Towards Robust Synthetic Data Generation for Simplification of Text in French

Tsourakis, Nikolaos

How to cite

TSOURAKIS, Nikolaos. Towards Robust Synthetic Data Generation for Simplification of Text in French. In: Machine learning and knowledge extraction, 2025, vol. 7, n° 3, p. 22. doi: 10.3390/make7030068

This publication URL: https://archive-ouverte.unige.ch/unige:186715
Publication DOI: 10.3390/make7030068

machine learning & knowledge extraction

MDPI

Article

Towards Robust Synthetic Data Generation for Simplification of Text in French

Nikos Tsourakis

Department of Translation Technology, TIM/FTI, University of Geneva, Bd du Pont-d'Arve 40, 1205 Genève, Switzerland; nikolaos.tsourakis@unige.ch

Abstract

We present a pipeline for synthetic simplification of text in French that combines large language models with structured semantic guidance. Our approach enhances data generation by integrating contextual knowledge from Wikipedia and Vikidia articles and injecting symbolic control through lightweight knowledge graphs. To construct document-level representations, we implement a progressive summarization process that incrementally builds running summaries and extracts key ideas. Simplifications are generated iteratively and assessed using semantic comparisons between input and output graphs, enabling targeted regeneration when critical information is lost. Our system is implemented using LangChain's orchestration framework, allowing modular and extensible coordination of LLM components. Evaluation shows that context-aware prompting and semantic feedback improve simplification quality across successive iterations.

Keywords: text simplification; synthetic data generation; large language models; LangChain

1. Introduction

Text simplification is a natural language processing (NLP) task that aims to reduce the lexical, syntactic, or structural complexity of text while preserving its original meaning. It plays a crucial role in making content more accessible for diverse audiences, including language learners, individuals with cognitive or reading disabilities, and children [1,2]. Moreover, text simplification has been shown to enhance downstream NLP tasks such as machine translation, information retrieval, and automatic summarization [3,4]. Despite these promising applications, the development of robust simplification systems remains constrained by the limited availability of high-quality parallel corpora (original and simplified sentence pairs), especially for languages other than English.

In the case of French, the scarcity of simplification resources is particularly acute [5]. Existing datasets are either limited in scale or domain-specific, hindering the training and evaluation of end-to-end systems capable of generalizing across text types and linguistic registers. Recent datasets like WiViCo [6] have begun to address this gap by aligning French Wikipedia and Vikidia content at the sentence level, yet they remain insufficient for training large-scale models due to their relatively small size and lack of contextual richness.

Large Language Models (LLMs) and their multilingual variants have demonstrated strong performance in zero-shot and few-shot text-generation tasks, including simplification [7,8]. In particular, their ability to perform controlled transformations with the appropriate prompt context has opened new avenues for synthetic data generation, providing a practical solution to the data bottleneck in low-resource tasks [9,10]. However, this approach raises concerns about factual accuracy, stylistic consistency, and semantic

check for updates

Academic Editor: Ján Paralič

Received: 26 May 2025
Revised: 4 July 2025
Accepted: 12 July 2025
Published: 19 July 2025

Citation: Tsourakis, N. Towards Robust Synthetic Data Generation for Simplification of Text in French. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 68. https://doi.org/10.3390/make7030068

drift, especially when simplification is performed out of context or without grounding in external knowledge [11,12].

To address these limitations, recent research has explored integrating knowledge graphs with LLMs to inject factual and structured context into generation pipelines [13,14]. In the context of simplification, knowledge graphs can help disambiguate concepts, ensure entity consistency, and improve coherence across long or complex input passages. These synergies make it possible to develop systems that are both linguistically and semantically robust.

In this paper, we present a pipeline for generating robust synthetic data for text simplification in French by prompting a powerful multilingual LLM (OpenAI GPT-4o) with structured contextual input. Our hypothesis is that the quality and relevance of the prompt context significantly impact the clarity, fidelity, and grammaticality of the generated simplifications. To examine this question, we constructed a multi-source input structure that provides both semantic grounding and document-level coherence to the LLM during generation. As the foundation of our data pipeline, we used the WiViCo dataset [6], which we extended by scraping the full Wikipedia and Vikidia articles associated with the original sentence pairs. This extension allowed us to reintroduce the missing context around each sentence and extract structured information that can guide the simplification process more effectively.

Our generation pipeline involves multiple auxiliary components, each contributing different types of contextual knowledge. These include the following: (1) a graph-based abstraction of entities and relations from the original sentence, (2) the paragraph or section in which the sentence originally appeared, (3) a knowledge graph extracted from the paragraph, (4) a refined LLM-generated synopsis of the document, (5) the key points extracted from the document for semantic orientation, and (6) feedback for perceived improvement in simplification. These sources are presented as part of the prompt to the LLM, offering it both local (sentence-level) and global (document-level) understanding to support accurate simplification.

Beyond the integration of contextual and conceptual knowledge into the simplification prompt, our pipeline also incorporates a second large language model in an evaluative role. This LLM-as-a-judge mechanism [15] operates post-generation, assessing the quality of the simplified output in terms of missing relationships in the output. If the output fails to meet expected standards, the evaluator triggers a regeneration cycle with additional guidance to the generator.

To implement and orchestrate this pipeline, we employed the LangChain framework (LangChain; LangChain Framework Documentation; https://www.langchain.com; accessed on 3 July 2025), which offers modular abstractions for building LLM applications, including chaining, memory, and structured output parsing. All code and prompt templates are provided in the following repository: https://gitlab.unige.ch/Nikolaos.Tsourakis/synth-data-simp-fr; accessed on 3 July 2025. LangChain allowed us to define reusable components such as prompt templates, output schemas, and runnable graphs that compose the multi-step document-processing and generation tasks. Its integration with tools like OpenAI's GPT-4o and Pydantic-based parsers ensured that our system was both extensible and robust.

In summary, this paper presents a modular, LLM-orchestrated pipeline for simplification of text in French, integrating structured semantic signals and iterative feedback. Beyond generation, our system performs evaluation and correction based on knowledge-graph alignment and simplicity scoring. This enables convergence toward outputs that balance fidelity and simplicity. We offer a reproducible framework for synthetic corpus creation that combines summarization, prompting, and graph-based evaluation.

The paper is organized as follows. The next section presents related work in the field. Section 3 discusses the data-acquisition process, while Sections 4 and 5 focus on two specific techniques used in this work. In Section 6, we present the overall orchestration pipeline; we then present the relevant results and discussion in Section 7. Section 8 concludes the paper.

2. Related Work

Text simplification has been widely studied as a subtask of text generation, with early approaches focusing on rule-based transformations and syntactic parsing [16,17]. These systems often relied on handcrafted rules or linguistic resources such as WordNet (Princeton University; About WordNet; https://wordnet.princeton.edu; accessed on 3 July 2025) to replace complex constructions with simpler equivalents [18]. More recent efforts have shifted toward data-driven methods, particularly those based on statistical and neural machine-translation models, where simplification is framed as a monolingual translation task [19,20].

The advent of sequence-to-sequence architectures such as LSTMs and later Transformer models has significantly improved the quality of automatic simplification [21,22]. Pretrained language models, including BERT and GPT, have also been adapted for the task in both supervised and unsupervised settings [23,24]. However, these models are typically trained on limited corpora such as Newsela [20] or Simple Wikipedia (Simple English Wikipedia; https://simple.wikipedia.org; accessed on 3 July 2025), both of which are primarily English-centric and domain-specific. Recent work by Jamet et al. [25] investigated the use of LLMs for simplification of text in French, focusing both on difficulty estimation and the generation of simplified versions. Their experiments demonstrated that multilingual models can effectively perform simplification even in low-resource settings without requiring extensive language-specific fine-tuning.

Nevertheless, the availability of high-quality resources remains a major bottleneck. Compared to English, significantly fewer resources are available for simplification in other languages. French, in particular, lacks large-scale aligned corpora for sentence-level simplification. Notable efforts include Alector [26], a manually simplified corpus of literary and scientific texts tailored for children and language learners, offering valuable insights into human-generated simplification strategies. While these resources provide valuable starting points, they are constrained by limited sentence coverage, domain specificity, or lack of parallel structure. Recent contributions like WiViCo [6], which aligns French Wikipedia and Vikidia content at the sentence level, represent a step toward broader coverage. However, all these datasets remain insufficient for training large-scale models due to their relatively small size, lack of linguistic diversity, and absence of contextual metadata. These factors are increasingly recognized as essential for robust simplification in real-world applications.

Some parallel efforts in multilingual or cross-lingual simplification have attempted to leverage transfer learning from English datasets, but these approaches are often constrained by differences in syntactic structure, word order, and lexical ambiguity [7]. Other methods, such as MUSS [27], avoid direct transfer by relying on unsupervised paraphrase mining and multilingual pretraining to build simplification models in languages like French. Nevertheless, there remains a significant need for synthetic methods that can generate high-quality simplification data in French from limited annotated resources.

The use of large language models to create synthetic data has become increasingly popular as a strategy for augmenting limited training sets. Prompt-based data-generation approaches have been employed for tasks ranging from self-instruction [28], summarization [29], and NLP queries into SQL statements [30], to data cleaning [31], paraphrasing [32], and structured data generation in domain-specific contexts such as the

medical context [33]. While these methods show promise, they also raise concerns related to hallucinations, repetition, and divergence from ground truth [11,34,35].

For simplification tasks, several works have explored generating sentence pairs using controlled prompts that instruct the model to simplify input text [23]. However, these often lack document-level context, which is critical for preserving semantic consistency and avoiding oversimplification of named entities or coreferential expressions.

2.1. Meaning Preservation vs. Simplicity

A central challenge in automatic text simplification is balancing meaning preservation with linguistic simplification. Systems that emphasize simplification often risk semantic drift (altering or omitting essential information), while those focused on preserving meaning tend to produce output that is only marginally simpler than the original. This trade-off has been a longstanding tension in the field and has shaped the design of various simplification architectures.

Earlier rule-based and syntax-driven systems prioritized fidelity to the original content by applying deterministic transformations [16,17], but they often lacked the flexibility needed for substantial simplification. In contrast, neural sequence-to-sequence models, particularly Transformer-based architectures, have enabled more aggressive simplification strategies but frequently suffer from semantic hallucinations or loss of critical detail [23,36].

Recent efforts have attempted to strike a balance by introducing mechanisms for control of generation, such as specification of target readability levels or lexical constraints [22]. Some models include auxiliary objectives or post-editing steps to reduce semantic loss, while others leverage external resources (e.g., paraphrase databases or alignment scores) to measure fidelity [37].

In this work, we explicitly address this duality by incorporating both structured semantic signals and LLM-based evaluation mechanisms. By extracting knowledge graphs from the original and simplified texts and comparing their relational structures, we ensure that core semantic content is retained. Simultaneously, we introduce iterative refinement driven by feedback on fluency and simplicity, allowing the model to converge toward outputs that are both accessible and faithful. This hybrid approach avoids the extremes of oversimplification and undertransformation, aiming for high-quality simplifications grounded in context.

2.2. LLM-as-a-Judge

Beyond the integration of contextual and conceptual knowledge into the simplification prompt, our pipeline also incorporates the LLM in an evaluative role. This LLM-as-a-judge mechanism operates post-generation, assessing the quality of the simplified output in terms of fluency, semantic fidelity, and simplicity. If the output fails to meet expected standards in any of these dimensions, the evaluator triggers a regeneration cycle with additional guidance to the generator. This architecture reflects a growing trend in NLP toward using LLMs not only as content generators but also as evaluators or critics [38]. This paradigm has shown promise in tasks like summarization [15], error correction [9], and data augmentation [39]. In a related context, it has been used to refine the simplification of spontaneous utterances in spoken dialogue through iterative feedback involving two distinct LLMs, a generator and an external evaluator [40]. In contrast, and for reasons discussed in the following section, we employ a single LLM to handle both tasks.

The use of self-assessment and iterative feedback allows for refinement of generated outputs and offers a cost-effective alternative to expensive human annotations, though it is not without limitations. Studies have highlighted that LLM evaluators may introduce biases, including verbosity preference, positional effects, or overfitting to prior outputs [41].

To mitigate these risks, our system incorporates prompt variation and external context validation, ensuring that the evaluator remains sensitive to structural and semantic integrity rather than to superficial cues. This generator–evaluator loop is particularly well-suited to the task of text simplification, where quality is often multi-dimensional and subjective. As generation is incorporated with in-loop assessment, the ultimate future aim is to construct a higher-quality, more reliable corpus of French simplifications suitable for training downstream Automatic Text Simplification (ATS) models.

2.3. Knowledge Graphs

Knowledge graphs (KGs) encode entities and their relations in a structured form and have been increasingly used to enhance language models by grounding outputs in factual context [13]. In text simplification, KGs help to preserve meaning by disambiguating references, ensuring factual accuracy, and guiding transformations toward more faithful outputs.

A growing body of work, including the roadmap presented by Pan et al. [42], highlights the potential of unifying LLMs and KGs in a mutually reinforcing framework. The paper categorizes integration strategies into three main paradigms: KG-to-LLM, where graph information is injected into the language model via prompts or adapters; LLM-to-KG, where models generate or complete knowledge graphs from raw text; and joint reasoning, where both modalities co-evolve in interactive or iterative loops. Our system aligns most closely with the KG-to-LLM paradigm, as its extracts knowledge graphs from text and uses them to structure the generation process.

We used lightweight KGs derived from named entity recognition and dependency parsing over both the target sentence and its context. These graphs, combined with summaries and key ideas, provide interpretable and efficient semantic input to the pipeline. While they are smaller than full-scale knowledge bases, they effectively capture core relations such as causality, temporality, and attribution. Since these KGs are incorporated into a prompt at evaluation time to guide the simplification process, especially by highlighting relevant text segments, it is important to maintain continuity and shared context by using the same LLM that generated them. For this reason, we employed the same model for both generation and evaluation.

2.4. Frameworks for Modular LLM Pipelines

LangChain and similar LLM-orchestration frameworks have facilitated the design of modular, multi-step generation pipelines. LangChain provides abstractions such as chains, prompts, memory, and document loaders, which make it suitable for building iterative systems like the one presented in this work. Its ability to integrate structured output parsers and external tools (e.g., web scrapers, graph extractors) has made it a key component of our implementation.

LangChain provides a high-level framework for structuring interactions with LLMs. It abstracts language-model operations into composable modules such as chains, agents, and graphs. While LangChain offers fine-grained control over prompt construction and response parsing, it lacks built-in facilities for persistent semantic memory or relational reasoning.

In our system, LangChain handles prompt orchestration and execution, while knowledge graphs extracted via LangChain's LLMGraphTransformer (LangChain; LLMGraphTransformer; https://python.langchain.com/api_reference/experimental/graph_transformers/langchain_experimental.graph_transformers.llm.LLMGraphTransformer.html; accessed on 3 July 2025) provide semantic anchoring. These two components function in synergy: the structured symbolic representation ensures fidelity, and the LLM adds fluency and generalization.

3. Data Acquisition

To construct a reliable and context-rich dataset for our simplification pipeline, we built upon the WiViCo dataset [6], which aligns French Wikipedia (complex) and Vikidia (simplified) articles. However, the original dataset provides only aligned sentence pairs and their respective source URLs, without the full contents of the article. This limitation hinders the ability to provide broader context during simplification and restricts downstream tasks such as knowledge-graph extraction and document-level summarization.

To overcome this barrier, we developed a dedicated scraping pipeline to obtain the full content of both Wikipedia and Vikidia pages referenced in the dataset. We focused exclusively on sentence pairs explicitly labeled as complex-to-simple, filtering out all other mappings. Using the original URLs provided by WiViCo, we issued requests to retrieve each corresponding article and parsed its structure using standard HTML parsing tools, specifically BeautifulSoup (Beautiful Soup; Python Package Index; https://pypi.org/project/beautifulsoup4/; accessed on 3 July 2025). The HTML content was segmented by semantic sections to retain logical grouping and enable progressive summarization later in the pipeline.

Since the original dataset dates from 2023, many of the URLs now point to pages that have undergone revisions or structural changes. These discrepancies can introduce noise or render segment alignment impossible. To address this, we implemented a two-step filtering strategy. First, we manually verified the retrieved documents for relevance, ensuring that the target segment still existed within the body of the article. In cases where slight modifications were present (e.g., punctuation changes or reordering), we applied fuzzy matching techniques to re-identify the corresponding content blocks with sufficient tolerance.

Additionally, to maintain the integrity of segment alignment, we discarded pages that had experienced substantial rewriting or that were no longer accessible. All content was chunked by section to enable modular processing in subsequent steps. This structure facilitates both contextual reasoning and the application of our progressive summarization method, which is described in Section 5.

Following this filtering and reconstruction process, we compiled a corpus of approximately ten thousand complex–simple text pairs, each enriched with their full article context. Part of this corpus served as the foundation for evaluating the modules in our pipeline. In the following two sections, we discuss techniques incorporated in this work.

4. Knowledge Graph Extraction

To enhance the semantic grounding of text simplification, we incorporated lightweight knowledge graphs derived from the input. These graphs are intended to expose key entities and their relations, supporting contextual reasoning and fact-preserving generation. The extraction of KGs is performed prior to simplification, with the resulting graphs passed as structured inputs to the language model alongside summaries and contextual metadata.

Each input is processed through the custom transformer interface of LangChain's LLMGraphTransformer, which relies on large language models to infer semantic triples in the form (subject, relation, object). These triples are stored and reused during simplification, evaluation, and regeneration steps. Their inclusion enables explicit control over semantic coverage, increasing the transparency and correctness of LLM outputs. The transformer accepts raw text as input and returns a lightweight, interpretable graphical representation of the content. Internally, the module uses a prompt-based approach to instruct the language model to identify salient entities and their relationships, including temporal, spatial, causal, and taxonomic links.

Figure 1 visually depicts the knowledge graph generated from a sentence after it had been processed with the transformer: *Tech Innovators hosted Future of AI in San Francisco in 2025, addressing issues related to AI Ethics and its impact on society, discussing key machine learning and human rights*. The resulting semantic triples, shown in the figure, offer a structured view of the event and its participants. Each triple captures a specific aspect of the input sentence; for instance, the event Future of AI is linked to its organizer Tech Innovators via the relation HOSTED and anchored in time and space through HELD_ON and HELD_IN links to 2025 and San Francisco, respectively.

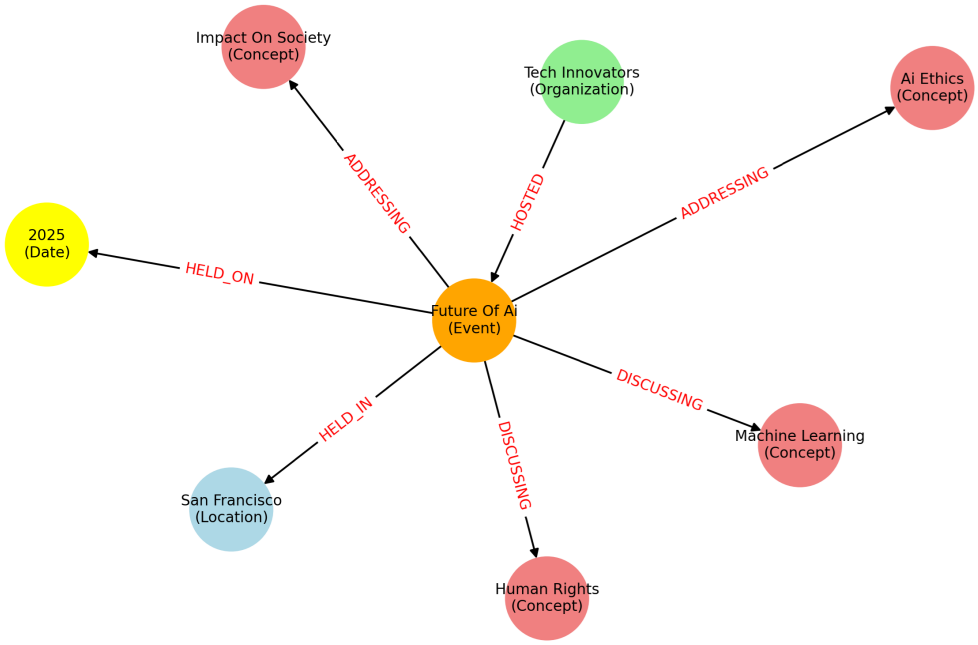

Figure 1. LLM-extracted knowledge graph from a sample sentence.

These extracted graphs are lightweight but semantically rich. They are used as part of the prompting context during simplification to ensure that the model retains key factual content and does not hallucinate critical details. Once extracted, the graphs serve multiple roles: they guide the simplification model's focus, enable coreference resolution across sentences, and provide redundancy checks against hallucinations. Since the graphs in this work are generated using the same version of the LLM employed in downstream simplification, they are stylistically and semantically aligned with the rest of the pipeline, enabling smoother integration. Notice that the knowledge-graph extraction is performed independently on the complex sentence and its surrounding context.

5. Progressive Summarization

The proposed progressive summarization is an iterative document-understanding technique designed to address the challenge of processing long-form content with limited-context models. Rather than attempting to summarize an entire document in a single pass, progressive summarization constructs a running summary that is updated as each new chunk of the document is processed. This approach, inspired by knowledge accumulation and refinement patterns, offers a scalable solution for summarizing documents too large to fit entirely within the context window of a language model.

LangChain provides a concrete implementation of this approach through its Refine-DocumentsChain module [43,44]. In this framework, the summarization process begins with the first document chunk, from which an initial summary is generated. Then, for each subsequent chunk, the model receives both the chunk and the current summary. The

model is prompted to produce an updated version of the summary that incorporates new information while preserving and coherently extending the existing content.

This technique differs from alternative summarization paradigms such as the MapReduceDocumentsChain [45], which generates individual summaries for each chunk and then merges them in a final reduction step. While more parallelizable, the map-reduce method lacks the continuous context propagation that progressive summarization offers.

In our pipeline, we adapted LangChain's refinement model to treat the progressively refined summary as a semantic anchor that evolves throughout the document. We first segmented each Wikipedia article into coherent chunks based on its internal section structure. Each section was encapsulated as an object containing both the text body and relevant metadata (e.g., section title and source URL). The result was a structured list of documents where each entry included its original source and a list of semantically meaningful sections, forming the input for the summarization pipeline. As each chunk of the source document is processed, the language model updates a structured data object containing the following:

- a running summary—a concise narrative of the document so far,
- a list of main ideas—key semantic points.

This cumulative context is later injected into the prompt for simplification, ensuring that the model simplifies sentences with awareness of their broader document-level context. The basic steps of the algorithm (Algorithm 1) are shown below:

Algorithm 1: Progressive summarization steps.

In: Document set D
Language model $\mathcal{M}$
Prompt template $\mathcal{P}$
Output schema $\mathcal{S}$
Out: List of refined summaries $\mathcal{R}$
Initialize empty list $\mathcal{R}$
for each document d in D **do**
  Set initial summary $s \leftarrow \mathcal{S}()$
  for each section c in $d.sections$ **do**
    Build prompt using s and c via $\mathcal{P}$
    Run model: $y \leftarrow \mathcal{M}(\text{prompt})$
    Parse output: $s \leftarrow \mathcal{S}.parse(y)$
  end for
  Append $\{d.source\_url, s\}$ to $\mathcal{R}$
end for
return $\mathcal{R}$

We applied progressive summarization to all Wikipedia articles extracted during the data acquisition phase described in Section 3. Each article was first segmented by section, and the summarization model incrementally processed these chunks to construct a coherent, running summary. This context-rich summary, together with a distilled list of main ideas (5 in total), was stored in a structured file and served as input to the simplification pipeline described in the following section.

6. LLM Orchestration

In this section, we present our approach to orchestrating the LLM, emphasizing the integration of structured semantic information through knowledge graphs. Our method enables the system to reason over contextual cues, identify missing semantics, and iteratively refine its output. This orchestration layer leverages LangChain's graph-based abstractions in conjunction with symbolic representations of meaning extracted via LLMGraphTransformer. To implement a multi-step simplification process, we defined the

custom orchestration graph shown in Figure 2. Each node in the graph corresponds to a transformation or analysis step, with conditional logic determining edge traversal. Not all nodes invoke a chain (some perform deterministic processing or control logic), but those that do are indicated with the notation Table ID → LLM, referencing the corresponding prompt definition included in Appendix A. This dynamic routing allows the system to adaptively correct simplifications based on semantic deltas. In all LLM invocations, we set the temperature to 0.0 and use the default values for top_p and max_tokens, ensuring deterministic outputs.

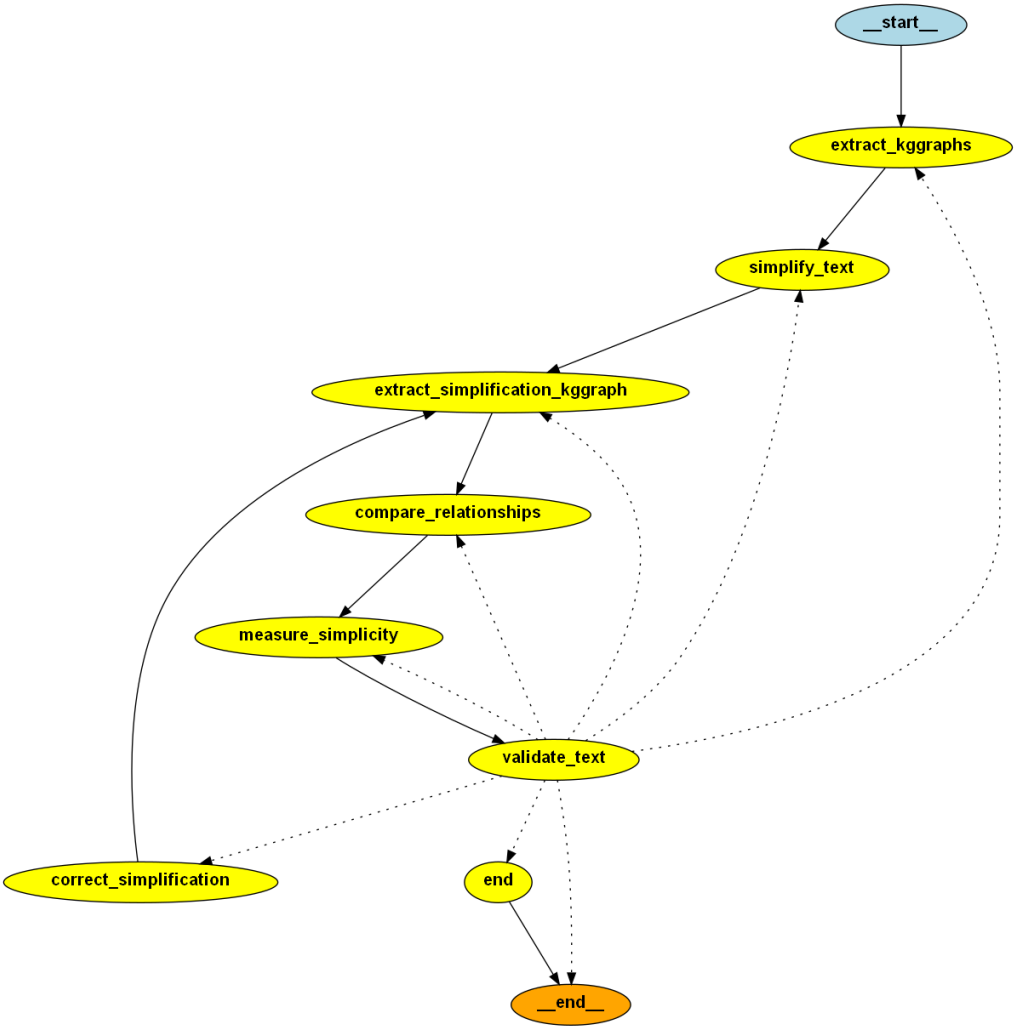

Figure 2. Orchestration graph of the main steps in the simplification pipeline. Solid arrows indicate mandatory execution flow, while dotted arrows represent conditional or optional transitions based on intermediate validation outcomes. Yellow nodes represent individual processing states in the pipeline. The blue _start_ node marks the entry point of the graph, and the orange _end_ node indicates successful termination.

The orchestration graph consists of the following nodes:

- extract_kggraphs: This entry point extracts structured knowledge graphs from both the input sentence and its broader context using the LLMGraphTransformer. It also retrieves the document-level summary and main ideas to provide semantic grounding for later steps.
- simplify_text: Using only the raw input, this node prompts the LLM to generate a baseline simplification. This serves both as an initial candidate and a reference point for iterative evaluation (Table A2 → LLM).

- extract_simplification_kggraph: A second invocation of the graph transformer is used to generate a knowledge graph from the newly simplified sentence. This enables structural comparison with the original input.
- compare_relationships: This component identifies semantically preserved and missing triple relations between the original and simplified graphs (Table A3 → LLM). These results serve as inputs to both the validation and scoring steps.
- measure_simplicity: The shared relationships are linked to their corresponding text spans in both the original and simplified sentences. Each span is scored on a 0–1 simplicity scale (Table A4 → LLM). Moreover, a separate LLM-based ranking compares the new simplification to the original candidate in terms of accessibility and fluency (Table A5 → LLM).
- validate_text: If any original relationship is missing from the simplified version, the result is deemed invalid. Otherwise, it is accepted and the pipeline proceeds to the final node.
- correct_simplification: When validation fails, this node prompts the LLM to revise the latest simplification. The prompt is enriched with missing semantic relations, previous candidates, and LLM-generated simplicity feedback to guide correction (Table A6 → LLM). The process then loops back to extract_simplification_kggraph.
  - In an alternative configuration, the correction prompt omits contextual information and relies solely on the input knowledge graph and simplicity feedback. This variation, analyzed in Section 7, allows us to isolate and assess the impact of both local and global context within the proposed pipeline.
- end: The final node terminates the pipeline either when a valid simplification is achieved or when the maximum number of iterations is reached.

We also applied a technique informally referred to as double-down prompting (Microsoft; Prompt Engineering Techniques; https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering; accessed on 3 July 2025), which involves repeating or emphasizing key instructions multiple times within the same prompt. This approach increases the salience of important constraints, such as style, tone, or formatting, and has been observed to improve the likelihood that LLMs adhere to the desired behavior. For Table A6, the technique was used to reinforce that simplifications must be grounded strictly in the input text and that the model should not introduce unnecessary content derived from the provided contexts.

In the following section, we show the proposed pipeline in action and analyze the effects of two configuration settings.

7. Results and Discussion

To evaluate the effectiveness of our simplification pipeline, we used the first 100 sentences from the dataset introduced in Section 3. Our goal was to assess how different prompting strategies influence the quality of generated simplifications. Specifically, we contrasted two variant conditions for Table A6: one that incorporated both the local and global information associated with the input and one that excluded this contextual information. Specifically, both prompts utilized the same underlying model and simplification logic, differing only in the presence or absence of contextual elements. By comparing these two conditions, we aimed to determine whether including structured contextual knowledge leads to measurably improved simplification quality.

We present results across three simplification iterations. The first iteration can be considered a baseline, as it applied the plain, minimal Table A2 prompt without any knowledge-graph refinement or contextual injection. The second and third iterations

progressively incorporated the refined Table A6 prompt, potentially guided by feedback or additional constraints. This iterative setup allowed us to examine how simplifications evolve over time and whether context applies a stronger influence in the initial or later stages of generation. When context was included in the prompt, the mean end-to-end processing time for the complete generation was 78 s ($\sigma = 33$ s). Figure 3 summarizes the key information for the tested conditions.

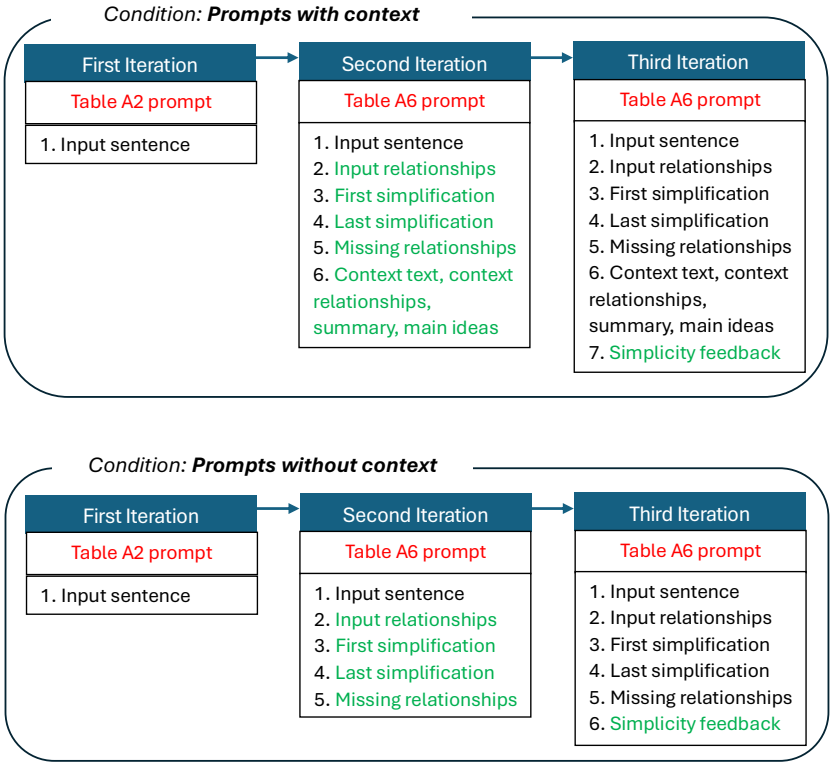

Figure 3. Prompts and inputs used in each of the three iterations and the two conditions. Additions from the previous iteration are shown in green.

Consequently, the analysis in the following sections is organized around two conditions (whether context is used or not) and is tracked across three successive simplification iterations.

7.1. Accepted Simplifications

Based on the plot shown in Figure 4, it is possible to see how the pipeline's validation mechanism behaves across different iterations and prompt conditions. As described in Section 6, the validate_text node checks whether all essential relationships extracted from the original sentence are preserved in the simplified output. If any required relationship is missing, the simplification is marked as invalid. In this respect, the extracted knowledge graphs, capturing both common and missing relationships, serve as a useful proxy for assessing meaning preservation.

In iteration 1, a large majority of simplifications were flagged as invalid: 91 in both the context and no-context conditions. This was expected, as the first iteration applied a basic, unrefined prompt that had not yet benefited from feedback or contextual guidance. However, we observed a notable improvement in subsequent iterations. In iteration 2, the number of valid simplifications increased substantially, reaching 27 (18 + 9) in the context setting and 31 (22 + 9) without context. This trend continued with modest effects in iteration 3, yielding 30 valid simplifications under the context condition and 34 under the no-context condition.

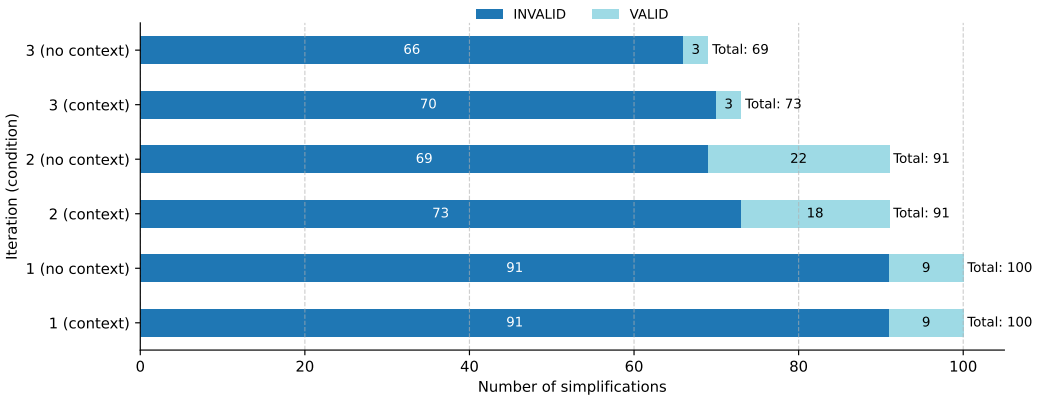

Figure 4. Validation results by iteration and condition.

These results suggest that iterative prompting plays a role in improving semantic fidelity. Through refining of the simplifications over the next steps, the model became more effective at preserving the core relationships extracted from the input. As already depicted in Figure 3, the iterative simplification process reflects a progression of conflicting optimization forces. The first iteration prioritizes surface-level simplification, often at the expense of meaning preservation. The second iteration introduces feedback based on missing semantic relationships, resulting in a strong corrective push toward fidelity and coverage. By the third iteration, the model is prompted to consider both simplicity and completeness simultaneously, balancing two often-competing objectives.

This balancing act may explain the relatively smaller improvement between iterations 2 and 3. While some outputs benefit from the dual emphasis, others may sacrifice previously recovered information in pursuit of greater simplicity. These dynamics suggest that while iterative prompting is beneficial, careful calibration is needed to prevent oscillation or overcorrection in later stages.

7.2. Evaluation of Automatic Metrics

Next, we relied on four automatic metrics commonly used in natural language generation tasks: BLEU, ROUGE-L, SARI, and BERTScore. BLEU [46] measures n-gram overlap between the predicted output and a reference sentence, which reflects surface-level fidelity and fluency. ROUGE-L [47] captures the longest common subsequence between texts and is sensitive to sentence-level structure. While these two metrics reward lexical similarity to the original input, they are not specifically tailored for simplification. To address this, we included SARI [48], a metric designed for the simplification task that measures how well words are added, deleted, or kept relative to both the input and a simpler reference (the Vikidia sentences obtained in Section 3). Unlike BLEU or ROUGE, SARI explicitly evaluates the transformation quality of simplification operations. Lastly, BERTScore [49] was used to assess the semantic similarity between output and reference, leveraging contextual embeddings from a pretrained multilingual BERT model. Table 1 shows the aggregated results of the four metrics.

The scores reveal only marginal differences between the context and no-context conditions, and these differences are unlikely to reflect a practically meaningful distinction in output quality. BLEU scores fell between 0.1039 and 0.1183; these values are low in absolute terms but not unexpected for a simplification task. BLEU was designed for machine translation and tends to penalize outputs that deviate lexically from the source, even if they are well-formed simplifications. ROUGE-L values, ranging from 0.3115 to 0.3288, indicated moderate lexical overlap and structural similarity with the source. SARI scores consistently hovered around 43.0 across conditions and iterations, and this score is

considered strong based on prior simplification benchmarks. This indicates that the model performs simplification operations regardless of prompt context. BERTScore values, which ranged from 0.7807 to 0.7879, were also high, suggesting strong semantic fidelity to the original Wikipedia sentence. Taken together, these results demonstrate that the system achieved stable and strong simplification performance under both prompting strategies. The addition of contextual information did not lead to a significant gain in the metrics, but neither did it impair the output.

Table 1. Automatic evaluation metrics (mean ± std) across iterations and conditions.

Iteration	Condition	BLEU	ROUGE-L	SARI	BERTScore
1	context	0.1183 ± 0.0962	0.3288 ± 0.1051	43.51 ± 6.65	0.7879 ± 0.0396
1	no context	0.1169 ± 0.0954	0.3287 ± 0.1064	43.35 ± 6.81	0.7876 ± 0.0397
2	context	0.1124 ± 0.0900	0.3203 ± 0.0967	42.86 ± 5.94	0.7850 ± 0.0363
2	no context	0.1168 ± 0.0919	0.3213 ± 0.1052	43.14 ± 6.57	0.7865 ± 0.0395
3	context	0.1138 ± 0.0878	0.3159 ± 0.1012	43.21 ± 5.83	0.7835 ± 0.0404
3	no context	0.1039 ± 0.0835	0.3115 ± 0.1005	43.44 ± 6.49	0.7807 ± 0.0385

7.3. Evaluation of Relationships

To better understand how the simplification process preserves or omits semantic content, we analyzed the distribution of extracted relationships across conditions, iterations, and relationship types. The data for this analysis were derived from the prompt of Table A3. For each simplification, we compared the structured-knowledge graph derived from the output with that of the input sentence, identifying relationships that were either preserved (common) or omitted (missing). Figure 5 presents boxplots of the numbers of common and missing relationships across the three simplification iterations and does so separately for the context and no-context conditions.

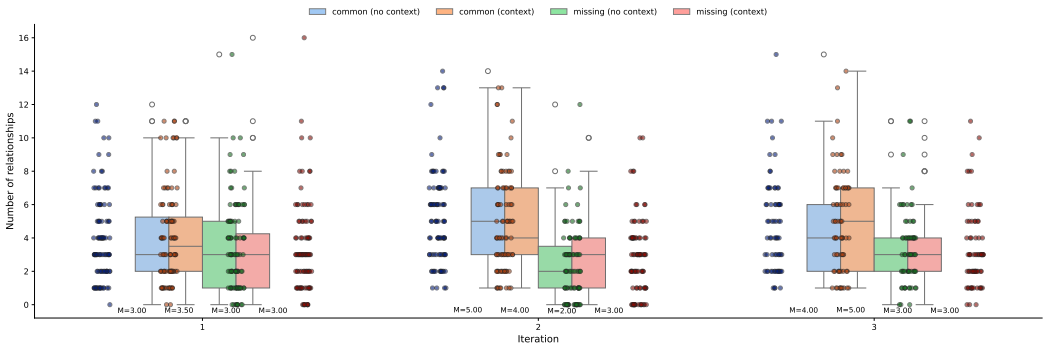

Figure 5. Distribution of common and missing relationships by iteration. Each dot is a data point showing the number of relationships at a specific iteration for a given configuration and condition.

The plots reveal several noteworthy trends. First, the number of common relationships tended to increase from iteration 1 to iteration 2 in both conditions, suggesting that iterative refinement helps in retaining more of the original meaning. This increase is particularly visible in the context condition, where the median number of preserved relationships rose from 3.5 in iteration 1 to 5.0 in iteration 3. In contrast, the number of missing relationships shows only modest changes, with a slight decrease in later iterations indicating improved semantic coverage in the simplifications. Interestingly, the variability in missing relationships is relatively consistent, suggesting that while iteration helps, some inputs remain prone to meaning loss regardless of prompt design.

Overall, the results suggest that including contextual knowledge (via local and global knowledge graphs) can support better preservation of semantic content, especially when

this step is combined with iterative simplification. While improvements are modest, the trend toward higher counts of common relationships under the context condition reinforces the role of structured context in guiding the model to maintain essential relational information.

7.4. Evaluation of Simplicity Scores

To evaluate how effectively the proposed pipeline improves simplification at a fine-grained level, we analyzed the simplicity scores assigned to relationship-specific text segments within each sentence. These scores were produced by the measure_simplicity node using the prompt of Table A4. For each relationship extracted from the input sentence, the prompt identifies the most relevant segment of text and rates its simplicity on a scale from 0 to 1, where 1 denotes maximum simplicity and 0 indicates very high complexity. This process is applied to both the original input and the corresponding simplification, yielding a structured and relationship-aware assessment of linguistic complexity.

Figure 6 shows the distribution of these simplicity scores across iterations and prompting conditions. As expected, simplified texts generally achieved higher scores than their original inputs. For example, in iteration 1, both the input (context and no-context) conditions have a median simplicity score of 0.85. In contrast, the simplifications show higher medians: 0.86 for the context condition and 0.88 for the no-context condition. This indicates a clear simplicity gain even after the first round of simplification.

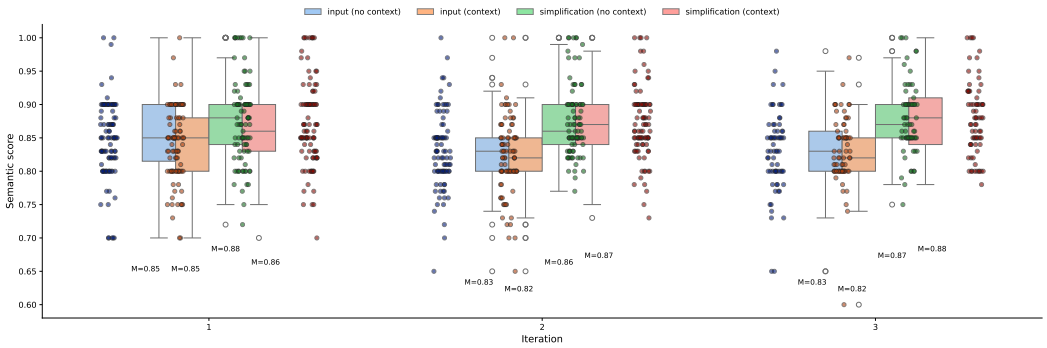

Figure 6. Distribution of simplification scores for the common relationships per iteration. Each dot is a data point showing the score at a specific iteration for a given configuration and condition.

Interestingly, the simplicity scores assigned to input segments were not constant across iterations. This is because the number of common relationships, those that appear in both the input and the simplification, changed from one iteration to the next. The validation component evaluates simplicity only for relationships that persist in both the original and simplified versions, so when more relationships are preserved (as seen in later iterations), the system evaluates a broader and often more representative subset of the input. As a result, variations in input scores across iterations reflect shifting evaluation targets, not changes to the input text itself.

The simplified outputs, meanwhile, maintained consistently high scores across iterations, demonstrating that the pipeline continued to generate accessible text throughout the refinement process. Moreover, these increases in simplicity scores were accompanied by a rise in the number of common relationships, particularly in the context condition (as seen in the previous section). This alignment between improved simplicity and enhanced semantic preservation suggests that the system is not simply shortening or rephrasing text, but is doing so while maintaining the structural and relational integrity of the original meaning. In other words, higher simplicity scores combined with more common relationships indicate not only simpler outputs, but also higher-quality simplifications.

7.5. Semantic-Retention Analysis

To better understand how semantic content is preserved or lost during sentence simplification, we analyzed the retention of relation clusters grouped by their semantic themes. Our goal was to identify which types of relations tend to be maintained versus omitted, offering insight into the priorities of simplification systems. The methodology consists of a multi-step process designed to assess how semantic content is preserved or lost during sentence simplification. First, relationships were extracted from both original and simplified sentences in the form of structured triples, which were organized on a per-sentence basis. Next, all unique relation labels were normalized (e.g., by replacing underscores with spaces) and embedded using a sentence-transformer model to capture their semantic similarity. Specifically, relation labels were embedded using the all-MiniLM-L6-v2 model from the Sentence-Transformers library [50], which provides lightweight and semantically meaningful sentence-level embeddings suitable for clustering tasks. These embeddings were then clustered using KMeans, with the optimal number of clusters determined via the Elbow Method and Silhouette Score. Each relation label was assigned to a semantic cluster, allowing sentences to be represented by sets of cluster IDs. For every original–simplified sentence pair, semantic clusters were compared to identify which were retained or dropped. For a given cluster, its retention rate was computed as follows:

$$\textit{Retention Rate} = \#\text{Retained}/(\#\text{Retained} + \#\text{Missing})$$

Here, a cluster is marked as retained for a sentence if it appears in both the input and simplified version; it is missing if it appears only in the input. This allowed us to compute how often each semantic cluster was preserved across the dataset.

Table 2 presents a contrast between the most consistently retained clusters (retention rate ≥ 0.75) and those that were never retained (retention rate = 0.0). Highly retained clusters tend to convey essential content such as purpose (e.g., USES, UTILIZATION), attribution (e.g., FOUNDED, DISCOVERED), or public relevance (e.g., POPULAR). These relations are likely considered central to sentence meaning and are preserved even in simplified forms.

Conversely, clusters with zero retention typically include auxiliary or peripheral content. This includes alternative names (e.g., ALSO_KNOWN_AS), descriptive elaboration (e.g., EXPLAINS, NARRATES), or collaborative context (e.g., TOGETHER_WITH). These are frequently omitted, likely due to their limited impact on core sentence understanding.

This analysis confirms that simplification systems prioritize propositional content while discarding contextual or supporting details, a finding consistent with previous work on simplification and information loss [2]. However, in certain domains (such as education, legal reasoning, or cultural studies), these omitted details may carry critical meaning. As a direction for future work, simplification models could benefit from semantic guardrails that dictate which types of relations should be preserved or excluded based on task requirements. For example, prompts or system-level constraints could be used to enforce the retention of attributional or causal relations (e.g., DISCOVERED_BY, CAUSE) in educational contexts such as Wikipedia articles, which often aim to balance clarity with comprehensive, sourced information. Developing domain-aware simplification pipelines with such guardrails would help maintain both accessibility and semantic fidelity, enabling more responsible and adaptive text generation.

Table 2. Semantic themes in relation clusters with high (≥0.75) or zero (0.0) retention in simplification.

Semantic Theme	Retention	Description
Cultural Relevance	High	Indicates public recognition or popularity (e.g., POPULAR, POPULAR_IN).
Attribution/Origin	High	Credits creators or originators of concepts (e.g., FOUNDED, DISCOVERED, AUTHOR).
Purpose/Function	High	Explains real-world use or application (e.g., USES, UTILIZATION, PRESCRIBE).
Intentional Control	High	Highlights agency in change or creation (e.g., MODIFIES, CONTROLS, DESIGNS).
Motivational Dynamics	High	Expresses cause or encouragement (e.g., PROMOTE, FACILITATE, TRIGGERS).
Alias Alternate Naming	Zero	Describes alternative names or labels (e.g., ALSO_KNOWN_AS, NICKNAMED).
Descriptive Elaboration	Zero	Adds narrative framing or detail (e.g., DESCRIBES, EXPLAINS, NARRATES).
Passive Acquisition	Zero	Involves receiving or inheriting something (e.g., WINS, INHERITS, RECEIVES).
Collaboration/Context	Zero	Specifies joint involvement or cooperation (e.g., WORK_WITH, TOGETHER_WITH).
Physical Enclosure	Zero	Indicates location within something (e.g., CONTAINED_IN, WRAPPED_IN).

8. Conclusions

This work presents a modular, context-aware pipeline for synthetic simplification of text in French, combining LLMs with symbolic semantic guidance through lightweight knowledge graphs. The system uses LangChain's orchestration tools to coordinate a multi-step simplification workflow that integrates document-level summaries, extracted semantic relations, and iterative feedback.

Central to the pipeline is the use of semantic triples that enable interpretable tracking of key relationships throughout the simplification process. These triples are used to compare the source and simplified versions, identify missing or altered content, and trigger regeneration when essential meaning is lost. Alongside the knowledge graphs, contextual information such as running summaries and key ideas guide the model toward preserving content while improving accessibility.

The results show that iterative prompting improves both semantic fidelity and simplicity across steps. Specifically, simplification quality, as measured by automatic scores on semantic relation coverage and LLM-based simplicity evaluations, tends to improve in each successive iteration. While gains are modest, they reflect a consistent trend toward better balance between maintaining key information and producing more accessible outputs. The combination of feedback and contextual grounding proves effective in steering the model away from oversimplification or semantic loss.

While our simplification pipeline was developed and evaluated in French, its underlying components are largely language-agnostic, making adaptation to low-resource languages feasible. The core LLMs used for simplification and relationship extraction are pretrained on diverse multilingual corpora, enabling basic transfer to other languages. However, the coverage and quality of these models can vary substantially. Languages that are underrepresented in pretraining data may yield less reliable outputs, particularly for tasks requiring nuanced semantic understanding. In addition, evaluation metrics such as SARI and BERTScore rely on the availability of reference simplifications and

high-quality embeddings in the target language, which may not always be accessible for low-resource scenarios.

Another parameter that requires consideration is the number of iterations, which is currently limited to three in our setup. This decision was based on prior experience showing that beyond a certain point, additional refinement leads to diminishing returns or oscillations in quality. Future work could explore adaptive control mechanisms that monitor convergence signals or quality metrics to determine when to stop or continue refinement.

A notable limitation of our system is the simplicity of the extracted knowledge graphs. Though they are lightweight and interpretable, these graphs may omit subtle or implicit semantic information. Future versions could incorporate more expressive representations, including temporal, coreferential, or hierarchical structures, to improve both generation and evaluation. We can consider enriching the KGs with external ontologies or structured commonsense resources such as ConceptNet [51]. These could be used to infer implicit links, expanding the graph beyond what is explicitly stated in the text. For instance, if a simplification omits a concept that is not directly mentioned but is contextually entailed, such augmentation could help detect it. Another promising solution is to augment KG representations with semantic parsing techniques. In particular, Semantic Role Labeling (SRL) has advanced significantly in recent years thanks to contextualized embeddings and pretrained transformers. Modern SRL models (such as [52]) accurately extract predicate–argument structures and offer a more compositional view of meaning than flat triples do. Similarly, Abstract Meaning Representation (AMR) parsing (e.g., [53]) offers another avenue for deeper semantic analysis. AMR graphs encode full-sentence meaning as structured, language-independent graphs, enabling more nuanced comparisons between the input and its simplification.

Another constraint is the reduced scale of the evaluation due to API usage costs. Although the full dataset contains roughly 10,000 aligned sentence pairs, only a subset was analyzed. Broader experiments could be conducted with more efficient prompting techniques or via local model deployment, allowing deeper analysis and broader generalization.

Furthermore, our evaluation did not involve human annotators. While LLM-generated scores and structure-based comparison provide scalable signals, they cannot fully capture aspects like fluency, appropriateness for children, or stylistic alignment with Vikidia. Including human feedback in future work would offer valuable validation.

Finally, while our pipeline relies on in-context semantic prompts, future versions may benefit from Retrieval-Augmented Generation (RAG) [54], especially in low-resource or domain-specific scenarios. By grounding generation in relevant external documents, RAG could further enhance the factuality and coherence of simplifications.

Funding: This research was funded by Swiss National Science Foundation grant number 197864.

Data Availability Statement: The code is available at the GitLab Repository referred in the paper. Full data will be made available upon reasonable request.

Conflicts of Interest: The author declares no conflicts of interest.

Appendix A

This appendix includes all prompt templates used throughout this work. In each case, variables passed to the language model are highlighted in blue. We employed English-language prompt templates to interact with the underlying LLMs, as many state-of-the-art options (especially those with strong multilingual capabilities) are trained with extensive English corpora and tend to exhibit more stable and coherent behavior when prompted in English.

Table A1. Prompt template used in the progressive summarization step.

You are maintaining a progressive summary of a document. This summary evolves over time and must remain complete and coherent for future reference. The knowledge base should be updated based on new information, while preserving all previously included content. Ensure the result remains understandable and contextually accurate. Keep the summary compact yet informative. New information should be logically integrated, drawing from the following sources: main ideas and the current summary. Current Knowledge Base: {info_base} Update Format Instructions: {format_instructions} New Content Chunk: {input} Important: The response must be written entirely in French.

Table A2. Prompt template used in the plain simplification step.

Simplify the input text into French, using Vikidia style (clear, simple, factual, age-appropriate).
Your simplification should be accurate, complete, and limited to what is in the input text.
Input Text: {input}
Provide the output in JSON format with the following structure:
{
“simplified_text”: “string-The simplified version of the input text.”
}

Table A3. Prompt template used for comparing relationships between original and simplified texts.

Given two lists of relationships—one from the original text and one from a simplified version—identify:
1. The relationships that are semantically equivalent across both lists, even if the wording, entity names, or phrasing differ slightly. Do not consider differences in language.
2. The relationships from the original text that are missing in the simplified version, meaning that their semantic content is not represented in any form in the simplified list.
Each relationship consists of a source, target, and relation type.
Input Relationships: {input_relationships}
Simplified Relationships: {simplified_relationships}
Provide the output in valid JSON with the following structure:
{
“common_relationships”: [
{ “source”: “string”, “target”: “string”, “relation”: “string” }
],
“missing_relationships”: [
{ “source”: “string”, “target”: “string”, “relation”: “string” }
]
}

Table A4. Prompt template used to evaluate relationship-level simplicity.

Extract the most relevant text segments from the input text for each of the following relationships. Then, evaluate and provide a simplicity score from 0 to 1 (where 1 is very simple and 0 is very complex) for each relationship.

Input Text: {input}.

Relationships: {`relationships`}

Provide the output in JSON format with the following structure:

```
{
  "relations_simplicity": [
    {
      "source": "string-The source entity",
      "target": "string-The target entity",
      "relation": "string-The type of relationship",
      "text_segment": "string-The extracted text segment related to this relationship",
      "simplicity_score": "number-A score from 0 to 1 indicating the simplicity of the segment"
    }
  ]
}
```

Table A5. Prompt template used for evaluating the simplicity of two candidate simplifications.

You are a language expert evaluating how well two candidate simplifications simplify a French text.

You are provided with:

- The original input text
- Two candidate simplifications (A and B)

Evaluate and rank the two candidates based only on how simple and accessible they are for young readers.

Input Text: {input}

Candidate Simplifications: {candidates}

Provide the output in JSON format with the following structure:

```
{
  "simplification_ranking": {
    "ranking": ["A", "B"], // from simplest to least simple
    "justification": "A brief explanation focused on simplicity and accessibility for young readers"
  }
}
```

Table A6. Prompt template used for refining a previously simplified text using simplicity feedback and missing semantic content.

The previous simplification did not fully meet the requirements.

Simplify the input text into French, using Vikidia style (clear, simple, factual, age-appropriate).

You are provided with:

- Input Text
- Candidate A: The first simplification attempt
- Candidate B: The most recent simplification (which needs improvement)
- Feedback comparing A vs. B
- A list of semantic relationships missing from B
- The context, knowledge graphs, summary, and main ideas for interpretation only

Table A6. *Cont.*

Do not add any facts not explicitly stated in the input. Revise B, improving its simplicity as guided by the feedback, and ensure all missing relationships are addressed.
Input Text: {input}
Candidate A: {simplified_text_plain}
Candidate B: {simplified_text}
Feedback on Simplicity: {simplicity_justification}
Input Knowledge Graph: {input_graph}
Context: {context}
Context Knowledge Graph: {context_graph}
Summary: {summary}
Main Ideas: {main_ideas}
 	
Missing Relationships in B: {missing_relationships}
 	
Your simplification should be accurate, complete, and limited to what is in the input text.
Please revise the previous simplified text, taking into account both the missing content and the simplicity feedback.
Provide the output in JSON format with the following structure:
{
"corrected_text": "string-The corrected simplified text ensuring all missing nodes and relationships are included."
}

References

1. Aluísio, S.M.; Specia, L.; Pardo, T.A.; Maziero, E.G.; Fortes, R.P. Towards Brazilian Portuguese Automatic Text Simplification Systems. In Proceedings of the 8th ACM Symposium on Document Engineering, Sao Paulo, Brazil, 16–19 September 2008; pp. 240–248.
2. Siddharthan, A. A Survey of Research on Text Simplification. *ITL-Int. J. Appl. Linguist.* **2014**, *165*, 259–298. [CrossRef]
3. Hasler, E.; de Gispert, A.; Stahlberg, F.; Waite, A.; Byrne, B. Source Sentence Simplification for Statistical Machine Translation. *Comput. Speech Lang.* **2017**, *45*, 221–235. [CrossRef]
4. Vickrey, D.; Koller, D. Sentence Simplification for Semantic Role Labeling. In Proceedings of the ACL-08: HLT, Columbus, OH, USA, 16–18 June, 2008; pp. 344–352.
5. Ormaechea, L.; Tsourakis, N. Automatic Text Simplification for French: Model Fine-Tuning for Simplicity Assessment and Simpler Text Generation. *Int. J. Speech Technol.* **2024**, *27*, 957–976. [CrossRef]
6. Ormaechea, L.; Tsourakis, N.; Schwab, D.; Bouillon, P.; Lecouteux, B. Simple, Simpler and Beyond: A Fine-Tuning BERT-Based Approach to Enhance Sentence Complexity Assessment for Text Simplification. In Proceedings of the International Conference on Natural Language and Speech Processing, Trento, Italy, 16–17 December, 2023.
7. Le Scao, T.; Fan, A.; Akiki, C.; Pavlick, E.; Ilić, S.; Hesslow, D.; Castagné, R.; Luccioni, A.S.; Yvon, F.; Gallé, M.; et al. Bloom: A 176b-Parameter Open-Access Multilingual Language Model. *arXiv* **2023**, arXiv: 2211.05100. [CrossRef]
8. OpenAI. GPT-4 Technical Report. 2023. Available online: https://openai.com/research/gpt-4 (accessed on 3 July 2025).
9. Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. Self-Refine: Iterative Refinement with Self-Feedback. In *Proceedings of the Advances in Neural Information Processing Systems*; Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S., Eds.; Curran Associates, Inc.: New Orleans, LA, USA, 2023; Volume 36, pp. 46534–46594.
10. Long, L.; Wang, R.; Xiao, R.; Zhao, J.; Ding, X.; Chen, G.; Wang, H. On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey. *arXiv* **2024**, arXiv:2406.15126.
11. Reynolds, L.; McDonell, K. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In Proceedings of the Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; pp. 1–7.
12. Chung, J.; Kamar, E.; Amershi, S. Increasing Diversity While Maintaining Accuracy: Text Data Generation with Large Language Models and Human Interventions. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Toronto, ON, Canada, 9–14 July 2023; Rogers, A., Boyd-Graber, J., Okazaki, N., Eds.; ACL: Vienna, Austria, 2023; pp. 575–593. [CrossRef]

13. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Yu, P.S. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 494–514. [CrossRef] [PubMed]
14. Yu, W.; Zhu, C.; Li, Z.; Hu, Z.; Wang, Q.; Ji, H.; Jiang, M. A Survey of Knowledge-Enhanced Text Generation. *ACM Comput. Surv.* **2022**, *54*, 1–38. [CrossRef]
15. Zheng, L.; Chiang, W.L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. Judging LLL-as-a-Judge with MT-Bench and Chatbot Arena. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 46595–46623.
16. Carroll, J.A.; Minnen, G.; Pearce, D.; Canning, Y.; Devlin, S.; Tait, J. Simplifying Text for Language-Impaired Readers. In Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics, Bergen, Norway, 8–12 June 1999; pp. 269–270.
17. Siddharthan, A. An Architecture for a Text Simplification System. In Proceedings of the Language Engineering Conference, Hyderabad, India, 13–15 December 2002; pp. 64–71.
18. Devlin, S. The Use of a Psycholinguistic Database in the Simplification of Text for Aphasic Readers. *Linguist. Databases* **1998**, *77*, 161–172.
19. Nisioi, S.; Štajner, S.; Ponzetto, S.P.; Dinu, L.P. Exploring Neural Text Simplification Models. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 85–91.
20. Xu, W.; Callison-Burch, C.; Napoles, C. Problems in Current Text Simplification Research: New Data Can Help. *Trans. Assoc. Comput. Linguist.* **2015**, *3*, 283–297. [CrossRef]
21. Zhang, X.; Lapata, M. Sentence Simplification with Deep Reinforcement Learning. *arXiv* **2017**, arXiv:1703.10931. [CrossRef]
22. Mallinson, J.; Lapata, M. Controllable Sentence Simplification: Employing Syntactic and Lexical Constraints. *arXiv* **2019**, arXiv:1910.04387. [CrossRef]
23. Maddela, M.; Xu, W.; Preotiuc-Pietro, D. Controllable Text Simplification with Explicit Paraphrasing. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Punta Cana, Dominican Republic, 7–11 November 2021; Association for Computational Linguistics: Punta Cana, Dominican Republic, 2021; pp. 3512–3526.
24. Tsourakis, N. *Machine Learning Techniques for Text: Apply Modern Techniques with Python*; Packt Publishing Ltd.: Birmingham, UK, 2022.
25. Jamet, H.; Shrestha, Y.R.; Vlachos, M. Difficulty Estimation and Simplification of French Text Using LLMs. In *Generative Intelligence and Intelligent Tutoring Systems*; Springer Nature: Cham, Switzerland, 2024; pp. 395–404. [CrossRef]
26. Gala, N.; Tack, A.; Javourey-Drevet, L.; François, T.; Ziegler, J.C. Alector: A parallel Corpus of Simplified French Texts with Alignments of Misreadings by Poor and Dyslexic Readers. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; pp. 1353–1361.
27. Martin, L.; Fan, A.; De La Clergerie, E.; Bordes, A.; Sagot, B. MUSS: Multilingual Unsupervised Sentence Simplification by Mining Paraphrases. *arXiv* **2020**, arXiv:2005.00352.
28. Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N.A.; Khashabi, D.; Hajishirzi, H. Self-Instruct: Aligning Language Models with Self-Generated Instructions. *arXiv* **2022**, arXiv:2212.10560.
29. Sahu, G.; Vechtomova, O.; Laradji, I.H. A Guide To Effectively Leveraging LLMs for Low-Resource Text Summarization: Data Augmentation and Semi-Supervised Approaches. *arXiv* **2025**, arXiv:2407.07341.
30. Li, Z.; Wang, X.; Zhao, J.; Yang, S.; Du, G.; Hu, X.; Zhang, B.; Ye, Y.; Li, Z.; Zhao, R.; et al. PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-Consistency. *arXiv* **2024**, arXiv:2403.09732.
31. Akella, A.; Manatkar, A.; Chavda, B.; Patel, H. An Automatic Prompt Generation System for Tabular Data Tasks. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies Volume 6: Industry Track, Mexico City, Mexico, 16–21 June 2024; pp. 191–200. [CrossRef]
32. Chai, Y.; Xie, H.; Qin, J.S. Text Data Augmentation for Large Language Models: A Comprehensive Survey of Methods, Challenges, and Opportunities. *arXiv* **2025**, arXiv:2501.18845.
33. Sufi, F. Addressing Data Scarcity in the Medical Domain: A GPT-Based Approach for Synthetic Data Generation and Feature Extraction. *Information* **2024**, *15*, 264. [CrossRef]
34. Guo, X.; Chen, Y. Generative AI for Synthetic Data Generation: Methods, Challenges and the Future. *arXiv* **2024**, arXiv:2403.04190. [CrossRef]
35. Liu, N.F.; Zhang, T.; Liang, P. Evaluating Verifiability in Generative Search Engines. *arXiv* **2023**, arXiv:2304.09848. [CrossRef]
36. Zhao, S.; Meng, R.; He, D.; Andi, S.; Bambang, P. Integrating Transformer and Paraphrase Rules for Sentence Simplification. *arXiv* **2018**, arXiv:1810.11193. [CrossRef]
37. Alva-Manchego, F.; Scarton, C.; Specia, L. Data-Driven Sentence Simplification: Survey and Benchmark. *Comput. Linguist.* **2020**, *46*, 135–187. [CrossRef]
38. Gilardi, F.; Alizadeh, M.; Kubli, M. ChatGPT Outperforms Crowd Workers for Text-Annotation Tasks. *Proc. Natl. Acad. Sci. USA* **2023**, *120*, e2305016120. [CrossRef] [PubMed]

39. Ding, B.; Qin, C.; Zhao, R.; Luo, T.; Li, X.; Chen, G.; Xia, W.; Hu, J.; Luu, A.T.; Joty, S. Data Augmentation using Large Language Models: Data Perspectives, Learning Paradigms and Challenges. *arXiv* **2024**, arXiv:2403.02990.
40. Ormaechea, L.; Tsourakis, N.; Bouillon, P.; Lecouteux, B.; Schwab, D. Towards High-Quality LLM-Based Data for French Spontaneous Speech Simplification: An Exo-Refinement Approach. In Proceedings of the Interspeech, Rotterdam, The Netherlands, 17–21 August 2025.
41. Huang, J.; Chen, X.; Mishra, S.; Zheng, H.S.; Yu, A.W.; Song, X.; Zhou, D. Large Language Models Cannot Self-Correct Reasoning Yet. *arXiv* **2023**, arXiv:2310.01798.
42. Pan, S.; Luo, L.; Wang, Y.; Chen, C.; Wang, J.; Wu, X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 3580–3599. [CrossRef]
43. LangChain. Summarization Use Cases (Python). 2024. Available online: https://python.langchain.com/v0.1/docs/use_cases/summarization/ (accessed on 3 July 2025).
44. LangChain. RefineDocumentsChain (JavaScript). 2024. Available online: https://js.langchain.com/v0.1/docs/modules/chains/document/refine/ (accessed on 3 July 2025).
45. LangChain. MapReduceDocumentsChain for Summarization. 2023. Available online: https://python.langchain.com/docs/how_to/summarize_map_reduce/ (accessed on 3 July 2025).
46. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 6–12 July 2002; pp. 311–318.
47. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of Summaries. In Proceedings of the Text Summarization Branches Out, Barcelona, Spain, 25–26 July 2004; pp. 74–81.
48. Xu, W.; Napoles, C.; Pavlick, E.; Chen, Q.; Callison-Burch, C. Optimizing Statistical Machine Translation for Text Simplification. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 401–415. [CrossRef]
49. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. BERTScore: Evaluating Text Generation with BERT. *arXiv* **2019**, arXiv:1904.09675.
50. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Hong Kong, China, 3–7 November 2019.
51. Speer, R.; Chin, J.; Havasi, C. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *arXiv* **2018**, arXiv:1612.03975. [CrossRef]
52. Shi, P.; Lin, J. Simple BERT Models for Relation Extraction and Semantic Role Labeling. *arXiv* **2019**, arXiv:1904.05255. [CrossRef]
53. Bai, X.; Chen, Y.; Zhang, Y. Graph Pre-Training for AMR Parsing and Generation. *arXiv* **2022**, arXiv:2203.07836. [CrossRef]
54. Gupta, S.; Ranjan, R.; Singh, S.N. A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions. *arXiv* **2024**, arXiv:2410.12837.