodal storytelling, which integrates figures, tables, and visual explanations in line with the narrative structure, represents a promising direction for making scientific content more accessible. Automatic generation and referencing of visual elements within the narrative could further improve comprehension, particularly for non-expert audiences.

Finally, future work could address challenges related to efficiency and implementation. Exploring efficient parameter finetuning or lightweight architectures would make personalized scientific storytelling more scalable and sustainable. Such efforts would facilitate the integration of these systems into real-world scientific communication workflows, including educational platforms, digital libraries, and research dissemination tools.

10.3 Final Remarks

This thesis explored personalized scientific storytelling as a means of bridging the gap between complex research findings and human understanding. By framing storytelling as a structured, persona-aware, and interactive task, the work demonstrates how large language models can be transformed from generic text generators into targeted tools for scientific communication.

Beyond the specific system presented, this work highlights the importance of principled design choices when implementing generative models in high-stakes settings. The combination of structured content planning, controlled narrative generation, and explicit user interaction mechanisms proves essential for balancing creativity, factual accuracy, and transparency.

The proposed approach emphasizes that effective scientific communication is not just a modeling problem, but an interdisciplinary challenge involving natural language processing, human-computer interaction, and systems engineering. Addressing this challenge requires not only powerful models, but also a focus on evaluation, usability, and accountability.

Ultimately, this thesis contributes to broadening the perspective on the role of AI in science: not only as a tool for discovery, but also as a mediator of knowledge. By enabling adaptive, traceable, and user-centered storytelling, systems such as SciTeller have the potential to make scientific knowledge more accessible, interpretable, and impactful for a diverse audience.

Bibliography

- [1] A. Afzal, J. Vladika, D. Braun, and F. Matthes, “Challenges in domain-specific abstractive summarization and how to overcome them,” in *Proceedings of the 15th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, INSTICC, SciTePress, 2023, pp. 682–689, ISBN: 978-989-758-623-1. DOI: 10.5220/0011744500003393.
- [2] B. Capili and J. K. Anastasi, “Methods to disseminate nursing research: A brief overview,” *AJN The American Journal of Nursing*, vol. 124, no. 7, pp. 36–39, 2024.
- [3] H. W. Chung et al., “Scaling instruction-finetuned language models,” *J. Mach. Learn. Res.*, vol. 25, no. 1, Jan. 2024, ISSN: 1532-4435.
- [4] A. Cooper, R. Reimann, D. Cronin, and C. Noessel, *About Face: The Essentials of Interaction Design*, 4th. Wiley Publishing, 2014, ISBN: 1118766571.
- [5] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient fine-tuning of quantized llms,” in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS ’23, New Orleans, LA, USA: Curran Associates Inc., 2023.
- [6] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, *Jukebox: A generative model for music*, 2020. arXiv: 2005.00341 [eess.AS]. [Online]. Available: <https://arxiv.org/abs/2005.00341>.
- [7] M. Guo et al., “LongT5: Efficient text-to-text transformer for long sequences,” in *Findings of the Association for Computational Linguistics: NAACL 2022*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds., Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 724–736. DOI: 10.18653/v1/2022.findings-naacl.55. [Online]. Available: <https://aclanthology.org/2022.findings-naacl.55/>.
- [8] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, “Spacy: Industrial-strength natural language processing in python,” 2020. DOI: 10.5281/zenodo.1212303.
- [9] J. Jansen, J. Salminen, S.-G. Jung, and K. Guan, “Data-driven personas,” *Synthesis Lectures on Human-Centered Informatics*, vol. 14, pp. i–317, Feb. 2021. DOI: 10.2200/S01072ED1V01Y202101HCI048.
- [10] Z. Ji et al., “Survey of hallucination in natural language generation,” *ACM Comput. Surv.*, vol. 55, no. 12, Mar. 2023, ISSN: 0360-0300. DOI: 10.1145/3571730. [Online]. Available: <https://doi.org/10.1145/3571730>.
- [11] A. Q. Jiang et al., *Mistral 7b*, 2023. arXiv: 2310.06825 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2310.06825>.

- [12] F. (Jiang and K. Hyland, “Rhetorical distinctions: Comparing metadiscourse in essays by chatgpt and students,” *English for Specific Purposes*, vol. 79, pp. 17–29, 2025, ISSN: 0889-4906. DOI: <https://doi.org/10.1016/j.esp.2025.03.001>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0889490625000134>.
- [13] J. R. Lewis, “The system usability scale: Past, present, and future,” *International Journal of Human–Computer Interaction*, vol. 34, no. 7, pp. 577–590, 2018. DOI: 10.1080/10447318.2018.1455307. eprint: <https://doi.org/10.1080/10447318.2018.1455307>. [Online]. Available: <https://doi.org/10.1080/10447318.2018.1455307>.
- [14] M. Lewis et al., “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. [Online]. Available: <https://aclanthology.org/2020.acl-main.703/>.
- [15] D. Li et al., “From generation to judgment: Opportunities and challenges of LLM-as-a-judge,” in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, Eds., Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 2757–2791, ISBN: 979-8-89176-332-6. DOI: 10.18653/v1/2025.emnlp-main.138. [Online]. Available: <https://aclanthology.org/2025.emnlp-main.138/>.
- [16] E. National Academies of Sciences and Medicine, *Communicating Science Effectively: A Research Agenda*. Washington, DC: The National Academies Press, 2017, ISBN: 978-0-309-45102-4. DOI: 10.17226/23674. [Online]. Available: <https://nap.nationalacademies.org/catalog/23674/communicating-science-effectively-a-research-agenda>.
- [17] OpenAI, *Chatgpt 5.1*, <https://chat.openai.com>, Large Language Model, 2025.
- [18] Qwen et al., *Qwen2.5 technical report*, 2025. arXiv: 2412.15115 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2412.15115>.
- [19] C. Raffel et al., “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: <http://jmlr.org/papers/v21/20-074.html>.
- [20] D. Rooein, A. C. Curry, and D. Hovy, *Know your audience: Do llms adapt to different age and education levels?* 2023. arXiv: 2312.02065 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2312.02065>.
- [21] Y. Sun, P. J. Wang, J. J. Y. Chung, M. Roemmele, T. Kim, and M. Kreminski, *Drama llama: An llm-powered storylets framework for authorable responsiveness in interactive narrative*, 2025. arXiv: 2501.09099 [cs.HC]. [Online]. Available: <https://arxiv.org/abs/2501.09099>.
- [22] H. Touvron et al., *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2302.13971>.
- [23] A. Vaswani et al., “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010, ISBN: 9781510860964.

- [24] B. Vendeville, L. Ermakova, P. De Loor, and J. Kamps, “Mirage: A metrics library for rating hallucinations in generated text,” in *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, ser. CIKM '25, Seoul, Republic of Korea: Association for Computing Machinery, 2025, pp. 6539–6543, ISBN: 9798400720406. DOI: 10.1145/3746252.3761644. [Online]. Available: <https://doi.org/10.1145/3746252.3761644>.
- [25] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, *Bertscore: Evaluating text generation with bert*, 2020. arXiv: 1904.09675 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/1904.09675>.

Ringraziamenti

A mia **mamma** e mio **papà**, il grazie più grande e più profondo. Per esserci sempre stati, anche quando non sapevo nemmeno io di averne bisogno. Per il sostegno costante, per la pazienza infinita, per i sacrifici silenziosi che non ho mai dovuto chiedere e che non sono mai stati fatti pesare. Per aver creduto in me anche nei momenti in cui io stesso facevo fatica a farlo. La vostra presenza è stata una base solida, una sicurezza, un punto fermo lungo tutto questo percorso. Grazie per l'esempio e per le parole che, nel tempo, sono diventate un metodo. "Meglio farlo bene subito così lo fai una volta sola, piuttosto che farlo male e perderci più tempo". Questo modo di vedere le cose mi ha accompagnato più di quanto forse immaginate, nello studio come nella vita. Grazie per avermi insegnato il valore dell'impegno, della cura, della responsabilità e per avermi trasmesso una determinazione concreta, mai urlata ma sempre presente.

A **Daiana**, mia sorella e la mia persona, per esserci sempre stata, in modo naturale e silenzioso, senza mai dover chiedere. Per il legame che ci unisce e che va oltre le parole, oltre le spiegazioni, oltre il tempo e la distanza. Per la complicità, il sostegno sincero e la presenza costante, soprattutto nei momenti in cui non servivano consigli, ma solo sapere di non essere solo e dover affrontare quelle che per me erano (e sono) montagne invalicabili. Un punto fermo, indelebile, che non potrò mai rimuovere.

Ad **Alice**, collega, amica, famiglia scelta. Abbiamo condiviso molto più di un percorso universitario: abbiamo condiviso giornate infinite, ansie pre-esame, risate, discussioni nate anche da una banale incomprensione, cene improvvisate e non, momenti di stanchezza e momenti di pura felicità. Abbiamo lavorato insieme, riso insieme, ma soprattutto siamo cresciuti insieme. Questo percorso senza di te non sarebbe stato lo stesso, e tante cose non sarebbero state altrettanto memorabili. Questo ringraziamento va oltre questo traguardo e oltre queste pagine, è un grazie che non riuscirò mai del tutto a comunicarti, per avermi letteralmente salvato da uno dei periodi più bui della mia vita. Un grazie anche a tutta la **famiglia Cavasino** per avermi accolto nella loro famiglia con un affetto sincero e spontaneo, facendomi sentire a casa anche quando casa era lontana. Per la disponibilità, per le attenzioni, per la naturalezza con cui mi avete fatto spazio. È qualcosa che porterò con me, sempre.

A **Matteo**, per esserci stato sempre, dall'inizio alla fine, in modo costante e autentico. Per il supporto quotidiano, per la pazienza, per la presenza e soprattutto per la capacità di capire anche quando non sapevo spiegare. Per avermi sostenuto nei momenti di dubbio, per avermi dato forza quando ne avevo poca, e per avermi aiutato a ritrovare equilibrio quando tutto sembrava più pesante. Per aver reso questo percorso più leggero semplicemente restando. Il tuo modo di esserci ha fatto davvero la differenza, e una parte importante di questo traguardo porta inevitabilmente anche il tuo segno.

Ai **nonni** e alle **nonne**, per l'amore incondizionato, per la presenza, per le chiamate e per le attenzioni che non sono mai mancate. E grazie, anche, per i taralli, i regalini e tutto quello che puntualmente compariva ogni volta che dovevo ripartire, piccoli gesti che dicevano molto più di mille parole.

Agli **zii**, **zie** e ai **cugini**, perché sapere di poter tornare, di avere una famiglia con cui condividere telefonate, pranzi, cene, feste e momenti insieme non è affatto scontato. È una fortuna grande, ed è bello sapere di avervi tutti.

Agli **zii Gino, Pino e Tino** che mi hanno risposto a qualsiasi ora del giorno e della notte.

A **Sara**, l'amica di una vita, **Nicole** e tutta la **famiglia Daniele**, o meglio il "Gruppo Vacanze" (e non solo). Perché da quando ero piccolo, la vostra amicizia, le vostre parole e anche solo semplici sguardi vostri mi hanno spesso comunicato forza, fiducia e ammirazione. Sensazioni che mi sono servite più di quanto possiate immaginare per andare avanti, soprattutto nei momenti in cui sembrava più complicato.

Ad **Adele**, la mia "migliore amica francese". Abbiamo costruito tantissimo in pochissimo tempo, e non so davvero come spiegare quanto tu sia stata importante per me. Ti bastava quasi nulla per capirmi, anche in silenzio capivi tutto, mi sei stata accanto nei momenti più complessi di questo periodo, mi hai aiutato concretamente negli studi, negli esami, nelle difficoltà di un percorso molto distante da tutto ciò che avevo fatto prima, ma anche e soprattutto nei consigli di vita, nelle chiacchierate fino a tardi e nelle uscite per svagare. Senza di te, questo cammino probabilmente avrebbe potuto interrompersi prima. Invece mi hai dato la forza di andare avanti anche quando vedevo tutto buio. Sei diventata il mio punto di riferimento, il mio mentore, una guida sicura e una fonte costante di ispirazione e ammirazione.

A **Flavia** e **Riccardo**, per i messaggi continui, per le parole sempre presenti e mai leggere. Per esserci stati con costanza, con sincerità e con quell'attenzione che fa la differenza. Anche a distanza, non mi avete mai fatto sentire solo.

A **Tiziano, Silvia, Matteo, Fabio, Daniele, Anastasia, Federica** e **Francesco**, per i progetti condivisi e il lavoro di squadra. Pezzi di strada fatti insieme che hanno contribuito a questo percorso.

A **Nicolò**, per avermi fatto riscoprire la bellezza di condividere uno spazio e una quotidianità, e per aver trasformato un'esperienza che veniva da un passato complicato in qualcosa di sereno e positivo.

A **Marta, Sofia, Stefano, Ilaria, Mick**, a tutti i miei "**amici di Torino**" e a **Torino**, per le cene, le feste, gli Halloween, i compleanni, le studiate al Comala, per le volte in cui abbiamo cucinato insieme fingendoci grandi cuochi (e sì, una volta ne abbiamo anche incendiata una...), le vacanze, i cortometraggi improvvisati fino a tarda notte e per tutti quei momenti che hanno riempito di vita anche i periodi più intensi. Per la leggerezza condivisa senza doverla cercare e per l'essere stati un pezzo di quotidianità. Per aver reso Torino non solo un luogo, ma un insieme di ricordi, di persone e di sensazioni che resteranno legate a una fase importante della mia vita, che ripenso spesso con nostalgia. Abbiamo passato momenti semplici, ma pieni, che mi hanno fatto bene davvero e ricordi che a posteriori porto con me come qualcosa di prezioso, perché so quanto mi abbiano dato

e quanto abbiano contribuito a rendere quel periodo così vivo.

Agli **amici italiani e non, conosciuti in Francia**, con cui ho condiviso questo pezzo di percorso lontano da casa. È stato bello conoscervi e sapere di non essere solo in questa esperienza.

E poi, a tutti coloro che, anche solo con una frase, un messaggio, uno sguardo o un gesto, mi hanno sostenuto e hanno creduto in me. Grazie. Questo traguardo è anche il risultato di tutte queste presenze, visibili e invisibili, che mi hanno accompagnato fino a qui.

In un percorso che molti pensano sia stato liscio, veloce, senza problemi, senza ansie, portato avanti solo grazie all’“intelligenza” e al “tanto sei bravo”, in realtà ci sono stati tanti dubbi, tanti sacrifici e soprattutto c’è stata tanta fatica. Per questo voglio dedicare l’ultimo ringraziamento a **me**. Un ringraziamento che funge più da promemoria per il futuro, per ricordarmi che è stato difficile, che ci sono stati momenti alquanto bui, dubbi profondi, silenzi pesanti. Per tutte le lacrime che non ho raccontato, per le paure che ho tenuto dentro, per la fatica che spesso ho minimizzato. Per aver avuto il coraggio di chiedere aiuto quando serviva e la forza di andare avanti quando sembrava impossibile. Per ricordarmi che la costanza vale più del talento e che la vera forza si costruisce nei giorni in cui nessuno applaude. Per aver affrontato paure che sembravano più grandi di me, per aver accettato di ricominciare quando qualcosa non andava, per aver sopportato il peso delle aspettative senza lasciarmi definire da esse. Non è stato un percorso perfetto, ma è stato mio. E sono orgoglioso di averlo portato fino in fondo. Oggi, posso dire che ce l’ho fatta non perché è stato facile, ma perché non mi sono arreso.