LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

STUDIENGANG INFORMATIK &
STUDIENGANG COMPUTERLINGUISTIK

Masterarbeit

in den Studiengängen Informatik und Computerlinguistik

an der Ludwig-Maximilians-Universität München

Fakultät für Mathematik, Informatik und Statistik

sowie

Fakultät für Sprach- und Literaturwissenschaften

A Memory-Based Method for Improving Long-Context LLM-as-a-Judge Assessments

vorgelegt von
Hermine Kleiner

Betreuer:	Ali Modarressi (M.Sc.)
Prüfer:	Prof. Dr. Hinrich Schütze
Bearbeitungszeitraum:	11.03.2025 – 29.07.2025

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, den 29.07.2025

. . . .
Hermine Kleiner

Statement of AI Usage

In this work, we have utilized generative AI tools to assist in certain aspects of content creation, including research, writing, and coding. Specifically, ChatGPT, NotebookLM, Grammarly and DeepL were employed to help with:

- implementation of baseline code structures, e.g., of the statistical tests used
- debugging of code
- translation of words and phrases into English
- detecting typos, syntactical and wording issues while writing
- providing explanations, summarization and examples for complex concepts used in scientific papers and other works
- rewriting of complex or hard-to-read phrases and sentences
- review LaTeX-commands and format diverse content into LaTeX-tables

These AI tools were used as aids to improve productivity and efficiency but did not replace critical thinking or the application of our own knowledge.

We explicitly disclose that the use of AI tools is by academic integrity guidelines, and have ensured that all AI-generated content is reviewed, verified, and fact-checked to avoid inaccuracies, errors, or “hallucinations” in the information provided.

In line with the policy, we have appropriately cited all external sources of information, including those suggested by the AI tools.

Munich, 29.07.2025

. . . .
Hermine Kleiner

Abstract

As Large Language Models (LLMs) are increasingly adopted as automated evaluators (so-called LLMs-as-a-Judge) their ability to evaluate long-context inputs remains underexplored and underperforming. While technical advances have extended LLM context windows to over 100,000 tokens, empirical studies reveal significant performance degradation when reasoning across long sequences. This thesis addresses this gap by introducing JUDGEMEMO, a structured memory-augmented evaluation framework inspired by human note-taking, designed to enhance LLM assessment reliability on long-form documents. We also present COFLUEVAL-LC, a novel diagnostic dataset built from Project Gutenberg texts, systematically manipulated to degrade fluency and coherence across varied structural and granular levels. Through extensive experiments, we show that traditional LLM-as-a-Judge setups struggle with long-context consistency and exhibit reduced sensitivity to subtle quality manipulations. In contrast, JUDGEMEMO significantly improves evaluation performance and robustness by segmenting documents into smaller sub-units, generating intermediate evaluations (*memory*), and feeding this structured memory back into the evaluation loop. Our results demonstrate that memory-augmented setups can mitigate the contextual limitations of LLMs, offering a more human-aligned, scalable path toward reliable automatic evaluation of long-form content.

Contents

1 Introduction

Background & Motivation Large Language Models have become a central technology in both research and everyday applications. Popular systems like ChatGPT from OpenAI (OpenAI et al., 2024), and Gemini (Gemini Team et al., 2025) showcase their wide adoption and utility in today's society. Beyond traditional natural language processing (NLP) tasks such as summarization, translation, and text generation, LLMs are increasingly deployed in interactive applications like conversational agents, intelligent search engines, and question answering systems.

Many of these real-world use cases require LLMs to process long and complex input sequences ranging from multi-turn dialogues to entire documents or books (Yuan et al., 2024). To meet this demand, significant progress has been made in extending LLMs' context windows. Innovations such as advanced positional encodings (Sun et al., 2023) and efficient methods like YaRN (Peng et al., 2023) have enabled models to handle inputs of up to 128k tokens without retraining. Extending context length is now widely recognized as a key direction in LLM development (An et al., 2024).

However, while modern LLMs can technically process long sequences, it remains unclear how well they actually retain and use relevant information across such inputs. Several studies have shown that model performance often degrades significantly, even before reaching the technical input limit (Modarressi et al., 2025a; Levy et al., 2024). Moreover, evaluations of long-context capabilities have largely focused on narrow task types such as question answering, summarization, or retrieval (An et al., 2024; Bai et al., 2024; Yuan et al., 2024), leaving many other real-world scenarios underexplored.

One emerging use case that especially highlights these limitations is the use of LLMs as evaluators - so-called LLM-as-a-Judge setups - where models assess the quality of generated content. This approach is promising due to its scalability and efficiency (Chiang and Lee, 2023; Wang et al., 2024). However, most current evaluation workflows do not account for the challenges posed by long or complex inputs. As these resulting prompts become more structured and lengthy, limitations in processing and contextual understanding can compromise the reliability and trustworthiness of the assessments (Gu et al., 2024).

Research Problem While LLM-as-a-Judge setups offer a cost-effective alternative to human evaluation (Chiang and Lee, 2023), existing studies primarily focus on short-context scenarios, such as evaluating summaries or binary correctness judgements (Zheng et al., 2023). This leaves a significant gap in understanding how well LLMs perform in evaluating long-form content or documents that require sustained attention and nuanced understanding across multiple quality dimensions. Unlike humans, who externalize memory by taking notes and aggregating judgements, current LLMs attempt to handle long inputs in a single forward pass, often leading to issues in recalling facts as well as contextual and temporal degradation of knowledge (Mallen et al., 2023).

In this work, we address this gap by proposing a memory-augmented framework for LLM-as-a-Judge evaluations that enables LLM to truly process long documents and therefore, enhances reliability, especially in long-context scenarios. Inspired by how humans evaluate long documents, for instance, through multiple reading passes, note-taking, and synthesizing key points, our approach leverages long-context decomposition and structured memory to guide the evaluation process. Instead of forcing the model to retain all context internally, we provide it with externally stored intermediate *section-wise reports* to guide and support the final assessment.

Research Questions (RQ) Building on this motivation, we pursue a systematic exploration of LLM-based evaluations in long-context scenarios and, accordingly, investigate the following research questions to assess their limitations and potential:

- **RQ1**: Is there a significant difference in evaluation performance between short-context and long-context setups when keeping all other parameters stable except for the input document length?
- **RQ2**: Can a structured memory mechanism mitigate the limitations of long-context inputs and improve the reliability and consistency of LLM-as-a-Judge evaluations?

These questions aim to uncover both the extent of long-context degradation and the effectiveness of memory augmentation in real-world evaluation settings.

Contributions To examine our research questions, our contributions are two-fold:

1. COFLUEVAL-LC: We introduce a novel diagnostic dataset designed to evaluate LLM-as-a-Judge performance under long-context conditions. COFLUEVAL-LC includes a gold-standard reference set and a manipulated set, where subtle yet detectable changes affect two core evaluation dimensions: fluency and coherence.
2. JUDGEMEMO: We propose a memory-augmented evaluation framework that mirrors human reading and assessment strategies. JUDGEMEMO breaks down long documents into smaller segments, generates intermediate notes, and uses these structured issue-reports as in-context memory to support final evaluations. This modular pipeline helps the LLM maintain context and deliver more consistent assessments over lengthy inputs.

By integrating memory into the evaluation process, we not only address the weaknesses of long-context inference but also introduce a framework that better aligns with human evaluative behaviour. Our dataset and method can serve as a foundation for future research on reliable automatic evaluation of long-form content.

For evaluation, we conduct a statistical analysis of the generated scorings. Our experiments demonstrate that standard LLM-as-a-Judge setups struggle with long-context inputs, exhibiting reduced sensitivity to subtle quality degradations in fluency and, especially, coherence. In contrast, our proposed memory-augmented framework, JUDGEMEMO, significantly improves evaluation accuracy and consistency by segmenting long documents and integrating structured memory. These findings support the hypothesis that memory-augmented approaches can mitigate long-context limitations and enhance the overall reliability of LLM-based evaluations.

Thesis Structure This thesis is structured in alignment with our core contributions. We begin by providing an overview of related work and theoretical foundations in Chapter 2. This chapter addresses key questions such as: What does LLM-as-a-Judge mean? What dimensions can be evaluated by LLMs? What are the challenges in LLM-based evaluation? Why is long-context processing difficult? And how can memory mechanisms help overcome these challenges?

Chapter 3 focuses on our diagnostic dataset, COFLUEVAL-LC. We detail the data collection process, dataset construction, manipulation techniques, and intermediate experiments used to identify the most effective manipulations incorporated into COFLUEVAL-LC.

In Chapter 4, we introduce our second major contribution, JUDGEMEMO, a memory-based framework designed to improve long-context evaluations with LLMs. This chapter includes a brief excursus into our prompt engineering efforts, followed by an in-depth discussion of the method's motivation and architecture.

Chapter 5 presents our experimental setup, ablation studies, and key results. We perform statistical analysis for single-prompt LLM-as-a-Judge setups as well as for our own method, JUDGEMEMO. Moreover, we test our framework under different settings, examining reporting strategies, scanning modes, and the influence of summaries.

Finally, Chapter 6 and Chapter 7 summarize our findings, discuss limitations, and outline directions for future work.

2 Related Work

This chapter surveys research on using large language models as evaluators, particularly in the context of assessing long-form content. It highlights prior work on LLM-as-a-Judge, outlines the challenges they face in handling long contexts, and reviews approaches that incorporate memory mechanisms to address these limitations. These areas directly inform the motivation and design of the proposed JUDGEMEMO framework.

2.1 LLMs as Evaluators

The concept of Large Language Models as a Judge (LLM-as-a-Judge) refers to the use of powerful LLMs to perform evaluation and assessment tasks traditionally conducted by human experts (Gu et al., 2024). This emerging paradigm leverages the models' impressive capabilities in human-like reasoning, instruction following, and determining whether specific inputs meet predefined criteria (Gu et al., 2024; Chiang and Lee, 2023). It offers a promising alternative to conventional evaluation methods, aiming to combine the scalability and consistency of automated tools with the nuanced, context-sensitive reasoning of human judgements (Gu et al., 2024), while also being more cost-effective and efficient (Chiang and Lee, 2023).

LLM-as-a-Judge is applied in a broad range of scenarios in natural language processing and beyond, including evaluation of other LLMs (cf. Section 2.4), data annotation, agent evaluation, reasoning and problem-solving, open-ended task assessments, and classical NLP evaluations such as summarization (Wang et al., 2024; Gu et al., 2024; An et al., 2024; Liu et al., 2024d; Lin and Chen, 2023; Hu et al., 2024).

The LLM-as-a-Judge evaluation process typically involves three key stages: input design (prompting), model selection, and output post-processing (Chiang and Lee, 2023; Gu et al., 2024). Each stage offers various configurations, ranging from input/output formats to scoring schemes and task structures (Li et al., 2024a).

2.2 Text Quality Metrics in LLM-Based Evaluation

LLM evaluations assess a variety of text quality metrics across different Natural Language Generation (NLG) tasks. While the terminology may differ, many metrics overlap significantly in what they measure. For instance, fluency, grammatically, and naturalness are often treated as distinct but frequently cover similar surface-level features such as grammar, spelling, and stylistic coherence (Celikyilmaz et al., 2020). Likewise, factuality, consistency, and reliability commonly target the model's ability to produce factually correct and non-contradictory information (Li et al., 2024a; Hu et al., 2024; Celikyilmaz et al., 2020).

Below, we summarize commonly used text quality metrics in LLM-based evaluations, along with their typical applications in various NLG tasks:

- **Grammaticality** and **Fluency** assess correctness and natural flow of the text. They are frequently used in summarization, translation, and dialogue generation (Celikyilmaz et al., 2020; Chiang and Lee, 2023; Liu et al., 2023; Lin and Chen, 2023; Hu et al., 2024).

- **Coherence** and **Consistency** measure the logical flow and internal alignment of ideas, especially in longer texts like stories or summaries (Liu et al., 2023; Lin and Chen, 2023; Hu et al., 2024; Celikyilmaz et al., 2020).
- **Relevance** and **Appropriateness** evaluate how well the output aligns with the input prompt or context (Chiang and Lee, 2023; Celikyilmaz et al., 2020; Hu et al., 2024).
- **Factuality** checks whether the output is accurate with respect to a reference source, often critical in summarization and QA (Chiang and Lee, 2023; Li et al., 2024a; Gao et al., 2023).
- **Helpfulness**, **Likability**, and **Harmlessness** capture subjective dimensions such as informativeness, engagement, and safety (Li et al., 2024a; Wang et al., 2024; Liu et al., 2024d).
- **Meaning Preservation** assesses whether the semantic intent remains unchanged after transformations, often used in adversarial settings (Chiang and Lee, 2023).
- **Overall Quality** or **Preference** provide holistic or comparative judgements, often combining several of the above criteria (Celikyilmaz et al., 2020; Liu et al., 2024c; Zhang et al., 2023).

Despite the diversity in naming, many of these metrics share overlapping definitions and are applied inconsistently across tasks. Our evaluation framework addresses this ambiguity by offering clearly defined metric descriptions, scoring rubrics, and consistent use across different evaluation scenarios.

2.3 Challenges in LLM Evaluations

The use of LLMs as judges for evaluation tasks offers substantial potential as a scalable and adaptable solution (Gu et al., 2024) but also introduces several important challenges:

First, LLMs may conflate distinct evaluation criteria, often producing similar scores across dimensions that human raters would clearly distinguish (Hu et al., 2024). This issue is partly due to ambiguous phrasing and inconsistent conceptualization of evaluation aspects, making it difficult for both humans and LLMs to fully understand the criteria and their interrelationships (Hu et al., 2024). Liu et al. (2024d) address this challenge by proposing a novel framework that iteratively aligns LLM-based evaluations with human preferences through Hierarchical Criteria Decomposition, where evaluation tasks are broken down into finer-grained criteria.

We adopt a similar philosophy, but instead of decomposing the evaluation criteria, we primarily decompose the document to be evaluated into smaller, more manageable units. This allows the model to focus more effectively on the relevant content. Furthermore, to mitigate metric confusion, we provide carefully designed definitions for each text quality metric, accompanied by detailed scoring guidelines within our evaluation setup.

Second, the use of numerical rating scales may fail to capture the subtleties of human judgement (Chiang and Lee, 2023). Just as human annotators may interpret scales differently, LLMs also exhibit subjective tendencies. For instance, they may disagree on whether punctuation errors qualify as grammatical issues (Chiang and Lee, 2023). While we adopt a Likert-style numerical scale in our framework, we additionally incorporate free-text justifications and offer explicit instructions about the meaning of each score.

Another technical issue is the challenge of parsing LLM outputs. Due to varied and sometimes inconsistent formatting, extracting reliable scores often requires manual inspection or complex rule-based processing (Chiang and Lee, 2023; Gu et al., 2024). While this issue cannot be entirely eliminated, we mitigate it by introducing a structured prompt design with clear output instructions, which provides a consistent and parsable format.

Moreover, LLMs are also prone to *hallucinations*: They may generate inaccurate or nonsensical content that appears plausible to its reader (Chiang and Lee, 2023). Furthermore, they can produce inconsistent results for the same input, a phenomenon known as the *self-consistency problem* (Chiang and Lee, 2023). This is often worsened by high sensitivity to prompt wording (Chiang and Lee, 2023; Liu et al., 2024d; Gu et al., 2024).

Finally, LLM-based evaluations are susceptible to biases learned from their training data, which can undermine fairness, reliability, and objectivity (Wang et al., 2024; Gu et al., 2024). These biases not only affect judgement quality but also raise broader ethical concerns about fairness, transparency, and accountability (Gu et al., 2024). This raises critical ethical and practical questions about the extent to which LLMs can or should replace human evaluators, particularly in high-stakes or subjective evaluations central to our research focus (Chiang and Lee, 2023; Gu et al., 2024).

2.4 Evaluating the Evaluators

As LLMs are increasingly used as automated judges in Natural Language Generation evaluation, ensuring their reliability and objectivity has become critical. Although prior work shows that LLM-based evaluations can match human-level performance in certain settings, concerns remain about their consistency, susceptibility to bias, and sensitivity to prompt design (Hu et al., 2024; Tan et al., 2024).

To address these challenges, recent research has proposed specialized evaluation pipelines. Tan et al. (2024) introduce a hierarchical evaluation framework and a benchmark (JudgeBench), which transforms existing datasets with ground truth labels into challenging response-pair tasks designed to test judgement accuracy. Their focus is on verifying whether LLM-based evaluators prioritize objective correctness over subjective preferences.

Similarly, Hu et al. (2024) propose a perturbation-based methodology that systematically alters text to target specific quality dimensions, revealing weaknesses in LLM evaluation behaviour. Their framework includes eleven well-defined NLG quality aspects and 18 carefully crafted perturbation attacks, forming the basis of our own manipulation techniques.

Despite these advances, a critical gap remains: None of these approaches explicitly examine how long-context inputs affect the reliability of LLM-as-a-Judge evaluations. Our work addresses this underexplored dimension by investigating how memory-based methods can improve LLM evaluation consistency in long-context scenarios.

2.5 LLMs and Long Context

Despite recent advances extending LLM context windows to hundreds of thousands of tokens, significant challenges persist in their ability to effectively comprehend and utilize long-context inputs (Li et al., 2024b; Levy et al., 2024). Increasing context length, e.g., through architectural techniques like YaRN (Peng et al., 2023), does not necessarily improve what Li et al. (2024b) call “true long-context understanding”. In fact, comprehensive evaluations repeatedly show that model performance degrades as input length increases, even when models are technically capable of accepting longer sequences (Bai et al., 2024; Levy et al., 2024; Zhang et al., 2024; Modarressi et al., 2025a).

Recent work has introduced several benchmarks to better evaluate these limitations. For example, LongBench (Bai et al., 2024) proposes a suite of long-context tasks, including summarization, question answering, and code completion, across both English and Chinese. However, many benchmarks focus on relatively moderate lengths. L-Eval (An et al., 2024) pushes this further, ranging from 3K to 2M tokens, and reveals significant performance gaps between commercial and open-source models. Li et al. (2024b) emphasize evaluating interdependent tasks on long, up-to-date documents, arguing that successful long-context processing requires more than simple span retrieval.

Pushing even further, ∞BENCH (Zhang et al., 2024) claims to be the first benchmark with an average input length exceeding 100K tokens. To prevent training data leakage, it introduces "fake novels" by systematically replacing key entities, allowing for clean evaluations of long-context understanding. Further evaluations such as LV-Eval (Yuan et al., 2024) introduce controlled setups to measure length sensitivity while accounting for potential memorization or leakage.

Levy et al. (2024) introduce FLenQA, a QA-style benchmark designed to isolate the effect of input length on reasoning ability. They find that reasoning accuracy drops sharply (e.g., from 92% to 68%) before reaching the model's maximum input length (already around 3K tokens). They also highlight the "Lost-in-the-Middle" phenomenon (Liu et al., 2024a), where models preferentially remember beginning and end segments but forget the middle content.

Modarressi et al. (2025a) critique current Needle-in-the-Haystack benchmarks, arguing that LLMs often succeed by exploiting literal string matches rather than demonstrating true long-context reasoning. Their findings show strong performance of LLMs on short contexts (under 1K tokens), but sharp degradation with increasing length.

While long-context capabilities are a major focus of recent research, most benchmarks center around retrieval, QA, or summarization tasks. Little work examines how long-context limitations affect LLMs in their role as evaluators, despite this use case becoming increasingly common. This raises a critical question: How reliable can LLM evaluations be if the model cannot robustly process the full input it is supposed to judge?

2.6 Enhancing LLMs with Memory

LLMs often struggle with long-context processing, including forgetting earlier inputs, failing to recall rare facts, and experiencing temporal degradation of knowledge (Mallen et al., 2023; Wang et al., 2025; Modarressi et al., 2025b). These limitations arise from their parametric memory and the computational constraints of their architecture (Mallen et al., 2023). To address these issues, several memory enhancement techniques have been proposed (Wang et al., 2025, 2023; Wang and Li, 2024).

Many approaches include scaling context length, which is limited by computational constraints, and parametric memory, where factual knowledge is stored in model weights but suffers from limited coverage and temporal decay (Mallen et al., 2023; Liu et al., 2024a). In contrast, retrieval-augmented generation (RAG) (Lewis et al., 2021) leverages non-parametric memory from external sources, with techniques like adaptive retrieval improving efficiency and factual precision.

Alternative methods use explicit read-write memory systems, like MEMLLM by Modarressi et al. (2025b), that store and retrieve structured knowledge. Similarly, MemoryBank (Zhong et al., 2024) introduces a general, scalable memory system that enables LLMs to write, retrieve, and update long-term memory across sessions, significantly improving multi-turn consistency and factual grounding.

Self-controlled memory (SCM) frameworks (Wang et al., 2025) manage both short- and long-term memory streams, enabling dynamic reasoning over ultra-long inputs without retraining the LLM. Architectural extensions, such as Memorizing Transformers by Wu et al. (2022) and LONGMEM by Wang et al. (2023), integrate external memory modules for scalable and persistent in-context learning. Additionally, fill-in-the-middle (FIM) training (Bavarian et al., 2022) equips models with bidirectional infilling capabilities to better utilize full-text structure.

In our setup, we adopt an in-context memory approach: A structured, JSON-formatted memory representation is provided to the LLM at inference time to guide final evaluation, offering a lightweight and interpretable mechanism for memory integration.

3 Part I - Building CoFluEval-LC

We divide our contributions into two main parts: (1) building a diagnostic dataset to evaluate LLMs evaluation performance on long documents, and (2) introducing a memory-based pipeline to address the long-context limitations in LLM-as-a-Judge evaluations. This chapter focuses on the first part.

We introduce our diagnostic dataset, **CoFluEval-LC** (**Co**herence and **Flu**ency **Eval**uation in **L**ong **C**ontext), designed to assess LLM's abilities to evaluate text quality in long-context scenarios. The dataset consists of two sub-datasets:

- The **Gold Dataset** serves as the baseline and reference for evaluating manipulated documents. It consists of 65 carefully selected and preprocessed documents from Project Gutenberg (Project Gutenberg, 1971). The task here is to evaluate the quality of each document in terms of coherence and fluency. Details on data source selection and the gold corpus creation process are provided in Section 3.1.
- The **Manipulation Dataset**, in contrast, is significantly larger, as it includes perturbed versions of the gold documents for six different manipulation types. In this dataset, gold documents are manipulated under controlled conditions, targeting different aspects of text quality, such as grammar or logical consistency, depending on the targeted evaluation metric. Thereby, each manipulation can be seen as a task to evaluate on. Further details on the construction of this dataset begin in Section 3.2.

We present the final statistics on CoFluEval-LC in Section 3.6.

To further isolate the effect of context length on evaluation performance, we additionally constructed a **S**hort-**C**ontext variant of the dataset called **CoFluEval-SC**, in which each document and its manipulations are truncated to approximately 2,000 tokens (also referred to as the `2K` setup). While not part of the core benchmark, this companion set serves as a baseline for evaluating LLM behaviour in reduced-context settings.

3.1 Data Collection

To evaluate the ability of large language models to understand and reason over long-form texts, we require a dataset of coherent, human-authored documents with appropriate length and complexity. This section details the criteria used to define suitable documents, the process of selecting a relevant data source, and the steps taken to preprocess the texts for downstream evaluation.

3.1.1 Selection Criteria

As a starting point, we require a reliable data source of human-written texts that can serve as a gold standard for evaluation. This is especially important, as these original documents will later be compared to manipulated versions to assess the sensitivity of LLMs to changes in quality.

Generally, the models evaluated in our experiments support context windows of up to 128K tokens (cf. Section 4.1.2). However, we hypothesize that effective comprehension declines as context length increases as shown in previous studies. This may occur because models lose track of earlier information, confuse details, or exhibit uneven attention across the entire input. To account for this, we selected documents between 8,000 and 16,000 tokens in length as recent studies (cf. Section 2.5) have shown that model performance already begins to degrade at around 3,000 to 4,000 tokens document length (Levy et al., 2024; Modarressi et al., 2025a). Thereby, to approximate document length, we use whitespace tokenization as it is model-agnostic and roughly corresponds to the number of words in a text.

Consequently, we define the following **data source requirements**: The data source should contain documents with rich discourse structure, non-trivial temporal and causal dependencies, and engaging narratives, including both narration and dialogue. Such complexity challenges the model's understanding and memory across extended contexts. To ensure consistency and appropriateness for narrative evaluation, we exclude poetry, song lyrics, and dramatic scripts from the dataset. Documents of these genres are filtered out during preprocessing (cf. Section 3.1.3).

3.1.2 Data Sources

During our data source selection process, we considered several existing datasets. **∞-Bench** (Zhang et al., 2024) is a recently introduced benchmark for evaluating the long-context capabilities of LLMs. It provides extremely long documents exceeding 100K tokens in both English and Chinese, which are significantly longer than what our experimental setup targets. The **L-Eval** suite (An et al., 2024) offers a more standardized evaluation framework for long-context LLMs by consolidating four existing datasets. However, only one of its tasks fits our criteria, and it includes only three documents within our desired length range.

Other datasets we reviewed already contain documents from Project Gutenberg. **NarrativeQA** (Kočiský et al., 2018) is a reading comprehension dataset comprising books from this source and movie scripts scraped from the web. Similarly, **BookSum** (Kryscinski et al., 2021), a summarization dataset, focuses on long-form narrative texts that also include content from Project Gutenberg. Based on this analysis, we decided to work directly with documents from the original corpus.

Project Gutenberg is the world's oldest digital library[1], offering over 75,000 free eBooks. It was founded by Michael Hart in 1971. For the purpose of this thesis, we used the version of the dataset available on Hugging Face[2] (Manu, 2022). The English subset of this dataset contains 38,026 books. Based on our length criteria, we extracted 471 documents containing between 8,000 and 16,000 tokens. Thereby, the shortest document in this set has 8,036 tokens, the longest contains 15,969 tokens, while the average length is approximately 11,820 tokens. This subset of 471 documents is a sufficiently large basis for our dataset, even if some texts are later excluded during the preprocessing phase.

3.1.3 Data Cleaning and Splitting

To prepare the extracted Project Gutenberg documents for manipulation and later LLM-evaluation, we implemented a multi-step preprocessing pipeline focused on cleaning and structuring raw texts.

Initially, documents were semi-automatically filtered to exclude non-narrative works such as poems, songs, and dramas by searching for indicative keywords in their beginning metadata. This process also included extensive manual verification to remove any remaining unwanted document types not caught automatically. Next, the remaining texts underwent automated cleaning: Bracketed content (e.g., editorial notes) and lines containing asterisks were removed to eliminate extraneous material. Underscores were stripped to improve readability. To better segment the text, line

[1] See https://www.gutenberg.org

[2] See manu/project_gutenberg on Hugging Face

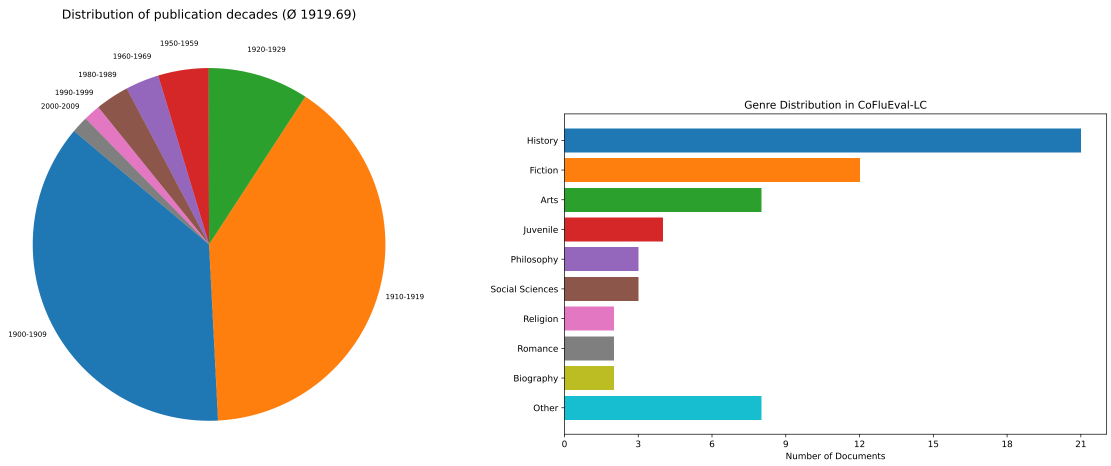
breaks were inserted after sentence-ending punctuation, while carefully preserving abbreviations to avoid false splits. Additionally, chapter headings were identified and standardized by detecting lines starting with “CHAPTER” or all-caps titles, facilitating downstream structural analysis.

The resulting 151 processed texts were saved as clean, well-formatted files, and accompanying metadata was generated to capture source information and enable traceability. Elaborate manual review was incorporated to ensure quality and consistency. This pipeline ensured the resulting documents were both coherent and suitably formatted for long-context evaluation by large language models.

3.1.4 Data Analysis

For our experiments, we define the *gold dataset* as the subset of documents published after 1900. Although we initially intended to retain all 151 documents, we observed significant stylistic and linguistic differences in those written before 1900. These documents often exhibited outdated language or structure, which risked distorting evaluation results. For example, a poorly structured text might receive a low score from the model even without manipulation, making it difficult to observe further degradation in quality.

After filtering, the resulting gold dataset consists of **65 documents**, with an average length of approximately 11,756 tokens per document (based on whitespace tokenization). It contains rich narrative content, including descriptions of artworks, fictional stories, and interpretive prose (cf. Figure 3.1b). Figure 3.1a shows the distribution of publication decades, including publication years ranging from 1900 to 2005, across our gold dataset and highlights the dominance of early 20th-century texts in the final gold dataset of CoFluEval-LC.



(a) Publication decades

(b) Genres

Figure 3.1: Overview of the 65 documents included in the gold dataset: (a) shows the distribution of publication decades, and (b) visualizes the most frequent genres, with infrequent ones grouped as “Other”.

These documents serve as the foundation for our later manipulation experiments. A detailed overview of their structural properties, including token, sentence, and paragraph statistics, is provided in Table 3.1.

General Information	
Total Number of Gold Documents	65
Data Source	Project Gutenberg
Temporal Coverage	1900-2005
Language	English
Main Genres	history, fiction, arts
Average Document Statistics	
Average Number of Characters	67457.98
Average Number of Tokens	
→ Whitespace-Tokenizer	11756.32
→ spaCy-Tokenizer	14981.35
→ Llama-3.3-70B-Instruct Tokenizer	16013.43
Average Number of Sentences*	178.46
Average Number of Paragraphs**	517.25

Table 3.1: General statistics of the gold document dataset
* determined by spaCy **determined by seperator: `\n\n`

3.2 Prerequisites for CoFluEval-LC

In the following, we introduce the two core text quality metrics chosen for the analysis in this thesis. In addition, we define the term *manipulation* and explain its role in the construction of the manipulated dataset as a part of CoFluEval-LC. We further introduce basic concepts for the application of manipulations based on structural and granular dimensions in Section 3.2.3.

3.2.1 Text Quality Metrics

To fit the scope of this thesis, we decided to focus on two core metrics for our evaluation approach. As discussed in Section 2.2, text quality can be assessed along various dimensions, with definitions and interpretations differing widely across the literature. Our goal is to include one metric that reflects deeper textual qualities, such as logical consistency, narrative progression, and structural coherence. The second metric is chosen to capture surface-level properties, such as grammatical correctness and syntactic well-formedness, which can be evaluated without deep understanding of the content. In line with this reasoning, we concentrate our analysis on two fundamental dimensions of text quality: *coherence* (C) and *fluency* (F).

Coherence is broadly understood as the quality that makes a text logically and thematically unified, allowing it to be perceived as a well-organized and meaningful whole rather than just a collection of related sentences. Several studies (e.g., Gao et al., 2023; Fabbri et al., 2021; Kryscinski et al., 2021) emphasize coherence as the collective quality of sentences working together to form a natural and consistent flow of ideas, rather than a disjointed heap of information. This involves a clear progression from sentence to sentence, ensuring the text builds logically and smoothly throughout. Other definitions focus on semantic relevance and appropriateness of content within the given context (Ke et al., 2022; Howcroft et al., 2020; Celikyilmaz et al., 2020). This includes the maintenance of consistent characters, themes, and narrative elements, which is especially important in longer, complex texts. Resources like (StudySmarter, 2024) elaborate on coherence as the logical and consistent connection between parts of a text, highlighting factors such as referential clarity, logical transitions, thematic recurrence, and the use of foreshadowing to guide the reader. Abrupt shifts, inconsistencies, or off-topic content are commonly noted as coherence disruptors.

Our understanding aligns with these perspectives, viewing **coherence** as how well the text holds together logically and thematically, featuring a clear and understandable line of reasoning with natural idea progression and sustained relevance throughout the text.

Type	Coherence Criterion
High-level	**Coherence**: It measures the quality of all sentences collectively: Do they make sense as a whole, with the context organized and connected logically?
Detailed	**Coherence**: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
Evaluation Aspects	**Logical Sequencing**: Are the ideas presented in a logical order? **Non-repetitiveness**: Is the text free from unnecessary repetition, or does it repeat information without adding new insights? **Smooth Connections**: Are the transitions between sections or ideas smooth, or are they abrupt? **Ambiguity Avoidance**: Does the text avoid ambiguity, or are there parts that are unclear? **Structural Consistency**: Is the narrative or argument structured in a consistent and clear manner?
Scoring Scale	**Score 5**: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding. **Score 4**: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text. **Score 3**: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience. **Score 2**: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort. **Score 1**: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

Table 3.2: **Coherence** (Coh./C) definitions and evaluation critieria. We define different definition granularities as well as evaluation aspects and a coherence scoring scale.

In the literature, *fluency* is commonly associated with surface-level aspects of text quality, yet its definitions span a spectrum of granularity and focus. Many works describe fluency as the quality of individual sentences, emphasizing grammatical correctness, proper spelling, and stylistic well-formedness (Gao et al., 2023; Fabbri et al., 2021; Jain et al., 2023; Kryscinski et al., 2021). Others take a broader view, considering how well sentences connect to one another and how the overall document is structured for ease of reading and information delivery (Lai and Tetreault, 2018). Several sources equate fluency with readability, linking it to factors such as natural phrasing, appropriate vocabulary, and absence of awkward or error-prone constructions (Celikyilmaz et al., 2020; Howcroft et al., 2020). Some definitions extend to formatting and presentation issues, such as capitalization and sentence clarity (Dang, 2006). While most definitions converge on fluency as the degree to which a text is grammatically correct and easy to read, they diverge in whether fluency also encompasses discourse-level organisation or stylistic cohesion.

In line with these perspectives, we define **fluency** in this work as the extent to which a text is well-formed at the grammatical and stylistic level, exhibiting correct syntax, clean typography, and natural phrasing that facilitates smooth and effortless reading.

Type	Fluency Criterion
High-level	**Fluency**: It measures the quality of individual sentences: Are they grammatically correct, non-repetitive, and in accord with common English usage, with clear meanings?
Detailed	**Fluency**: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
Evaluation Aspects	**Syntax and Grammar**: How well-constructed are the sentences in terms of grammatical correctness? **Spelling and Punctuation**: Are there any noticeable errors in spelling or punctuation that disrupt the flow? **Word Choice**: Is the vocabulary appropriate for the context, and does it contribute to a smooth reading experience? **Phrasing**: Are the phrases well-structured, or are they awkwardly worded? **Flow**: Does the text flow smoothly from one sentence to the next, or are there abrupt transitions?
Scoring Scale	**Score 5**: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding. **Score 4**: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability. **Score 3**: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways. **Score 2**: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort. **Score 1**: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

Table 3.3: **Fluency** (Flu./F) definitions and evaluation critieria. We define different definition granularities as well as evaluation aspects and a fluency scoring scale.

Inspired by Hu et al. (2024), we define the definitions and main quality criteria for each metric in Table 3.2 and Table 3.3.

3.2.2 Definition and Purpose of Manipulations

Similar to Hu et al. (2024), we apply a set of perturbation attacks to the gold-standard documents in our dataset. We refer to them as *manipulations* in our work.

A manipulation m is defined as a deliberate and controlled alteration of a specific feature or property of a text document, performed with the intention of degrading a targeted quality dimension. Thereby, it preserves the overall structure and readability of the document wherever possible.[3]

[3] Nonetheless, we acknowledge that certain manipulations may have incidental effects on the non-targeted dimension.

Formally, let $D_{\text{gold}} = \{d_{\text{gold}}^{(1)}, d_{\text{gold}}^{(2)}, \ldots, d_{\text{gold}}^{(a)}\}$ denote the set of a unaltered documents as we have obtained from the procedures in Section 3.1, where $d_{\text{gold}}^{(i)} \in D_{\text{gold}}$ is a gold document. A manipulation function $m_{e,t_{\text{mani}}}$ is parameterized by a target evaluation metric $e \in \{\mathrm{C}, \mathrm{F}\}$, where **C** stands for **C**oherence and **F** for **F**luency, and a manipulation type t_{mani} as introduced in Section 3.3. Given a gold document $d_{\text{gold}}^{(i)}$, the corresponding manipulated document is defined as:

$$d_{e,t_{\text{mani}}}^{(i)} = m_{e,t_{\text{mani}}}(d_{\text{gold}}^{(i)}) \tag{3.1}$$

Each manipulated document $d_{e,t_{\text{mani}}}^{(i)}$ is thus a functionally dependent transformation of a gold document, selectively altering features relevant to e while minimizing unintended side effects on other aspects of text quality.

The purpose of these manipulations is to generate controlled, contrastive document pairs for the targeted evaluation of LLM-based text quality evaluators. For a given $d_{\text{gold}}^{(i)}$, multiple manipulated variants may be created, each differing in the targeted quality dimension e and/or manipulation type t_{mani}. This many-to-one relationship facilitates pairwise and groupwise comparisons across different manipulation conditions while keeping the source content constant.

3.2.3 Scope of Manipulations and Granularity Levels

All manipulations presented in Section 3.3 are designed to operate in different scopes and levels of granularity, allowing for both fine-grained and coarse-grained modifications.

Scope of Manipulations

We categorize our manipulations not only by metric and type, but also by their applicable scope. Therefore, we differentiate between two main dimensions: *structure* and *density*.

For the *structural* dimension, a manipulation can be applied to a given document at one of four hierarchical scopes: paragraph-, section-, chapter-, or document-level. These scopes correspond to progressively larger structural units, determining how much of the text is affected by a single manipulation operation.

a) **paragraph-level**: In this work, we define a *paragraph* as a block of text delimited by two consecutive newline characters (`\n\n`). To qualify for textual manipulation, a paragraph must exceed a specified minimum length, which is configurable via a parameter during the manipulation process. We also applied constraints to exclude chapter headings and titles from the paragraph content.

b) **section-level**: A *section* is defined as a sequence of m consecutive paragraphs whose combined length reaches at least 3000 characters by default. We opted not to define a section solely based on a fixed number of paragraphs, as paragraph lengths can vary significantly across documents, resulting in inconsistent section sizes. The minimum required character count for a section is configurable via a parameter when the manipulation is applied. We implement constraints to exclude chapter headings and titles from the section content.

c) **chapter-level**: Assuming a document composed of distinct chapters, as defined by the author's original structure, manipulations are applied at the chapter level, meaning that the *entire chapter* is affected. This scope is only applicable when the chapter structure can be reliably extracted from the document during preprocessing.

d) **document-level**: The largest possible scope for manipulation is the *entire document*. Unlike the previously discussed scopes - paragraph, section, and chapter - document-level manipulation does not rely on intermediate structural boundaries or size constraints. Instead, it treats the full text as a single unit.

Depending on the document to be manipulated and the manipulation itself, not all hierarchical scopes are applicable. For the final dataset presented in Section 3.6, we systematically evaluated various hierarchical scopes to establish the optimal level of application for each manipulation type (cf. Section 3.4).

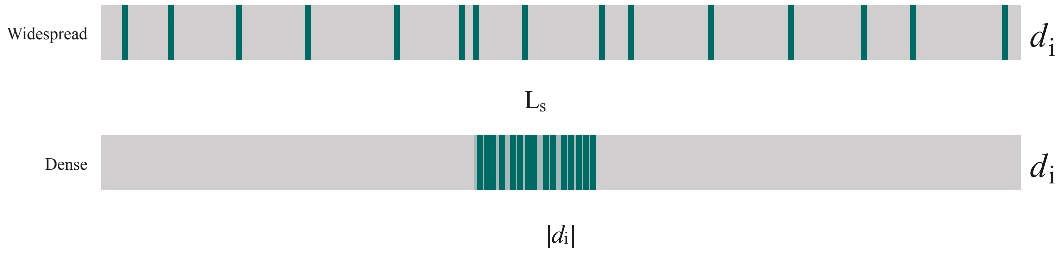


Figure 3.2: Widespread vs. dense manipulation at document-level. Grey bars represent a manipulated document d_i (simple notation for readability), dark green segments illustrate where manipulations are applied within the document. L_s is the span in which localized manipulations are applied (dense setting).

For the *density* dimension, we distinguish between two settings: (1) *Widespread* applied manipulations are randomly distributed across the entire document, whereas (2) *dense* applied manipulations are localized randomly distributed within a specific region. The total amount of manipulated content remains constant in both cases. This distinction was introduced to investigate whether the model is more sensitive to manipulations when they are dispersed throughout the document, as opposed to being concentrated in a single, heavily affected region. In Section 3.4, we examine whether the distribution pattern influences the model's perception or handling of degraded textual quality. Density is applicable for document-level fluency manipulations only.

To demonstrate how manipulation density influences the text, consider the following example (cf. Figure 3.2): When introducing a (typo) rate of $r = 2$ (meaning 2%) into a given document $d_{\text{gold}}^{(i)}$, the total number of characters to be affected by typos, denoted as n_o (number of operations to be performed), is computed as:

$$n_o = \text{round}\left(r \cdot \frac{|d_{\text{gold}}^{(i)}|}{100}\right) \tag{3.2}$$

where $|d_{\text{gold}}^{(i)}|$ denotes the total number of tokens in the gold document (determined by whitespace-tokenization). This manipulation can either be *randomly distributed* across the entire document or *localized randomly distributed* within a specific segment. In the localized case, the span of the manipulated region, denoted L_s, is determined by:

$$L_s = n_o \cdot n_{\text{lpw}} \cdot \beta \tag{3.3}$$

Here, $n_{\text{lpw}} = 5$ is the average number of letters per word in English (Norvig, 2012), and $\beta = 2.5$ is an empirically determined scaling factor introduced to mitigates the risk of producing entirely nonsensical character sequences by preserving enough surrounding content to allow for partial inference of the original word forms.

Granularity in Document-Level Manipulations

Each document-level manipulation can be further specified by the level of application, distinguishing between token-level and character-level operations.

a) **token-level**: These operate on *word units*, involving substitutions, deletions, or rearrangements. For example, all entity-based manipulations as well as removing or swapping words fall into this category. Token-level manipulations are mostly relevant for semantic or syntactic perturbations and are applied document-wide in dense or widespread settings, depending on the specific manipulation type.

b) **character-level**: These affect individual *characters*, such as inserting typos or punctuation mistakes. Such changes are typically applied at document-level in dense or widespread settings.

Please refer to Figure A.13 in the appendix for a comprehensive overview of all the concepts introduced in this section.

3.3 Text Manipulation Techniques

To facilitate the generation of diverse textual variations for analysis and experimentation, we implemented a range of rule-based text manipulation strategies, as detailed in the following sections. This serves our overarching goal of evaluating the ability of large language models to assess text quality in *long-context scenarios*. We hypothesize that LLMs, while effective on shorter documents, may struggle to fully capture meaning and identify quality-related issues when confronted with significantly longer texts. By introducing controlled disruptions, we aim to test whether LLMs can still detect such manipulations and reflect them appropriately in their evaluations.

Our approach focuses on manipulations grounded in structural features of the preprocessed documents. These manipulations leverage established NLP techniques such as named entity recognition, regular expressions, and tokenization. We deliberately refrain from using content-aware manipulation techniques that would require deeper semantic understanding, as these would necessitate the use of LLMs themselves or involve extensive manual validation, which is outside the scope of this work.

A visual overview of the manipulation types introduced in this section is provided in Figure A.13. We further provide Table A.3 as a structured summary of all implemented manipulations, including their IDs, types, and short descriptions.

Preliminaries However, before we deep-dive into coherence and fluency manipulations, we first give some fundamental definitions that we will continually refer to in the following explanations:

Resulting from Section 3.2.3, we give three different ways of defining the gold document $d^{(i)}_{\text{gold}}$, dependent on the applied scope and level of granularity of a manipulation.

1. In terms of paragraph-level manipulations, let a gold document $d^{(i)}_{\text{gold}}$ (not manipulated so far) be a sequence of paragraphs:

$$d^{(i)}_{\text{gold}} = [p_1, p_2, \dots, p_n] \tag{3.4}$$

where each p_j denotes a paragraph, defined as a block of text delimited by two consecutive newline characters. A gold document $d^{(i)}_{\text{gold}}$ consists of n paragraphs in total.

2. If a manipulation operates on document-level, focusing on tokens, let the gold document $d_{\text{gold}}^{(i)}$ be a sequence of tokens:

$$d_{\text{gold}}^{(i)} = [tok_1, tok_2, \ldots, tok_{|d_{\text{gold}}^{(i)}|}] \tag{3.5}$$

   where each tok_j denotes a token (as referred to as *word*) obtained by applying a whitespace-tokenizing function to the gold document.

3. Consequently, for document-level manipulations tackling characters, we define $d_{\text{gold}}^{(i)}$ as a sequence of characters:

$$d_{\text{gold}}^{(i)} = [char_1, char_2, \ldots, char_c], \quad char_j \in \Sigma \tag{3.6}$$

   where $char_j$ is a character in the alphabet Σ and $d_{\text{gold}}^{(i)}$ consists of c characters in total.

Furthermore, we define a sequence of named entity mentions extracted from a sequence of tokens (cf. Equation 3.5) as

$$\mathcal{E} = [ent_1, \ldots, ent_x] \tag{3.7}$$

where each ent_j is a single or multi-token entity span mention labeled with a supported entity type t_{ent} and x is the number of all entity mentions detected by spaCy[4].

3.3.1 Coherence Manipulations

Following the definitions of coherence introduced in Section 3.2.1, we introduce a set of manipulations whose aim is to affect the overall quality of all sentences of a text collectively. Therefore, we define three subcategories of coherence manipulations: *Logical Flow Disruptions, Plot Inconsistencies, Anaphora Resolution.* All of them prioritize different aspects of coherence, each maintaining various manipulation types. Here, we highlight only the most relevant implemented manipulations; a comprehensive overview of all others is provided in the appendix (cf. Section A.3).

Logical Flow Disruptions

Logical Flow Disruptions are manipulations that intentionally disturb the argumentative or inferential progression of a text. These manipulations aim to degrade the coherence of the document at a discourse level, specifically, by weakening or breaking the logical relationships between ideas, sentences, or paragraphs.

As detailed in Section 3.2.3, we introduce different structural operation scopes in which manipulations can be applied. For simplicity, we focus on explaining the inner workings of all logical flow disruptions at the paragraph level only. All here presented manipulations can be applied either with a count-based or a ratio-based number of operations.

Swap Content [11] As we aim to evaluate the sensitivity of LLMs to structural disruptions, especially in long-form text, we implement the *content swap operation*. This manipulation tests whether an LLM tasked with evaluating the document can detect coherence breaks caused by these rearrangements, particularly in long-context scenarios where logical flow across distant sections is crucial.

[4] See spaCy Documentation: https://spacy.io/usage/linguistic-features#named-entities

Specifically, we randomly perform n_o swaps, where each swap exchanges the position of two distinct paragraphs within a gold document $d^{(i)}_{\text{gold}}$. To preserve the integrity of the manipulation, no paragraph is involved in more than one swap, ensuring that all swapped pairs are unique and non-overlapping. This manipulation is only applied if a sufficient number of valid paragraphs are available to construct the desired number of unique swap pairs; otherwise, the operation is aborted. We give a high-level example for $n_o = 1$ swaps in Figure A.4.

Given $d^{(i)}_{\text{gold}}$ is a set of paragraphs as defined in definition 3.4, we identify a subset $\mathcal{I} \subset \{1, \ldots, n\}$ of paragraphs that exceed a configurable minimum length (e.g., 50 characters). We then randomly sample two distinct, non-overlapping indices $j, k \in \mathcal{I}$ and swap their positions using uniform sampling without replacement:

$$m_{C,\,\text{swap}}(d^{(i)}_{\text{gold}}) = [p_1, \ldots, p_{j-1}, p_k, p_{j+1}, \ldots, p_{k-1}, p_j, p_{k+1}, \ldots, p_n] \tag{3.8}$$

This function is applied n_o times to create the final manipulated document $d^{(i)}_{C,\text{swap}}$.

The updated paragraph sequence is then reassembled into a single text body, maintaining original formatting via preserved delimiters.

Remove Content [12] Furthermore, to introduce abrupt logical jumps and reduce textual coherence, we implement the *content removal operation*. This manipulation tests whether an LLM can detect the absence of meaningful content when portions of a document are omitted. Interestingly, we observe that detecting these removals can also be challenging for human readers, particularly when the removed content is subtly embedded within the broader narrative structure.

We illustrate a high-level example for $n_o = 1$ removal in Figure A.5.

Given $d^{(i)}_{\text{gold}}$ is a set of paragraphs as defined in Definition 3.4, we identify a subset $\mathcal{I} \subset 1, \ldots, n$ such that each $p_j \in \mathcal{I}$ satisfies a minimum paragraph length constraint (e.g., 50 characters). We then uniformly sample n_o distinct indices $j_1, j_2, \ldots, j_{n_o} \subset \mathcal{I}$ without replacement, provided that $|\mathcal{I}| \geq n_o$. If not enough valid candidates exist, the operation is aborted.

These indices define the paragraphs to be removed. The manipulated document $d^{(i)}_{C,\text{remove}}$ is obtained by deleting the corresponding entries from the sequence:

$$d^{(i)}_{C,\text{remove}} = m_{C,\,\text{remove}}(d^{(i)}_{\text{gold}}) = [p_k \mid k \in 1, \ldots, n \setminus j_1, \ldots, j_{n_o}] \tag{3.9}$$

The updated paragraph sequence is then reassembled into a single text body, preserving original formatting via newline delimiters. Note that this operation results in a token reduction in the manipulated document.

Exchange Content [17] In addition to the insert content manipulation (cf. Section A.3), we implement the *content exchange operation*. In contrast to the pure insert or remove manipulations, this operation replaces existing content segments with foreign content drawn from unrelated documents. This setup approximately maintains the overall document size, allowing for controlled comparisons across original and manipulated versions. Figure 3.3 shows an example of two paragraph-level exchanges ($n_o = 2$).

Concretely, we select n_o valid paragraphs from a gold document $d^{(i)}_{\text{gold}}$ and replace them with paragraphs sourced from external documents. Each inserted paragraph matches the unit type (e.g., paragraph, section, or chapter) and satisfies a configurable minimum length. The segments to be exchanged are selected randomly. To avoid the manipulated document increasingly resembling a single donor source, especially at higher values of n_o, we deliberately use different donor documents for each exchange (n_o in total).

Let $D_{\text{insert}} = \{d^{(1)}_{\text{insert}}, \ldots, d^{(n_o)}_{\text{insert}}\}$ be a set of n_o external donor documents, each providing one paragraph for replacement. Each donor document $d^{(s)}_{\text{insert}}$ is parsed as a list of paragraphs, e.g., $d^{(s)}_{\text{insert}} = [q^{(s)}_1, q^{(s)}_2, \ldots, q^{(s)}_m]$.

We first identify a subset $\mathcal{I} \subset \{1, \ldots, n\}$ of valid paragraph indices in $d^{(i)}_{\text{gold}}$ that satisfy a minimum length threshold. Then, for each exchange operation eo, we:

- sample a paragraph index $z_{eo} \in \mathcal{I}$ from the gold document, and
- sample a paragraph $q^{(s_{eo})}_{j_{eo}}$ from a selected donor document $d^{(s_{eo})}_{\text{insert}}$

Each selected gold paragraph $p_{z_{eo}}$ is replaced by $q^{(s_{eo})}_{j_{eo}}$, yielding the manipulated document

$$\begin{aligned} d^{(i)}_{C,\,\text{exchange}} = m_{C,\,\text{exchange}}(d^{(i)}_{\text{gold}}) = [\,& p_1, \ldots, p_{z_1-1},\ q^{(s_1)}_{j_1},\ p_{z_1+1}, \ldots, \\ & p_{z_2-1},\ q^{(s_2)}_{j_2},\ p_{z_2+1}, \ldots, \\ & \cdots,\ p_{z_{n_o}-1},\ q^{(s_{n_o})}_{j_{n_o}},\ p_{z_{n_o}+1}, \ldots, p_n\,] \end{aligned} \tag{3.10}$$

where each gold paragraph $p_{z_{eo}}$ is replaced by donor paragraph $q^{(s_{eo})}_{j_{eo}}$ for every exchange operation index $eo \in \{1, \ldots, n_o\}$.

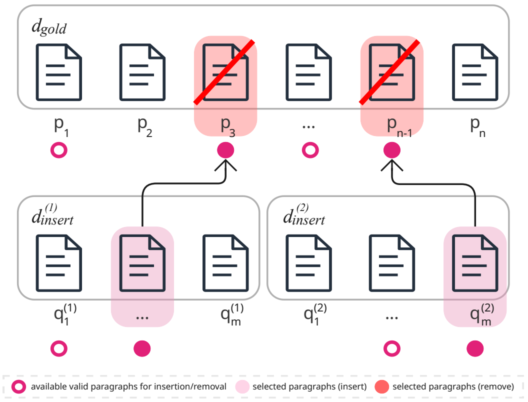
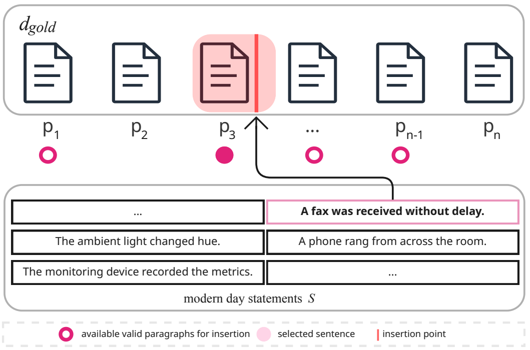


Figure 3.3: Illustration of the paragraph-level exchange manipulation for $n_o = 2$



...	**A fax was received without delay.**
The ambient light changed hue.	A phone rang from across the room.
The monitoring device recorded the metrics.	...

Figure 3.4: Illustration of injecting anachronistic 21st-century content into narratives for $n_o = 1$

Plot Inconsistencies

Plot Inconsistencies include all semantic-level coherence disruptions. They affect the internal consistency of the narrative or conceptual world described by a document. This type of manipulation introduces contradictions or logical impossibilities in the text, disrupting a reader's ability to construct a coherent mental model of the story or argument.

Temporal Inconsistencies [24] We apply the *temporal inconsistency injection* operation, to subtly violate historical plausibility in long-form narratives. This manipulation appends modern-day (21st-century) references, such as mentions of smartphones, social media, or digital platforms, to otherwise temporally consistent paragraphs drawn from earlier periods (cf. Section 3.1.4). These injected sentences are locally grammatical and stylistically neutral, but introduce anachronisms that break historical coherence.

As this manipulation is applied at paragraph level only, we follow Definition 3.4. Furthermore, we identify valid paragraph indices $\mathcal{I} \subseteq \{1, \ldots, n\}$, where each paragraph p_j meets a minimum length requirement. From this subset, we randomly select n_o indices $\{j_1, \ldots, j_{n_o}\}$. To each corresponding paragraph p_{j_k}, we append a sentence $\text{sent}_k \in \mathcal{S}$, sampled from a curated list of modern-day statements:

$$d^{(i)}_{C,\,\text{temp-incon}} = m_{C,\,\text{temp-incon}}(d^{(i)}_{\text{gold}}) = [p_1, \ldots, p_{j_1} + sent_1, \ldots, p_{j_{n_o}} + sent_{n_o}, \ldots, p_n] \quad (3.11)$$

The resulting text retains its original formatting and paragraph boundaries. However, the inserted statements introduce subtle contradictions with the document's temporal setting, mimicking real-world anachronisms (cf. Figure 3.4).

Swap Entities [21] To introduce subtle yet consequential inconsistencies in narrative continuity, we implement the *entity swapping operation*. This manipulation targets named entities (e.g., characters) and permutes a subset of them such that their textual labels are reassigned across different positions in the document. While each entity remains grammatically valid in its new position, the global coherence, particularly of character roles or plot causality, may break in a way that challenges both LLMs and human readers to detect.

Formally, let $d^{(i)}_{\text{gold}}$, defined as in Definition 3.5, denote the token sequence of the gold document, and let $\mathcal{E}$, defined as in Definition 3.7, be the sequence of single-token entity mentions; in our experiments, we focus on $t_{\text{ent}} = \texttt{PERSON}$[5].

To ensure meaningful swaps, we define the set of unique entity surface forms as

$$\mathcal{U} = \{ent \in \mathcal{E} \mid ent \text{ is } \textit{unique}\},$$

which contains distinct entity mentions by their text labels, preventing swaps between identical surface forms and thus avoiding degenerate substitutions (e.g., "Alice" cannot be swapped with another mention of "Alice").

We randomly sample a subset $\mathcal{E}_{\text{sub}} \subseteq \mathcal{U}$ of size n_o, where n_o is computed as a random integer percentage ratio $r \in [start, end]$ of the total number of unique entities $|\mathcal{U}|$. Here and in the following, $start$ and end are user-defined lower and upper bounds for the integer percentage of entities to swap (e.g., $start = 10$, $end = 30$). Formally,

$$n_o = \max\left(2,\ \text{round}\left(r \cdot \frac{|\mathcal{U}|}{100}\right)\right). \quad (3.12)$$

A minimum of $n_o = 2$ is enforced to ensure a valid cyclic swap operation.

We then define a cyclic permutation π over $\mathcal{E}_{\text{sub}}$, such that each entity ent_i is replaced with $\pi(ent_i)$, and update the text accordingly:

$$d^{(i)}_{C,\,\text{entity-swap}} = m_{C,\,\text{entity-swap}}(d^{(i)}_{\text{gold}}) = \texttt{replace}(ent_j \mapsto \pi(ent_j)) \quad \forall ent_j \in \mathcal{E}_{\text{sub}}. \quad (3.13)$$

Figure A.8 conceptually depicts a cyclic permutation of $n_o = 3$ entities (`start` = `end` = 60), where the sampled subset is $\mathcal{E}_{\text{sub}} = \{\text{Alice}, \text{Clara}, \text{Bob}\}$.

Exchange Entities [22] Similar to the swap entities manipulation, we apply the *entity exchange operation*. This manipulation replaces a subset of named entities (e.g., person names or locations) with entirely new entities of the same type. Unlike entity swapping, which permutes

[5] cf. Table A.1 in the appendix for further details.

existing names, this method introduces external substitutions, disrupting narrative grounding and potentially introducing plot-level incoherence.

We define $d_{\text{gold}}^{(i)}$ and $\mathcal{E}$ analogously to Definitions 3.4 and 3.7, respectively. The number of replacements n_o is determined based on a sampled integer percentage $r \in [start, end]$ of eligible entities in $\mathcal{E}$, ensuring $n_o \geq 2$ to allow for meaningful substitution.

From a predefined pool of alternative entities $\mathcal{A} = \{a_1, \ldots, a_z\}$, we sample n_o unique substitutes $\mathcal{A}_{\text{sub}} = \{a'_1, \ldots, a'_{n_o}\} \subset \mathcal{A}$, such that each $a'_k \notin d_{\text{gold}}^{(i)}$.

Each selected entity $ent_j \in \mathcal{E}$ is then replaced at a randomly chosen mention span, resulting in:

$$d_{C,\text{ entity-exchange}}^{(i)} = m_{C,\text{ entity-exchange}}(d_{\text{gold}}^{(i)}) = \texttt{replace}(ent_j \mapsto a'_j) \quad \forall j = 1, \ldots, n_o. \tag{3.14}$$

Figure A.10 illustrates this process by replacing $n_o = 3$ (`start` = `end` = 60) known entity mentions $\{\text{Alice}, \text{Clara}, \text{Bob}\} \in \mathcal{E}$ with unrelated alternative names $\mathcal{A}_{\text{sub}} = \{\text{Sebastian}, \text{Lena}, \text{Sarah}\}$.

To avoid degenerate replacements, sampled substitutes are guaranteed not to appear in the original document. Only single-token named entities with capitalized first letters are eligible, consistent with narrative conventions.

Anaphora Resolution

Anaphora Resolution manipulations intentionally weaken the referential clarity of a text by corrupting how entities are tracked across discourse. This includes replacing specific noun phrases with vague pronouns or generic placeholders, resulting in referential ambiguity and disrupting the reader's ability to resolve coreference chains.

As all manipulations in this section operate at the document level (tokens) and target the same entity types $t_{\text{ent}} = \{\texttt{LOC}, \texttt{GPE}, \texttt{ORG}, \texttt{FAC}\}$, we define the original document as in Definition 3.5 and the sequence of entity mentions $\mathcal{E}$ as in Definition 3.7. Unlike the swap entities and exchange entities manipulations in Section 3.3.1, we now also include multi-token entities, not just single-token ones.

Entity-to-Term Replacement [32] The *entity-to-term replacement* manipulation replaces named entities with generic replacement terms such as *thing* in order to reduce entity specificity and introduce ambiguity. The replacement is adjusted for sentence-initial capitalization and plural forms where applicable, ensuring grammatical coherence.

Given the Definitions 3.5 and 3.7, an integer replacement percentage ratio $r \in [start, end]$ is randomly sampled, and the number of replacement operations n_o to perform is computed as:

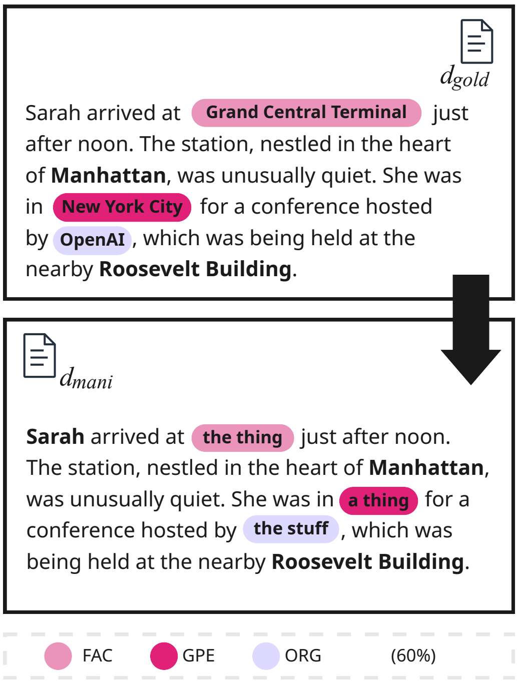
$$n_o = \text{round}\left(r \cdot \frac{|\mathcal{E}|}{100}\right) \tag{3.15}$$

similarly to the swap entities manipulation.

From the extracted entity set $\mathcal{E}$, we again uniformly select a subset $\mathcal{E}_{\text{sub}} \subset \mathcal{E}$ of size n_o. Each entity $ent_j \in \mathcal{E}_{\text{sub}}$ is then substituted with a randomly chosen replacement term $term_j$ from a user-defined list. In the following experiments in this work, we decided to use the terms *thing* and *stuff* for this manipulation. Each exchanged term can be optionally preceded by an appropriate article ("a", "an", or "the") based on the given context in $d_{\text{gold}}^{(i)}$.

Figure 3.5 shows a sample illustration where $n_o = 3$ selected entities (highlighted) are replaced by the generic terms *stuff* and *thing*, demonstrating the loss of entity specificity while preserving grammatical structure.

$$d^{(i)}_{C,\text{ ent-term}} = m_{C,\text{ ent-term}}(d^{(i)}_{\text{gold}}) = \texttt{replace}(ent_j \mapsto term_j) \quad \forall ent_j \in \mathcal{E}_{\text{sub}} \tag{3.16}$$

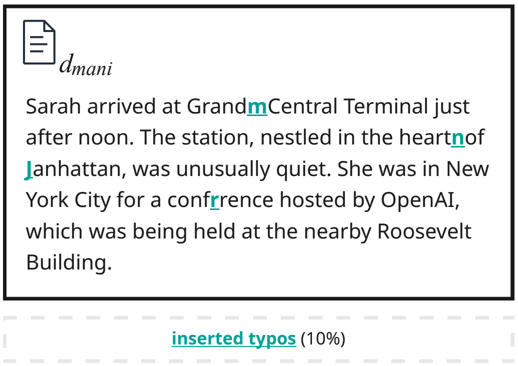

d_{gold}

Sarah arrived at Grand Central Terminal just after noon. The station, nestled in the heart of **Manhattan**, was unusually quiet. She was in New York City for a conference hosted by OpenAI, which was being held at the nearby **Roosevelt Building**.

d_{mani}

Sarah arrived at the thing just after noon. The station, nestled in the heart of **Manhattan**, was unusually quiet. She was in a thing for a conference hosted by the stuff, which was being held at the nearby **Roosevelt Building**.

Figure 3.5: Example of named entity substitution using generic terms for $n_o = 3$


d_{mani}

Sarah arrived at GrandmCentral Terminal just after noon. The station, nestled in the heartnof Janhattan, was unusually quiet. She was in New York City for a confrrence hosted by OpenAI, which was being held at the nearby Roosevelt Building.

Figure 3.6: Illustration of character-level typo injection for $n_o = 4$

3.3.2 Fluency Manipulations

Our implemented *fluency manipulations*, on the other hand, aim to affect the quality of individual sentences (cf. Section 3.2.1). We achieve this by focusing on introducing various grammatical errors to our gold documents, such as inserting *typos* or *shuffeling word order*.

All fluency manipulations are applied on a document level, with some of them affecting individual tokens (cf. Equation 3.5) and others affecting characters within these tokens (cf. Equation 3.6). As for coherence, we only introduce the most important manipulations in this part; further manipulations can be found in the appendix (cf. Section A.3).

Grammatical Errors

The manipulation category of *Grammatical Errors* includes all types of manipulations that degrade the grammatical well-formedness of a text. These manipulations focus on altering syntactic correctness and structural aspects of grammar, while attempting to leave the overall meaning and coherence of the text as intact as possible.

All here described types of manipulation can be applied in widespread or dense settings (cf. Section 3.2.3).

Typos [41] With the *typo insertion* operation, we introduce character-level noise by inserting random typos into the text, simulating common typing errors.

Specifically, a subset of characters is selected from $d_{\text{gold}}^{(i)}$ and each is replaced with a nearby key on a QWERTY keyboard layout (e.g., replacing `t` with `r` or `y`)[6].

This process preserves surface fluency while degrading lexical integrity, making it harder for models to rely on exact token identities.

Let $d_{\text{gold}}^{(i)}$ denote the character sequence of the input document (Definition 3.6). In the ratio-based setting, the number of typo insertion operations to perform is sampled based on integer corruption percentage $r \in [\text{start}, \text{end}]$ as in Definition 3.2.

A set of character indices $\mathcal{I}_{\text{sub}} = \{j_1, \ldots, j_{n_o}\} \subset \{1, \ldots, |d_{\text{gold}}^{(i)}|\}$ is then sampled uniformly. For each index $j_k \in \mathcal{I}_{\text{sub}}$, the character $char_{j_k}$ is substituted with a randomly chosen neighboring key from a fixed QWERTY adjacency map:

$$d_{F,\,\text{typo}}^{(i)} = m_{F,\,\text{typo}}(d_{\text{gold}}^{(i)}) = \texttt{replace}(char_{j_k} \mapsto \text{qwerty\_near}(char_{j_k})) \quad \forall j_k \in \mathcal{I}_{\text{sub}} \tag{3.17}$$

Figure 3.6 illustrates the resulting manipulated document after inserting character-level typos. Given a total of 41 tokens and a typo rate of 10%, this corresponds to $n_o = 4$ character substitutions.

Incorrect Verb Tenses [42] The *Verb Tense Change* manipulation primarily targets the grammatical fluency of a document by altering the tense of verbs and auxiliary verbs. By introducing mismatches in tense usage, the manipulation degrades the natural flow and grammatical correctness of the text. While the disruption is localized at the token level, it can also produce secondary effects on temporal coherence, making the sequence of events less intuitive to follow.

As this manipulation operates at the document level and targets token-level structures (verbs), we use the definition of the gold document as in Definition 3.5. From this token sequence, we extract all verbs and auxiliary verbs using part-of-speech tagging[7]. These constitute the candidate pool for modification.

Given a manipulation intensity parameter $r \in [start, end]$, the number of tense-flipping operations n_o to be applied is computed similarly to other token-level manipulations, such as:

$$n_o = \text{round}\left(r \cdot \frac{|\mathcal{V}|}{100}\right) \tag{3.18}$$

for the ratio-based setting, where $\mathcal{V}$ denotes the set of verb and auxiliary tokens in $d_{\text{gold}}^{(i)}$.

Depending on whether the manipulation is distributed globally or concentrated locally (dense), a set of n_o verbs $\mathcal{V}_{\text{sub}}$ is selected uniformly at random either from the full document or from a localized span L_s determined by a sliding window over adjacent sentences (cf. Equation 3.3).

Each selected verb v_j undergoes a tense inversion: If v_j is tagged as past tense, it is inflected into the present tense (either third person singular or base form, depending on subject agreement); if it is in present tense, it is transformed into the past tense:

$$d_{F,\,\text{verb-tense}}^{(i)} = m_{F,\,\text{verb-tense}}(d_{\text{gold}}^{(i)}) = \texttt{inflect}(v_j) \quad \forall v_j \in \mathcal{V}_{\text{sub}} \tag{3.19}$$

We give an example in Figure 3.7 for $r = 25$ (25% intensity) resulting in $n_o = 2$ verb tense change operations out of eight (auxiliary) verbs (highlighted in bold) in the document.

[6] We initially adapted the idea and code of Matthew Anderson (Jul 5, 2019 at 22:33) from stackoverflow: Issue 56908331.

[7] We use spaCy in combination with the LemmInflect extension. See LemmInflect Documentation: https://lemminflect.readthedocs.io/en/latest/ for more information.

Shuffling Word Order [43] To degrade the fluency of a document, we introduce the *sentence-based word order shuffling* manipulation that randomly exchanges the positions of words within sentences. By disrupting the canonical order of tokens, this manipulation introduces local syntactic noise that can lead to awkward or confusing phrasing.

Although the manipulation operates at the sentence level, affecting sequences of tokens as defined in Definition 3.5, it can also impact local coherence when semantic or grammatical dependencies are broken (cf. Figure A.17).

Given the manipulation intensity parameter $r \in [start, end]$, the number of operations to be performed (equals the number of sentences to be affected), denoted n_o, is computed as:

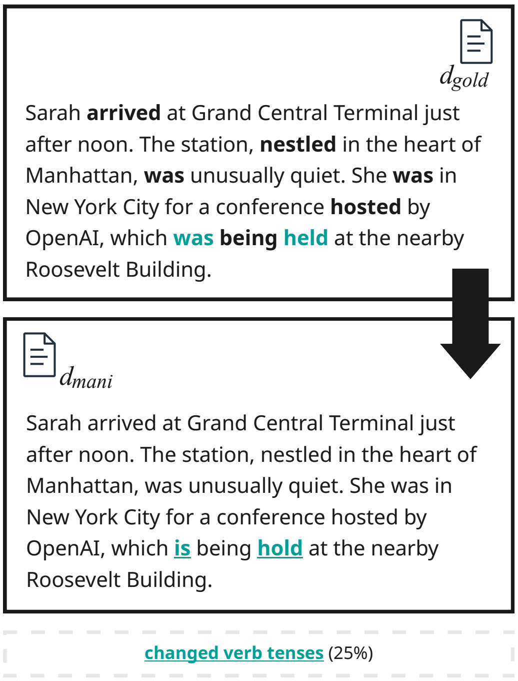
$$n_o = \text{round}\left(r \cdot \frac{|\mathcal{S}|}{100}\right) \tag{3.20}$$

in the ratio-based setting, where $\mathcal{S} = [s_1, s_2, \ldots, s_w]$ denotes the ordered sequence of tokenized sentences in the document.

For each selected sentence $s_j \in S_{\text{sub}}$, either chosen randomly throughout the document or from a localized dense segment as per Definition 3.3, a random pairwise swap of internal tokens is applied, excluding the first and last tokens as well as punctuation to maintain basic sentence integrity:

$$d^{(i)}_{F,\text{ word-order}} = m_{F,\text{ word-order}}(d^{(i)}_{\text{gold}}) = \texttt{swap}(tok_k, tok_l) \quad \forall tok_k, tok_l \in s_j \tag{3.21}$$

Figure 3.8 illustrates this manipulation for $r = 33$ (33% intensity), swapping words in one out of three sentences, which results in altered token order that impacts fluency while potentially affecting coherence locally.

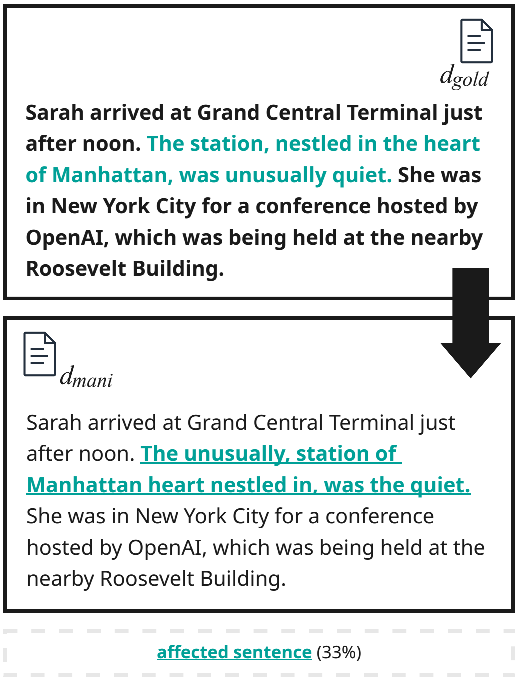


Sarah **arrived** at Grand Central Terminal just after noon. The station, **nestled** in the heart of Manhattan, **was** unusually quiet. She **was** in New York City for a conference **hosted** by OpenAI, which **was being held** at the nearby Roosevelt Building.

Sarah arrived at Grand Central Terminal just after noon. The station, nestled in the heart of Manhattan, was unusually quiet. She was in New York City for a conference hosted by OpenAI, which **is** being **hold** at the nearby Roosevelt Building.

changed verb tenses (25%)

Figure 3.7: Illustration of verb tense manipulation for $n_o = 2$ (25% intensity)



Sarah arrived at Grand Central Terminal just after noon. The station, nestled in the heart of Manhattan, was unusually quiet. She was in New York City for a conference hosted by OpenAI, which was being held at the nearby Roosevelt Building.

Sarah arrived at Grand Central Terminal just after noon. **The unusually, station of Manhattan heart nestled in, was the quiet.** She was in New York City for a conference hosted by OpenAI, which was being held at the nearby Roosevelt Building.

affected sentence (33%)

Figure 3.8: Illustration of the sentence word order swap manipulation for $n_o = 1$

3.4 Effectiveness of Manipulations

Given the extensive number of implemented manipulations, it was not feasible to evaluate the model's response to all of them within the scope of this thesis. To this end, we begin by applying all implemented manipulations to a diverse selection of 20 preprocessed Project Gutenberg documents (CoFluEval-LC $_{20}$). The selected documents have an average length of 10520.9 tokens, determined by simple whitespace-tokenization (cf. Figure A.1). As an initial step, we focus on identifying the most effective manipulations with respect to each evaluation metric (cf. Section A.4 in the appendix).

Rather than exhaustively testing all manipulations, this section aims to identify those that best trigger controlled degradations in coherence and fluency, serving as reliable probes for evaluation. We aim to determine the three most impactful manipulations for each metric, which will serve as the basis for subsequent experiments in this work.

To support this analysis, we generate shortened versions of the original documents by extracting sub-documents of approximately $u = [1000, 2000, 3000]$ tokens, using a whitespace-based tokenization for simplicity. Each sub-document consists of the first u tokens of the original document, maintaining the initial structure and content sequence. To ensure readability, we avoid cutting off sentences at the end of sub-documents.

3.4.1 Vanilla Evaluation Setup

As a starting point, we employ Llama-3.3-70B-Instruct in a static, memory-less configuration, referred to as the *Vanilla Evaluation Setup*. It uses prompt `v1` (cf. Prompt B.2) to ensure a consistent and standardized evaluation procedure. To further minimize prompt-induced variance, we apply a minimal system instruction across all experiments (cf. Prompt B.1).

In this setup, the LLM performs a document-level evaluation based on quality criteria defined in the user prompt. The inference process consists of:

- **Input:** A concatenation of the evaluation prompt (`v1`), which includes detailed task instructions, and the document to be assessed.
- **Output:** A semi-structured, free-form response comprising qualitative feedback, a summary of strengths and weaknesses, and two scalar ratings - one for fluency and one for coherence - each ranging from 1 to 5.

This configuration serves as the primary point of comparison for later prompt refinements and for our memory-augmented evaluation framework introduced in Chapter 4.

Metric	$\text{Gold}_{\texttt{1K}}$	$\text{Gold}_{\texttt{2K}}$	$\text{Gold}_{\texttt{3K}}$	$\text{Gold}_{\texttt{Full}}$
Fluency	4.700	4.600	4.650	4.650
Coherence	4.850	4.850	4.750	4.650

Table 3.4: Gold Scores for all document sets (differentiated by their length/number of tokens)

As a reference point, we begin by letting Llama-3.3-70B-Instruct evaluate the unmodified (gold) documents from our 20-document subset (CoFluEval-LC $_{20}$; cf. Figure A.1), assigning scores from a Likert scale ranging from 1 (worst) to 5 (best) (cf. Table 3.4). The given scores establish a baseline for comparison with the manipulated versions. Since the selected gold documents originate primarily from the early 1900s, we consider this evaluation approach to be more accurate and appropriate than simply assuming each document would receive a perfect score of 5.

3.4.2 Matrix Analysis

To gain a more systematic understanding of manipulation effects, we conduct a matrix-based analysis focusing on the reduced subset of manipulations. These are evaluated along two primary dimensions:

- **Document Length:** represented by three fixed token counts (1000, 2000, and 3000), and
- **Manipulation Intensity:** implemented at three levels of modification strength, either count-based (number of operations applied) or ratio-based.

This setup results in a total of nine distinct configurations per manipulation, allowing us to assess the stability and sensitivity of evaluation metrics across varying document lengths and manipulation intensities.

We investigate the *Delta Value* (Δ) instead of absolute scores to measure the relative change introduced by a manipulation, making it easier to assess which manipulations have stronger or more subtle effects. Consequently, we compute the Δ-values between a manipulated document $d^{(i)}_{e',t_{\text{mani}}}$[8] and its corresponding gold document $d^{(i)}_{\text{gold}}$ as follows:

$$\Delta^{(i)}_{e,t_{\text{mani}}} = Score_e(d^{(i)}_{e',t_{\text{mani}}}) - Score_e(d^{(i)}_{\text{gold}}) \tag{3.22}$$

where $Score_e(\cdot) \in [1, 5]$ denotes the score assigned by the chosen judging LLM for the evaluation metric e.

Initially, a **count-based approach** is employed, applying the same number of operations across all sub-document lengths. Since we work with fixed-length documents, this setup helps us estimate an upper bound on manipulation intensity, indicating when subtlety begins to break down and the manipulations become overly disruptive.

Two manipulations originally designed to target coherence, *Swap Content* and *Remove Content*, do not meet this criterion. As shown in the heatmaps (Figure A.18 and Figure A.19), both manipulations lead to only minor reductions in evaluation scores, with fluency often being more affected than coherence. This trend runs counter to our expectations for effective logical-flow disruptions[9].

By contrast, *Exchange Content* produces the desired pattern (cf. Figure 3.9): Coherence scores drop significantly with increasing manipulation intensity and shorter document lengths, while fluency remains relatively stable. This aligns well with our expectations for manipulations that disrupt global discourse structure.

For testing narrative-level inconsistencies, we examine the *Temporal Inconsistency Injection* manipulation in a count-based setting (Figure A.14). While the overall impact on coherence is moderate, the effect scales with the number of inserted inconsistencies. We retain this manipulation for further experiments, as it introduces logically implausible plot elements, such as anachronisms, that challenge narrative coherence without reducing readability. The goal is not to bury subtle inconsistencies but to introduce contextually implausible events that the LLM can plausibly detect.

Lastly, the *Incorrect Verb Tenses* manipulation is evaluated to identify an acceptable manipulation threshold before coherence degrades too severely. As shown in Figure A.16, fluency scores decline notably with increasing intensity, as expected. Coherence scores also begin to drop, though more gradually. Based on these trends, we set a maximum manipulation intensity of 5% for this method to maintain the intended focus on fluency degradation without introducing unintended coherence issues.

[8] e' denotes the target metric for a manipulation t_{mani}

[9] To mitigate the issue of document shortening introduced by the *Remove Content* manipulation, a filler mechanism is used. After content removal, we append continuation text from the original document to restore its initial token count.

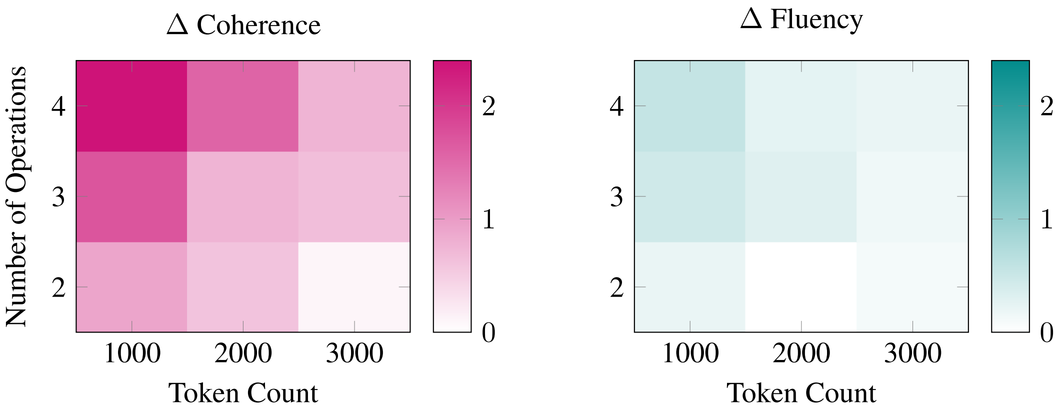


Figure 3.9: Heatmaps for **Exchange Content** [17]: Count-based analysis at paragraph-level showing the absolute change (Δ) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and number of operations.

Based on insights from the count-based strategy, we adopted a **ratio-based** approach where the number of manipulations scales proportionally with document length. This ensures a more balanced comparison across varying input sizes, addressing the inconsistency of the count-based approach, which applies a fixed number of manipulations regardless of document size and thus distorts impact, particularly in longer documents.

For the selected structural scope manipulations, this approach computes the number of operations n_o based on document length using the general formula:

$$n_o = \left(\frac{|d_{\text{gold}}^{(i)}|}{1000} \right) + b \tag{3.23}$$

where $|d_{\text{gold}}^{(i)}|$ is the token count of the gold document i and b is a manipulation-specific bias controlling the base level of disruption.

- For *exchange content*, we use $b = 2$.
- For *temporal inconsistencies*, we use $b = 1$.

For all the document-level manipulations (no matter if for fluency or coherence), we compute the number of operations as described in Section 3.3.

In contrast, manipulations such as *Swap Entities*, *Exchange Entities*, and *Entity-to-Term Replacement* rely heavily on the presence of named entities within each document. Hence, we follow a ratio-based strategy from the outset. To estimate the feasible manipulation intensity, we analyse entity distributions in gold documents. The statistics are presented in Table 3.5, which summarizes entity counts across different document lengths and entity types.

Given the distributions, we experiment with relatively high manipulation intensities to ensure that even short documents (e.g., 1000 tokens) contain enough entities to trigger manipulations.

Results are illustrated in heatmaps (Figures A.20, A.21, and A.15). For *Swap Entities*, the observed effects are minimal. *Exchange Entities* shows moderate side effects on fluency and only limited impact on coherence, with notable effects only at very high intensities. This would require replacing over half the entities in a document, which is beyond our intended subtlety threshold. *Entity-to-Term Replacement*, while also affecting fluency at higher intensities, produces more consistent and moderately declining coherence scores, suggesting a better trade-off for our purpose.

	Single-Token		Multi-Token
$\lvert d_{\textbf{gold}}^{(i)}\rvert$	**PERSON**	**LOC/FAC/GPE/ORG**	**LOC/FAC/GPE/ORG**
1000	9.3	10.3	17.05
2000	22.65	21.15	33.6
3000	33.9	32.17	50.4

Table 3.5: Average entity counts detected by spaCy's NER pipeline in gold sub-documents, grouped by entity type and sub-document token length ($\lvert d_{\text{gold}}^{(i)}\rvert$).
cf. Table A.1 for more information on spaCy's entity labels

On the fluency side, both *Typos* and *Shuffling Word Order* prove effective at degrading fluency while mainly preserving coherence (cf. Figures 3.10 and A.17). For *Typos*, we conclude that manipulation intensities between 1% and 3% are sufficient to introduce subtle but impactful degradation.

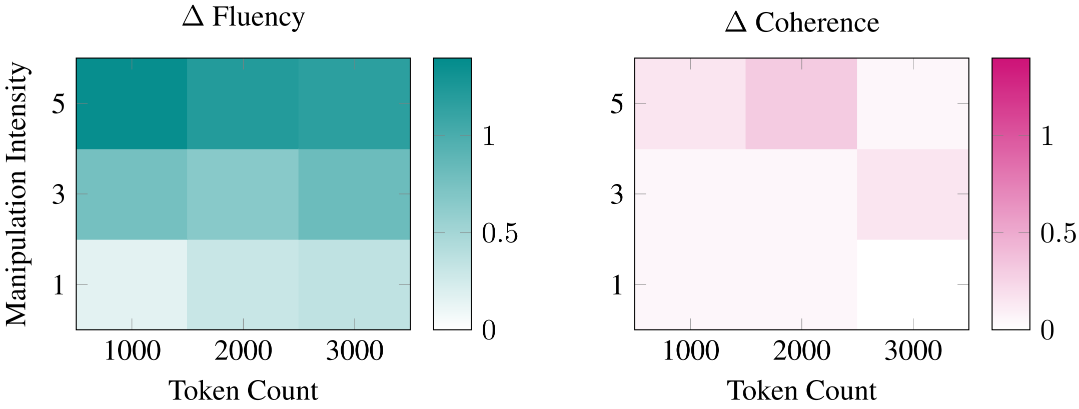


Figure 3.10: Heatmaps for **Typos** [41]: Ratio-based analysis at the paragraph level showing the absolute change (Δ) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and manipulation intensities (in percent, %).

We summarize our overall findings in Table 3.6.

3.4.3 Selected Manipulations

Based on the presented results (cf. Table 3.6), these six manipulations were selected for their ability to introduce controlled yet meaningful degradations along their respective dimensions:

In terms of manipulations targeting **coherence**, we include *exchange content* [17], *temporal inconsistency injection* [24], and *entity-to-term replacement* [31]. For **fluency**-targeting perturbations, we use *incorrect verb tenses* [42], *typos* [41], and *shuffling word order* [43].

3.5 Manipulation Validation and Sanity Check

Since we expect LLMs to encounter difficulties when evaluating long contexts, we perform a diagnostic comparison between two document length configurations as a sanity check for the selected manipulations. To this end, we use CoFluEval-SC (referred to as the 2,000-token version in previous sections) as a short-context variant of the dataset.

Mani$_{\text{ID}}$	**Affected Metric(s)**	**Analysis**	**Included**
[11]	Fluency	limited and inconsistent effect, especially on fluency; not strong or reliable enough to serve as a robust manipulation	✗
[12]	Fluency	little to no consistent impact on either fluency or coherence, suggesting it is ineffective for the intended purpose	✗
[13]	Coherence	consistent impact on coherence; slight effect on fluency; increasing document length	✗
[14]	Coherence	at paragraph+section levels limited, inconsistent effects on fluency and coherence; chapter-level manipulations cause a more substantial coherence drop but may be less practical due to structural constraints	✗
[17]	Coherence	decrease in coherence with increasing manipulation intensity and shorter document length	✓
[21]	both	effect on coherence is minimal and inconsistent; degrades fluency, especially at higher intensities	✗
[22]	both	only minor and inconsistent changes in both scores; likely too subtle or ineffective for reliably impacting text quality	✗
[24]	both	minor fluency decrease; coherence decreases with higher n_o; moderate effectiveness for coherence manipulation	✓
[31]	Fluency	effectively degrades both fluency and coherence, with stronger effects at higher intensities	✗
[32]	both	fluency decreases more with higher manipulation intensity; coherence shows moderate and fairly consistent declines across intensities and lengths	✓
[41]	Fluency	strongly affects fluency but preserves overall coherence, making it suitable for fluency-specific testing	✓
[42]	Fluency	significantly degrades fluency and moderately affects coherence at higher intensities, indicating it disrupts both surface quality and some structural flow	✓
[43]	Fluency	effectively reduces fluency and somewhat affects coherence; suitable for fluency-focused evaluation with minor structural disruption	✓
[44]	Fluency	requires a very high intensity to produce an effect which then has an equally strong impact on coherence	✗
[45]	Fluency	expected effect on fluency; effects on coherence are not neglected with increasing intensity	✗

Table 3.6: Summary of manipulation selection process. Each candidate manipulation is categorized by the actual affected metric(s) (not to be confused with the targeted metric). Only manipulations that produce a reliable, dimension-specific effect without compromising subtlety or comparability were retained for the final evaluation.

We first identify the ideal intensity settings for each manipulation that offers multiple configurations (cf. Section 3.4). Using these selected configurations, we then apply the manipulations to the full-length documents, maintaining the same manipulation ratios used for the 2K versions. This setup also allows us to examine the impact of widespread versus dense perturbation strategies at the document level.

Based on the matrix analysis results for [41] and [43], we conducted further experiments to determine the optimal manipulation intensities for these two types. We applied multiple intensity levels to the 2K versions of our selected 20-document subset. For each setting and the gold reference, we report the mean scores and the 95% confidence intervals. Results are shown in Figures A.23a and A.23b. For a manipulation to be considered effective, we expect the confidence intervals of the manipulated documents to be non-overlapping with those of the gold documents, but not too far apart, indicating that the manipulation is detectable yet subtle. The optimal settings were determined to be 5% for Shuffling Word Order and 2% for Typos.

We also examined the effect of manipulation density for the above types. Specifically, we applied the same manipulation intensity under two conditions: a dense and a widespread distribution. We then compared the resulting mean scores and 95% confidence intervals against the gold documents for the 2K subset. The results (cf. Figures A.24a and A.24b) show that dense manipulations are too subtle to be reliably detected by the model, even in short contexts. While increasing the manipulation intensity might improve detectability, it would also lead to highly corrupted document regions, resulting in incoherent and implausible content. This is not the goal of our manipulations; rather, we aim for cases where the model must detect subtle, context-level inconsistencies.

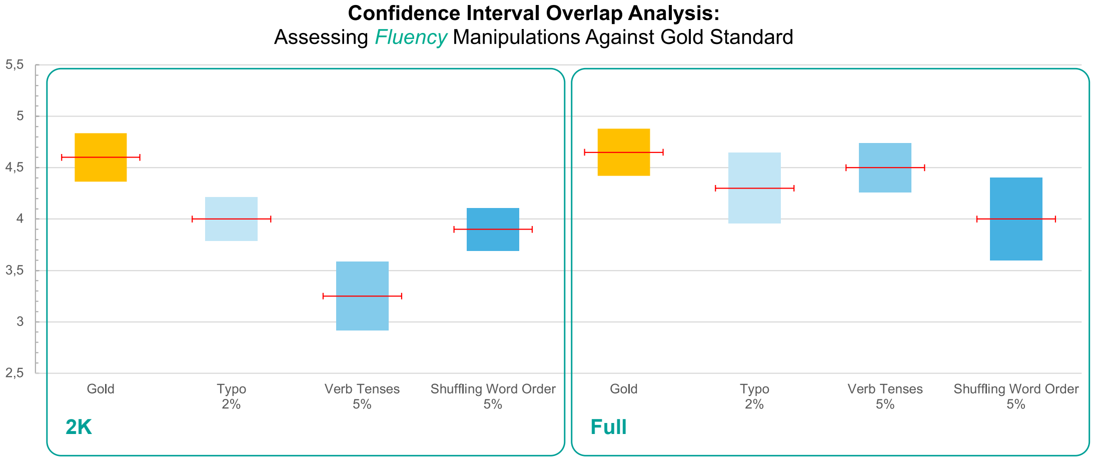


Figure 3.11: 95% confidence intervals of fluency scores assigned by the model to gold and manipulated documents in both the 2K and Full input-length settings. The red horizontal lines represent the mean fluency score obtained for each task.

Based on these optimized manipulations, Figures 3.11 and 3.12 compare the model's scoring behaviour on 2K versus Full documents for each task[10]. We report average scores for fluency and coherence, as assessed by Llama-3.3-70B-Instruct over all 20 documents, shown as red horizontal lines. The boxes represent the 95% confidence intervals for each manipulation and the gold baseline.

In terms of fluency, we observe that the 2K manipulated documents show no overlap between their confidence intervals and those of the corresponding gold documents (orange), indicating clear model sensitivity to the perturbations. However, in the Full document condition, the intervals

[10] We use the term *task* to refer to evaluating either the gold documents or one of the manipulated versions (six manipulation types in total).

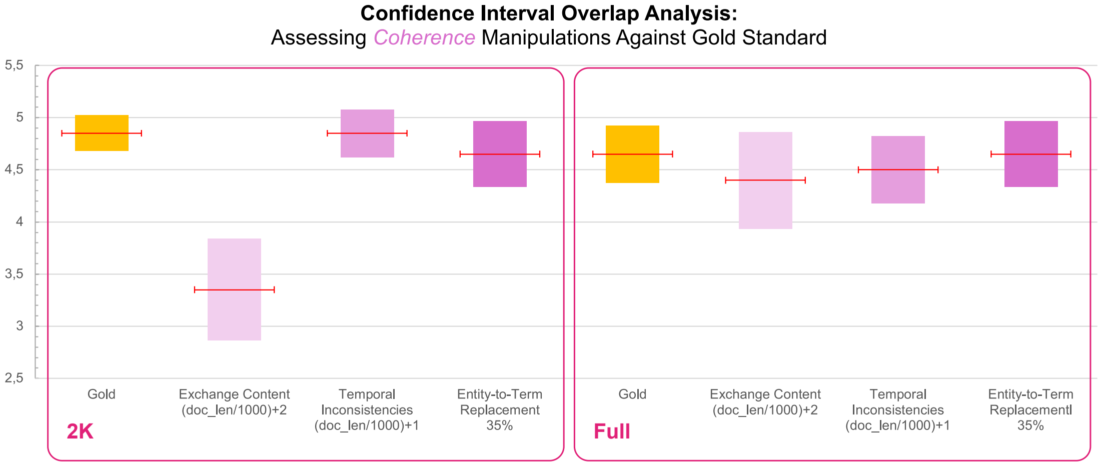


Figure 3.12: 95% confidence intervals of coherence scores for gold (orange) and manipulated (pink) documents across two input-length settings: `2K` and Full. The red horizontal lines denote the mean coherence score per task.

for [41] and [42] manipulations overlap with those of the gold documents. This indicates a reduced ability of the model to detect such manipulations in longer contexts.

For coherence, we observe a similar but less consistent pattern. Only for the [17] manipulation, we see the same separation between manipulated and gold documents in the `2K` setting. For [24] and [31], Llama-3.3-70B-Instruct already struggles to distinguish the manipulations from gold documents in the shorter `2K` condition. Nonetheless, these two manipulation types will therefore be analysed in more detail in the following sections.

3.6 CoFluEval-LC: Dataset Statistics

Our final dataset, CoFluEval-LC, includes six evaluation tasks, one for each manipulation type, as well as an evaluation task on the original gold documents, which serves as a baseline for reference scores on unaltered text. This setup enables us to measure the impact of each manipulation and assess the model's ability to evaluate the quality of the perturbed documents in CoFluEval-LC. The manipulation dataset itself consists of 390 documents in total (65 gold documents × 6 manipulation types).

We provide a high-level overview of the dataset structure in Figure 3.13, and report the parameter settings used to apply each manipulation under controlled conditions to the gold documents in Table 3.7.

Mani$_{\text{ID}}$	**Metric**	**Intensity**	**Affected Unit**	**Structural Scope**	**Density**
[17]	Coherence	$\left(\frac{\lvert d^{(i)}_{\text{gold}}\rvert}{1000}\right)+2$	of all tokens[11]	paragraph	n/a
[24]	Coherence	$\left(\frac{\lvert d^{(i)}_{\text{gold}}\rvert}{1000}\right)+1$	of all tokens	paragraph	n/a
[32]	Coherence	35%	of all GPE/LOC/-FAC/ORG entity mentions	document	n/a
[41]	Fluency	2%	of all tokens	document	widespread
[42]	Fluency	5%	of all sentences	document	widespread
[43]	Fluency	5%	of all verbs	document	widespread

Table 3.7: Configuration details for selected manipulations, including intensity, affected unit, scope, and density. Mani$_{\text{ID}}$ denotes the internal identifier for each manipulation. Each manipulation is applied to each of the 65 documents in the gold dataset $\mathcal{D}_{gold}$ (cf. Section 3.1).

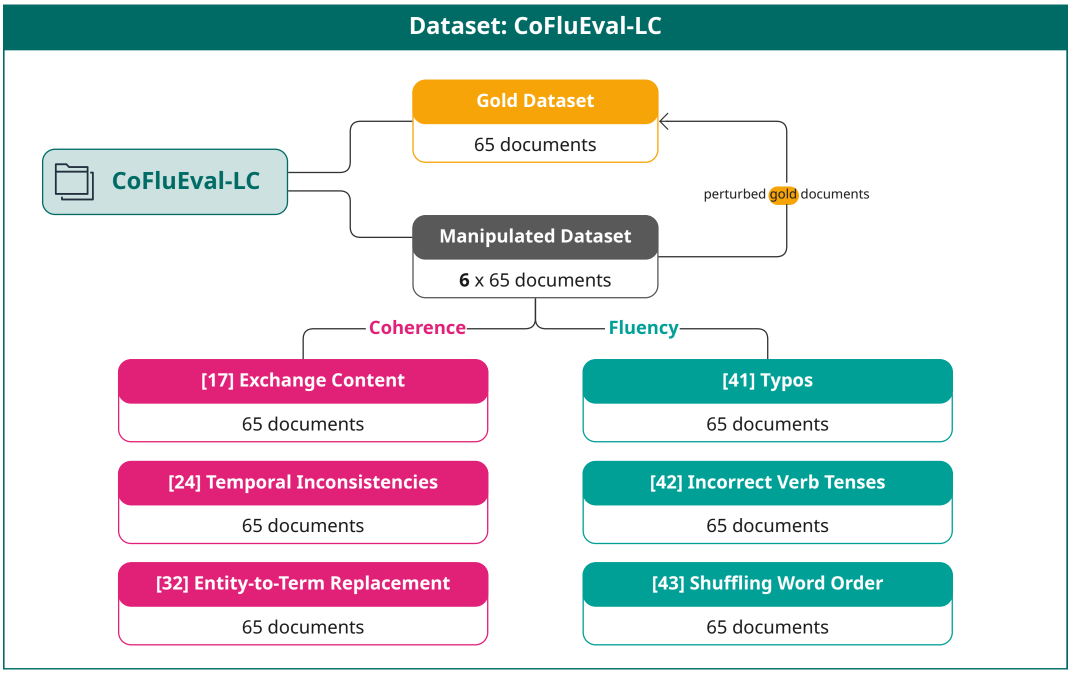


Figure 3.13: Overview of the CoFluEval-LC dataset. The dataset consists of a 65-document gold subset and a manipulation subset generated by applying six targeted manipulations under controlled conditions. Each manipulation type focuses on specific aspects of text quality such as fluency or coherence.

4 Part II - JudgeMemo: Memory-Augmented LLM Evaluation

The second major part of our contribution addresses the long-context limitations inherent in LLM-as-a-Judge evaluations. To tackle this challenge, we introduce a memory-based evaluation framework, which we refer to as JudgeMemo.

Before delving into the architecture and inner workings of the framework, we first provide insights into the prompt engineering process. This step was crucial for developing a stable, robust, and effective prompt for both our baseline LLM evaluations and the JudgeMemo evaluation procedures, which we compare in Chapter 5.

4.1 Designing Effective Prompts for Evaluation Tasks

As described in Section 3.5, we initially used a very light-weight prompt (cf. Prompt B.2) to perform sanity checks on our manipulations using Llama-3.3-70B-Instruct. Specifically, to improve the quality of these evaluations and ensure fair comparisons between the vanilla baseline evaluation approach and the method introduced in this chapter, we invested effort into prompt engineering.

Thereby, our goal was to provide the model with sufficient task-specific information without introducing bias toward our manipulations. Additionally, we required a structured output, including a clear listing of a text's main issues, to feed into our memory mechanism in JudgeMemo.

4.1.1 Iterative Prompt Refinement

Starting from Prompt `v1`, we progressively refined our approach in Prompts `v2` and `v3` by clarifying the definitions of fluency and coherence (see Tables 3.2 and 3.3) and instructing models to annotate identified issues with labeled tags, such as `[SPELLING]` or `[LOGIC]`, accompanied by brief explanations (cf. Prompt 4.1). These tags enable us to group issues by category when generating the final assessment report (see Section 4.3), allowing for a more concise and structured summary without redundant listings.

Prompt v2

```
...
mark each issue with a tag indicating its type, such as:
- [GRAMMAR] for grammar or punctuation issues
- [SYNTAX] for sentence construction problems
- [LEXICON] for word choice/redundancy
- [LOGIC] for unclear connections or reasoning gaps
- [STRUCTURE] for organisational issues across paragraphs or sections
- [CLARITY] for vague or ambiguous phrasing
- [TRANSITION] for weak or missing flow between ideas
...
```

Prompt 4.1: Excerpt from Prompt `v2`: This version introduces specific issue tagging and more comprehensive definitions of fluency and coherence to guide the model during the assessment.

Full prompt available on https://github.com/hkleiner/JudgeMemo

For the `v4` prompts, we adopted a prompting strategy from Liu et al. (2024d), guiding the model to first list relevant aspects of fluency and coherence and then apply those aspects to the actual evaluation task (`v4-2`), replacing static metric definitions with specific model-generated evaluation criteria as listed in Tables 3.2 and 3.3.

The `v5` prompt series builds on the structure proposed by Hu et al. (2024). Starting by introducing a more structured evaluation format, prompt `v5-1` surfaced key challenges such as inconsistent outputs and vague scoring guidelines. The following versions introduced clarifications and formatting constraints as well as bringing back explicit labeling of issues to support traceability. Given the high variance in assigned scores and the lack of recognizable rules, Prompt `v5-4` incorporated a more granular scoring scale, adapted from Hu et al. (2024), with explicit definitions for each score. Thereby, we give clear guidelines to the model when to award, for instance, a score of 4 instead of 5, reduce score variance and improve calibration.

Prompt v5-4

...
Metric Accuracy Scale:
FLUENCY
- Score 5: Entirely fluent, grammatically correct, and well written.
- Score 4: Only containing some minor non-fluent parts or grammatical errors that basically have no effect.
- Score 3: Fluent in general, with some obvious grammatical errors that hinder the flow of the text.
- Score 2: There are major grammatical errors, repetition, syntactic structures, and missing components, but some fluent segments.
- Score 1: Not fluent at all, full of meaningless fragments and unclear contents.

COHERENCE
- Score 5: Entirely coherent, well-structured and well-organized, building from sentence to sentence to form a coherent body of information among all the sentences.
- Score 4: Only containing some minor disconnected parts that basically do not affect the overall coherence.
- Score 3: Coherent in general, with some obvious abrupt shifts and unclarity that is not resolved in the text.
- Score 2: There are many disconnected parts and inconsistencies, but the overall context could be understandable with some effort.
- Score 1: Not coherent at all, many logical gaps, no progression, and many sections that feel disjointed or out of sync with the rest.

...

Prompt 4.2: Excerpt from Prompt `v5-4`: This prompt introduces explicit metric accuracy scales and structured issue labeling for more consistent and interpretable automatic evaluation of fluency and coherence.
Full prompt available on https://github.com/hkleiner/JudgeMemo

Later refinements, such as Prompts `v5-5` to `v5-7`, targeted specific weaknesses: verbosity in the models outputs (`v5-6`) and score sparsity. Regarding the latter, we observed that models struggled to distinguish between moderate and strong performance within the existing five-point scale, which is addressed in `v5-7` by allowing half-point ratings. This prompt also relaxed edge-case definitions to reduce the model's hesitation to assign extreme scores (i.e., 1 or 5), likely caused by overly rigid criteria, and encourage greater use of the full scale. The scoring scales used across prompts are shown in Tables 3.2 and 3.3. With Prompt `v5-8`, we reintroduce the hierarchical decomposition strategy from Liu et al. (2024d), evaluating fluency and coherence in separate prompts to minimize cross-metric interference.

Our final set of refinements (`v6`-series) explores the effects of definition granularity and metric separation. While Prompts `v6-1` and `v6-2` rely on high-level definitions, Prompts `v6-3` and `v6-4` incorporate the detailed versions for both fluency and coherence. We report results and insights in Section 4.1.3.

4.1.2 Models

Llama-3.3-70B-Instruct by AI@Meta (2024) is a text-only, multilingual, instruction-tuned autoregressive language model accessible via Hugging Face[12]. The model was pretrained on approximately 15 trillion tokens of data from publicly available sources. It features an optimized

[12] See meta-llama/Llama-3.3-70B-Instruct on Hugging Face

transformer architecture and is specifically tuned for multilingual dialogue use cases. Llama-3.3-70B-Instruct supports an input context window of up to 128K tokens. In our experiments, we utilize a quantized version of the model (FP8).

Llama-3_3-Nemotron-Super-49B-v1 is a novel large language model developed by Bercovich et al. (2025) and released in 2025[13]. It is a derivative of Llama-3.3-70B-Instruct and serves as a reasoning-focused model, supporting a context length of 128K tokens. The model was designed to reduce memory consumption by leveraging a novel Neural Architecture Search approach. It supports both reasoning and non-reasoning tasks. According to Llama-3.3-70B-Instruct, the model shares the same pretraining data with a cutoff date of 2023.

Qwen3-32B (Yang et al., 2025)[14] is the latest generation in the Qwen series of large language models. It supports both hybrid reasoning and non-reasoning modes within a single architecture. The model is multilingual and instruction-tuned, with a context window of up to 128K tokens. Qwen was trained on approximately 36 trillion tokens spanning 119 languages and dialects.

4.1.3 Prompt Evaluation Findings and Challenges

We evaluated all prompts across several open-source LLMs to better understand how prompt design influences scoring behaviour. Each model interprets prompts differently, depending on its architecture, pretraining corpus, and fine-tuning strategy. A well-designed prompt should generalize across models and reliably result in higher scores for gold documents than for their manipulated counterparts. Our evaluation focuses on three models - Llama-3.3-70B-Instruct, Llama-3_3-Nemotron-Super-49B-v1, and Qwen3-32B- with the latter two tested in both reasoning and non-reasoning configurations.

A major limitation of this setup is the small sample size: We used only five documents from the larger 20-document subset to reduce computational cost and inference time (CoFluEval-LC $_5$; cf. Figure A.2). While this subset may not be representative of the full distribution, it prevents overfitting during prompt tuning and still reveals important behavioural trends, helping to guide model and prompt selection for our method.

Following the same evaluation framework as in Section 3.5, we tested each model on both full-length documents and their truncated 2,000-token versions. Figure 4.1 reports the average fluency and coherence scores for the gold documents across all prompt versions and the tested document lengths for all models. We further analysed the effects of prompt design on how models score manipulated documents by computing the difference between the score of each manipulated document and its corresponding gold version. These differences are referred to as Δ-values (cf. Equation 3.22). Figure 4.2 presents the average Δ-values per prompt version, evaluation metric, and document length. Ideally, we seek consistently negative delta values, which indicate that models assign lower scores to manipulated documents, which is the desired behaviour when manipulations degrade fluency or coherence.

We give further insights in the Figures A.26 and A.27 as well as Tables A.7, A.8, A.9, A.10, and A.11.

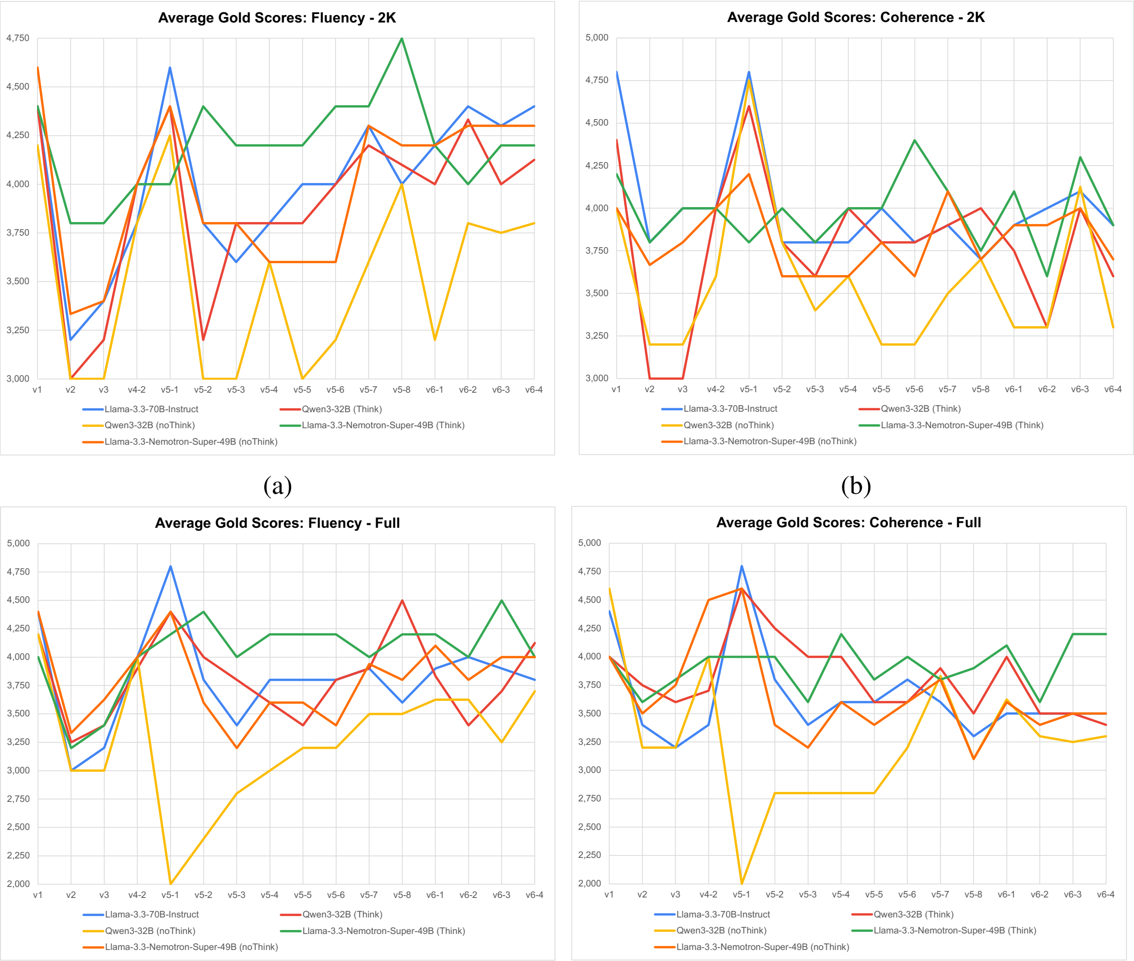
Starting with **Llama-3.3-70B-Instruct**, we observe from the gold scores that our initial prompt (`v1`) performed surprisingly well for this model and served as a strong baseline. However, subsequent prompt versions generally performed worse, with the model assigning lower scores to gold documents compared to `v1`. We hypothesize that our initial prompt formulations may have been

[13] See nvidia/Llama-3_3-Nemotron-Super-49B-v1 on Hugging Face

[14] See Qwen/Qwen3-32B on Hugging Face

too vague to encourage the model to critically evaluate the documents. Alternatively, our labeling strategy combined with detailed issue descriptions might have led to overly harsh evaluations, causing the model to overweight each issue during scoring. The results for v5-1 show that introducing a more structured prompting format helps the model produce better evaluations. However, reintroducing the labeling approach leads to lower gold scores again, suggesting this method may make the model more agnostic to issues. In general, the Full gold documents are rated slightly lower across prompts, with fluency typically receiving higher ratings than coherence.

Looking at the average manipulation deltas, we find that for full-length documents, delta values are generally lower, indicating the model struggles to detect the artificial manipulations and often rates manipulated documents similarly to gold ones. Notably, for coherence, the model sometimes overrates manipulated documents, assigning higher scores than to their corresponding gold versions (positive deltas). On the other hand, for the shortened 2K documents, the model is more effective at detecting issues, with more negative deltas, particularly in coherence.



(a)

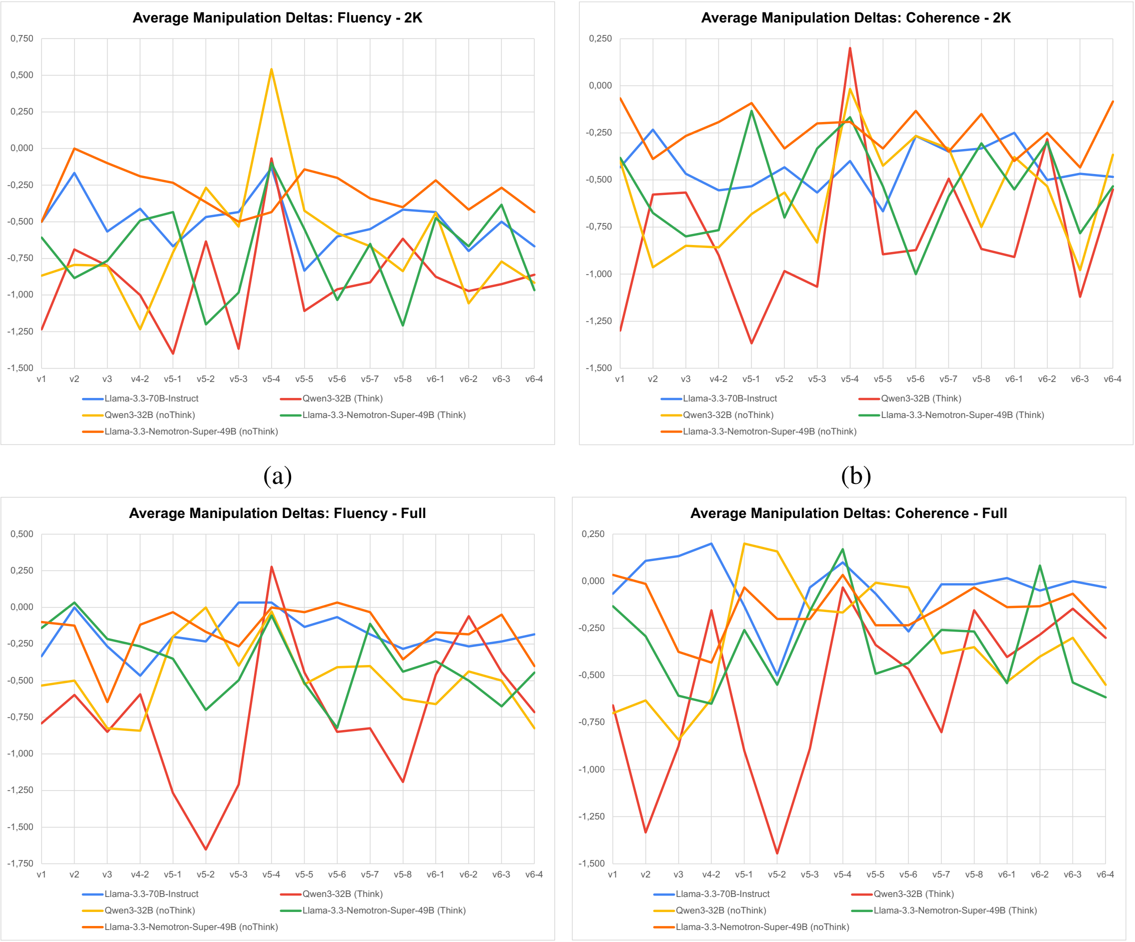
(b)

(c)

(d)

Figure 4.1: Average gold scores Score(d_{gold}, e) for each evaluation metric e. The plots show results for fluency and coherence on both the 2K and Full document settings. Each subfigure reports the average score for the five models across all prompt versions.

For **Llama-3_3-Nemotron-Super-49B-v1-noThink** (without reasoning), gold score trends mirror those already described: Earlier prompt versions underperform, and structured formats like v5-x and v6 help stabilize performance. In qualitative analysis, we observed that this model sometimes fails to generate valid or parsable responses, especially on full-length manipulated documents, often exceeding the allowed generation length or getting stuck in repetitive explanations. Structured prompts and scoring refinements help improve performance modestly, with gold scores increasing slightly and becoming more stable.



(a) (b)

(c) (d)

Figure 4.2: Average manipulation deltas (cf. Equation 3.22) for each evaluation metric e, computed across all manipulation types t. The plots show results for fluency and coherence on both the `2K` and `Full` document settings. Each subfigure reports the average delta for the five models across all prompt versions.

Contrary to the gold evaluations, the manipulated `Full` documents are often rated closer to the gold versions than their `2K` counterparts. Sometimes, they even get overrated, as seen with Llama-3.3-70B-Instruct. This suggests that LLMs continue to struggle with long-context understanding, casting doubt on their reliability, especially in evaluation settings.

Llama-3_3-Nemotron-Super-49B-v1-Think, by contrast, shows greater stability. It tends to avoid scoring `2K` gold documents' coherence below 3.750 and achieves higher fluency scores than other models. It also exhibits more consistent behaviour across prompt versions, suggesting increased robustness to prompt changes. The relatively small gap between `2K` and `Full` scores might indicate that reasoning mode improves the model's long-context handling. However, it's important to note that our evaluation is based on a limited sample size. Additionally, generation issues still occur.

Among the other models, manipulated `Full` documents generally receive less score deduction than their `2K` versions, which contradicts the assumption that these models handle longer contexts better. We again observe coherence overratings on `Full` documents and significant prompt-to-prompt variability in delta values, especially for `2K` fluency, highlighting the model's sensitivity to prompt phrasing, something not clearly reflected in gold score comparisons alone.

Qwen3-32B-noThink demonstrates high variability in scoring across prompts. Qualitative analysis reveals major generation issues, as well as ambiguity in the output, e.g., failure to clarify which metric a score refers to. Later prompt refinements help slightly, with `v6` prompts showing more stable behaviour.

When evaluating manipulated documents, `v5-4` performs poorly for this model, producing significant fluency overratings. Later prompt refinements improve the delta values, increasing reliability.

Lastly, with **Qwen3-32B-Think**, we again see that prompts like `v2` and `v3` perform poorly, while more structured prompts starting from `v5-1` lead to better evaluation behaviour. The delta distributions between `2K` and `Full` documents are similarly shaped. However, this model also suffers from generation length issues, particularly when it gets trapped in repetitive reasoning loops. This affects both gold and manipulated documents, especially at full length.

On the manipulation set, `v5-4` again fails, with the model overrating manipulated documents. It also struggles to identify manipulations in full-length texts, suggesting challenges in tracking issue impact over long contexts. That said, one might argue that some issues have less relative impact when viewed across an entire document. However, since we apply the same manipulation intensity ratio across both length settings, this should not introduce systematic bias.

Effects of Decomposition and Granularity of Definitions In our `v6` prompt series, we systematically investigate the effects of:

a) *Decomposition*: evaluating each metric in isolation using separate prompts, and

b) *Definition Granularity*: varying the level of detail in metric definitions.

Our findings (cf. Figure 4.1 and 4.2) suggest that there is no clear tendency as to which pairing works best across all models, as outcomes depend both on the model architecture and the evaluation metric in question.

However, prompts with more detailed definitions (`v6-3`, `v6-4`) consistently lead to higher gold scores and larger (more negative) manipulation deltas, especially for coherence. This suggests that definition granularity has a strong influence on evaluation quality. The effect of decomposition is less consistent: while `v6-4` (decomposed, detailed) occasionally outperforms `v6-3` (composed, detailed), the differences are small and vary by model and metric. For instance, Llama-3.3-70B-Instruct shows slightly better coherence deltas with `v6-4`, whereas Llama-3_3-Nemotron-Super-49B-v1-Think performs comparably on `v6-3` and `v6-4`.

Based on this analysis, we consider prompt `v6-3` (cf. Prompt B.3) as the most suitable prompt for both our vanilla baseline evaluations and our proposed method. It combines detailed definitions for fluency and coherence with a composed evaluation format, assessing both metrics within a single prompt.

4.2 Motivation for JudgeMemo

Our prompt engineering experiments already indicate that current models struggle to reliably process long texts (see Section 4.1). These limitations also appear in our vanilla evaluation setup, detailed in Chapter 5.

To address these issues, we propose a memory-augmented framework for LLM-as-a-Judge evaluations, designed to enable models to handle long documents more effectively and produce more reliable judgements in extended-context scenarios.

Our method draws inspiration from how humans likely approach such evaluation tasks. Rather than forming an overall judgement after a single read, human evaluators tend to take notes while reading, recording flaws relevant to the evaluation metric. As argued by Liu et al. (2024d), humans also avoid evaluating all aspects at once. Instead, they break the task into smaller, focused steps and may reread the text when needed.

We abstract this behaviour by decomposing the input document into manageable segments (up to 3,000 tokens), which models can process in a single forward pass, consistent with findings by Modarressi et al. (2025a); Levy et al. (2024). Each segment is evaluated independently, and the model's assessments are stored as structured *notes* for later use. This reduces the model's burden of internal memory retention.

In the final step, these intermediate notes are compiled into a structured report, which the model uses to assess the entire document. We hypothesize that this approach improves evaluation quality on long contexts. Additionally, we explore variations where the report is combined with either a document summary or the complete document itself. Further architectural details of the framework are provided in Section 4.3.

4.3 Method Description and Architecture

We introduce **JUDGEMEMO**, a memory-based framework designed to enhance LLM-as-a-Judge assessments in long-context settings. JUDGEMEMO decomposes the evaluation of lengthy documents into structured subtasks, which are processed through a modular pipeline to produce more interpretable and robust judgements.

As shown in Figure 4.3, the JUDGEMEMO Processor initiates the evaluation by activating a scanning module (red) that segments the input document into smaller, manageable sections. Each of these sections can optionally be passed through a summarization module (purple), which generates concise representations of their content. These summaries are useful for condensing contextual information when evaluating the sections using the judging module (green).

The `Judge` module (green), which employs an LLM-as-a-Judge approach, then evaluates each section, optionally incorporating its summary, in terms of fluency and coherence. The resulting evaluations are parsed using the parsing module (light blue) into structured scores and detected issues, forming a lightweight, interpretable memory of the document.

The `MemoryCreator` component (dark blue) consolidates all this information into a structured memory representation, which is subsequently passed to the `Judge` module for a final evaluation. This final stage can optionally include a full-document summary and/or the complete document itself, allowing the judge to take broader context into account when assessing overall fluency and coherence. The resulting final assessment is returned to the JUDGEMEMO Processor.

In the following sections, we describe the pipeline further by giving insights into its modular architecture. All modules are controlled by the JUDGEMEMO `Processor`.

4.3.1 Scanner

The `Scanner` (cf. Figure 4.3 - red) module provides functionality to divide a given text into subsections of approximately s tokens, where s is referred to as the *scan range*. To avoid introducing biases through rigid text segmentation, we do not enforce a strict section size of exactly s tokens. Instead, slight deviations are allowed when a strict cut would compromise the coherence

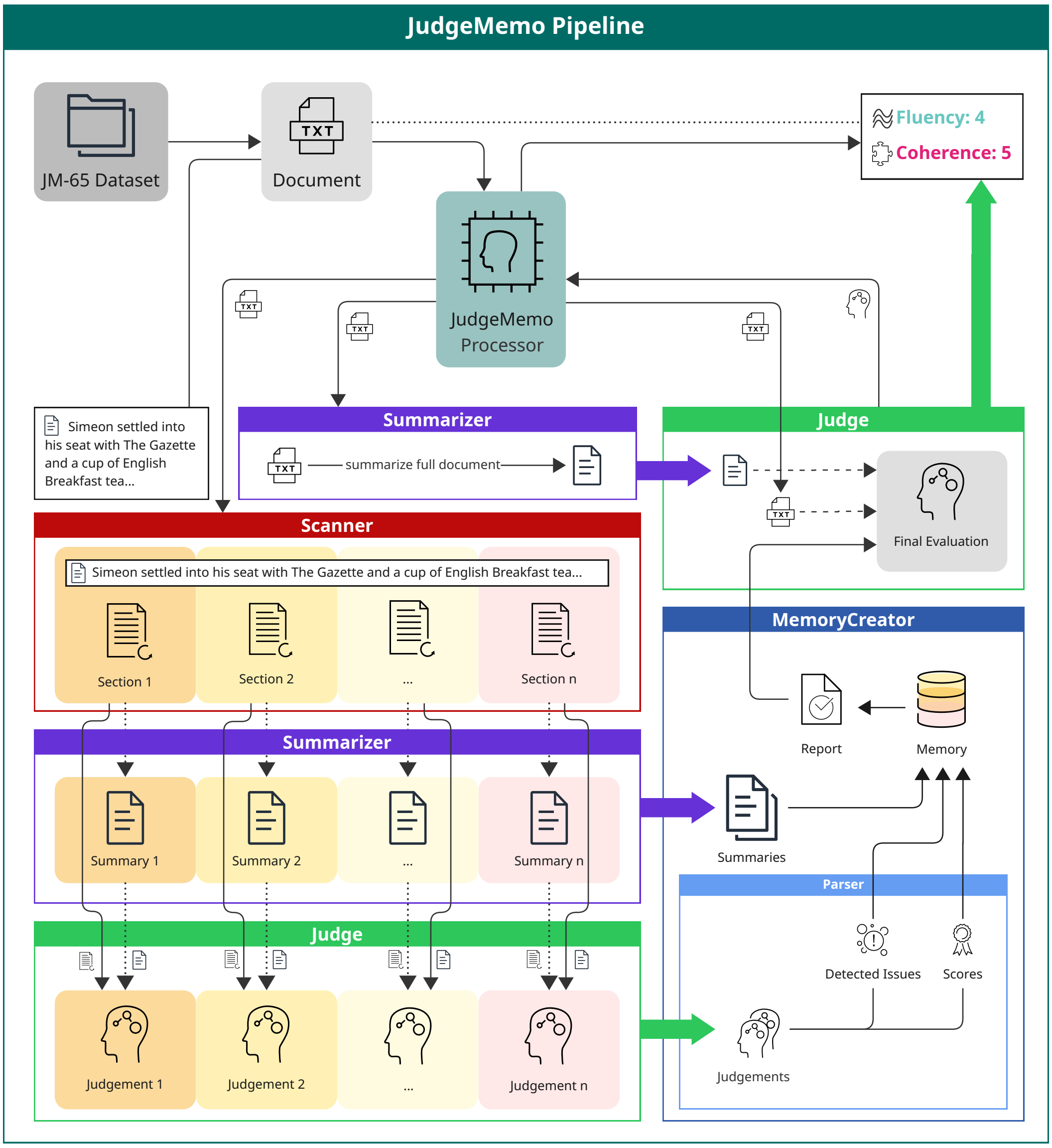


Figure 4.3: Architecture of the JudgeMemo. Starting from a text document from CoFluEval-LC, the system parses input into sections, generates multiple summaries, and passes them through a judgement model that detects issues and assigns evaluation scores for each section. The memory creation module stores structured judgements, which are used to compile the final evaluation report. The pipeline is evaluated on coherence and fluency.

or fluency of the text, particularly when it would split a sentence or word.

The choice of s directly influences the number of resulting subsections (which we refer to as *scans*) produced from the text. Each *scan* contains a unique section number, the corresponding section text, a start character index (inclusive), and an end character index (exclusive) based on the given full text.

To examine how different segmentation strategies impact the coherence of the resulting sections later on, we implement two scanning modes: *hard mode* and *stride mode*.

In **hard mode**, the text is segmented into non-overlapping sections, each approximately s tokens long. Once a section ends, the next begins immediately at the following token. This produces a sequence of

$$n = \lceil L/s \rceil \tag{4.1}$$

scans, where L is the total number of tokens in the text. See Figure 4.4 for a visual illustration.

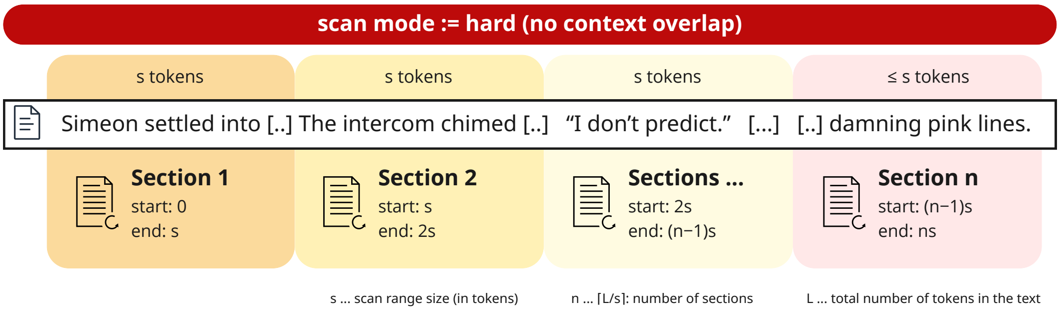

scan mode := hard (no context overlap)
s tokens
s tokens
s tokens
≤ s tokens
Simeon settled into [..] The intercom chimed [..] “I don’t predict.” [...] [..] damning pink lines.
Section 1
start: 0
end: s
Section 2
start: s
end: 2s
Sections ...
start: 2s
end: (n−1)s
Section n
start: (n−1)s
end: ns
s ... scan range size (in tokens)
n ... ⌈L/s⌉: number of sections
L ... total number of tokens in the text

Figure 4.4: Illustration of *hard* scan mode. The input text is split into consecutive sections of approximately s tokens, with no overlap. Each section starts where the previous one ended. The `start` index marks the first character of the section, and the `end` index is exclusive (i.e., it refers to the character position where the next section begins). No previous context is carried over between sections.

In contrast, **stride mode** introduces overlap between sections. Each scan still covers approximately s tokens (*start* inclusive, *end* exclusive), but also includes additional context from the preceding section. Specifically, a proportion $o \in (0, 1)$ of the scan range (i.e., $\lfloor s \cdot o \rfloor$ tokens) is taken from the end of the previous section and added as previous context attribute to the current scan. An overlap ratio $o = 0.0$ is equivalent to hard mode, while $o \geq 1.0$ would result in the entire previous section being reused as context. This scanning approach is visualized in Figure 4.5.

4.3.2 Summarizer

As an optional module, we incorporate the `Summarizer` (cf. Figure 4.3 - purple) into the pipeline. The `Summarizer` generates summaries of given inputs using an LLM when requested by the JudgeMemo `Processor`. We distinguish between two use cases: (1) summarizing scanned sections to support intermediate evaluations during the processing workflow, and (2) summarizing the entire document as part of the final evaluation input at the end of the pipeline. Both types of summarization are optional and not required for the pipeline to function.

4.3.3 Judge

Compared to our baseline model approaches, our framework includes the deployment of an LLM-as-a-Judge in two places in the pipeline (cf. Figure 4.3 - green):

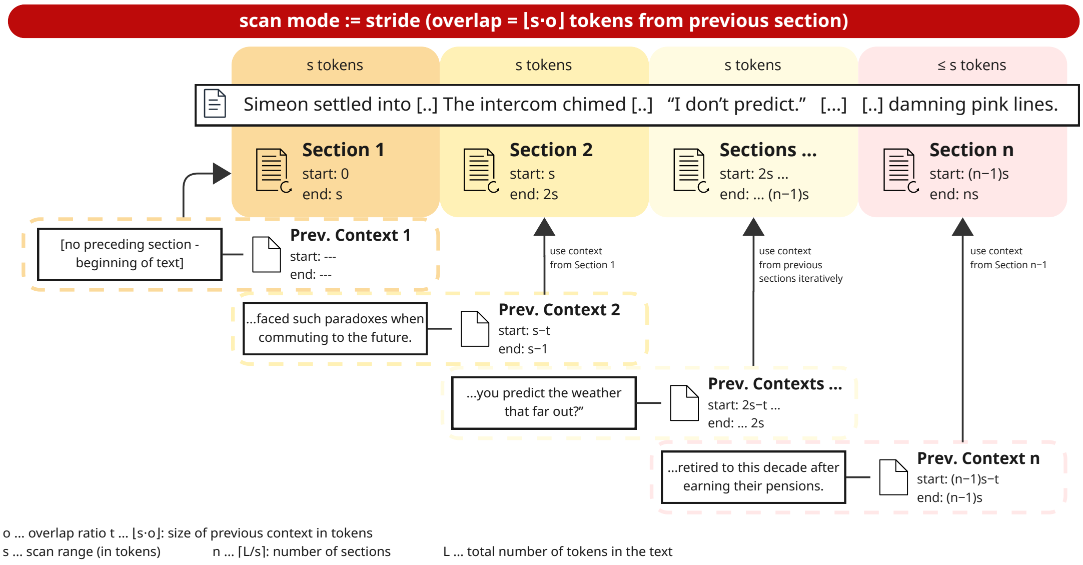

scan mode := stride (overlap = ⌊s·o⌋ tokens from previous section)
s tokens
s tokens
s tokens
≤ s tokens
Simeon settled into [..] The intercom chimed [..] "I don't predict." [...] [..] damning pink lines.
Section 1
start: 0
end: s
Section 2
start: s
end: 2s
Sections ...
start: 2s ...
end: ... (n−1)s
Section n
start: (n−1)s
end: ns
[no preceding section - beginning of text]
Prev. Context 1
start: ---
end: ---
use context from Section 1
use context from previous sections iteratively
use context from Section n−1
...faced such paradoxes when commuting to the future.
Prev. Context 2
start: s−t
end: s−1
...you predict the weather that far out?"
Prev. Contexts ...
start: 2s−t ...
end: ... 2s
...retired to this decade after earning their pensions.
Prev. Context n
start: (n−1)s−t
end: (n−1)s
o ... overlap ratio t ... ⌊s·o⌋: size of previous context in tokens
s ... scan range (in tokens)
n ... ⌈L/s⌉: number of sections
L ... total number of tokens in the text

Figure 4.5: Illustration of *stride* scan mode. Each section includes up to $\lfloor s \cdot o \rfloor$ tokens of context from the end of the previous section (shown as "Prev. Context"). Sections still advance in blocks of size s, but additional context is added to each `scan` instance (coloured boxes). The section is annotated with `start` and `end` character positions (`end` is exclusive).

1. **Intermediate Section Evaluations**: Each scanned section of the document (*scan*), produced by the `Scanner`, is evaluated individually. Depending on the selected scan mode (with or without overlap from the previous section; cf. Section 4.3.1) and whether section summarization is enabled, either both the generated summary and the section text, or only the section text, are passed to the `Judge`. If summarization is disabled and scan mode is set to *hard*, only the raw section text is evaluated, and no contextual information from preceding sections is included in the input. For all evaluations, we use a slightly adapted version of prompt `v6-3`. Regardless of the evaluation configuration, the `Judge` returns an evaluation for each section, including fluency and coherence scores, along with a list of identified issues per metric. See Figure 4.6 (left) for an example section evaluation output. These evaluation results are then attached to the corresponding *scan* and further processed by the `MemoryCreator` later on.
2. **Final Document Evaluation**: To obtain a final evaluation for each document processed through the pipeline, we apply the `Judge` module once more to assess overall fluency and coherence. In contrast to the intermediate section evaluations, the `Judge` now receives the report generated by the `MemoryCreator`, along with the full-text summary and/or the full text itself, depending on the configuration. Its task is to evaluate the complete document based on the provided input in terms of coherence and fluency, and to return a final score for each metric (cf. Prompts B.11, B.12 and B.13).

This dual-stage evaluation strategy enables a more fine-grained and holistic assessment of the document quality, allowing both localized (section-level) and global (document-level) judgements.

4.3.4 MemoryCreator and Parser

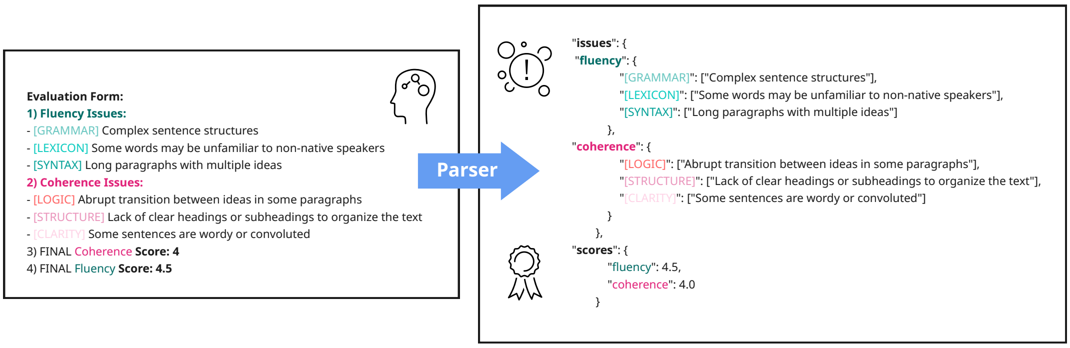
To assist in the final evaluation process, we introduce a *structured memory component* that provides three main functionalities:

1. Parsing the issues and scores from each section evaluation
2. Storing structured scan information in a memory format
3. Creating a structured report that aggregates the memory back into text format, highlighting issues and scores for each section

These tasks are embedded within the `MemoryCreator` module in our pipeline (cf. Figure 4.3, dark blue).

In the first step, before memory construction begins, the `MemoryCreator` triggers the `Parser` module (cf. Figure 4.3, light blue) to convert each section evaluation into a structured format. This parser employs a rule-based approach, utilizing a range of regular expressions to transform the textual evaluation of a section into a structured dictionary. An example is shown in Figure 4.6.

The output of the parser is returned to the `MemoryCreator` and forms a section entry, along with the optional section summary and metadata about the section's start and end positions. This process is repeated for all sections of a document to generate the complete structured memory for the text.



Evaluation Form:
1) Fluency Issues:
- [GRAMMAR] Complex sentence structures
- [LEXICON] Some words may be unfamiliar to non-native speakers
- [SYNTAX] Long paragraphs with multiple ideas
2) Coherence Issues:
- [LOGIC] Abrupt transition between ideas in some paragraphs
- [STRUCTURE] Lack of clear headings or subheadings to organize the text
- [CLARITY] Some sentences are wordy or convoluted
3) FINAL Coherence Score: 4
4) FINAL Fluency Score: 4.5

```
"issues": {
 "fluency": {
          "[GRAMMAR]": ["Complex sentence structures"],
          "[LEXICON]": ["Some words may be unfamiliar to non-native speakers"],
          "[SYNTAX]": ["Long paragraphs with multiple ideas"]
        },
"coherence": {
          "[LOGIC]": ["Abrupt transition between ideas in some paragraphs"],
          "[STRUCTURE]": ["Lack of clear headings or subheadings to organize the text"],
          "[CLARITY]": ["Some sentences are wordy or convoluted"]
        }
      },
"scores": {
        "fluency": 4.5,
        "coherence": 4.0
      }
```

Figure 4.6: Illustration of how key components from a free-text section evaluation are identified and reflected in the structured output produced by the `Parser`. Highlighted tokens indicate the alignment between the original evaluation text and the resulting structured representation.

Finally, the `MemoryCreator` generates an evaluation report used in the second `Judge` component based on the structured memory. It aggregates the parsed data into a coherent textual report, presenting section-wise issues and intermediate scores across the document.

5 Experiments and Results

In this thesis, we aim to investigate the limitations of the LLM-as-a-Judge framework when applied to long-context documents (RQ1). We hypothesize that, as demonstrated in prior work across other tasks (cf. Section 2.5), LLMs begin to struggle with fair, robust, and faithful evaluations when presented with documents exceeding 2,000 tokens (Levy et al., 2024; Modarressi et al., 2025a).

Furthermore, we evaluate our proposed method, JUDGEMEMO, on both short- and long-context evaluation tasks (RQ2). In doing so, we investigate whether incorporating structured memory and human-like evaluation behavior can help overcome the limitations associated with long-context scenarios.

We provide implementation details and further insights in Appendix A.5.

5.1 Evaluation Design

To answer our research questions, we design an evaluation task in which an LLM is given a (long) document and asked to assess its quality based on two text-level metrics: *fluency* and *coherence*. Each metric is rated on a Likert scale from 1 to 5, allowing for half-point increments (e.g., 4.5). To reduce variation stemming from the model's own interpretation of these concepts and their respective scales, we craft an evaluation prompt that includes detailed definitions for both metrics as well as accuracy scales. These accuracy scales guide the model in assigning scores by specifying the severity of issues corresponding to each rating level (cf. Section 4.1).

Our evaluation design investigates two central aspects derived from the research questions introduced in Chapter 1:

First (RQ1), we assess whether there is a significant difference in evaluation performance between short- and long-context documents in LLM-as-a-Judge setups. For this, we use our vanilla evaluation setup described in Section 3.4.1, where an LLM performs document-level evaluations solely based on the prompt and the input document. The model produces a semi-structured, free-form evaluation including a rating for each metric. This setup is a pure inference task, relying entirely on the LLM's reasoning capabilities within the provided prompt context.

Second (RQ2), we examine whether a structured, memory-based approach, inspired by human-like task-solving behaviour, can mitigate the challenges associated with long-context documents, thereby improving the reliability and consistency of LLM-as-a-Judge evaluations. To explore this, we apply our proposed method, JUDGEMEMO, to the same set of documents. JUDGEMEMO extends the classical inference task by using a document-scanning approach enhanced with memory components, allowing the model to better retain and reason over long-range dependencies and issues distributed across many tokens.

The evaluation prompt used for the vanilla baseline is `v6-3`. For JUDGEMEMO, distinct prompts are needed depending on the evaluation configuration. All of these are derived from `v6-3` and were designed to remain as close as possible to the original in order to ensure a fair comparison. However, minor modifications were necessary to accommodate the structural requirements of our method. These prompts are documented in the appendix (cf. Prompts B.7, B.8, B.9, and B.10 for section-wise evaluations, as well as Prompts B.11, B.12, and B.13 for the final evaluation stage).

5.2 Evaluation Measures and Significance Testing

In addition to the performance measures introduced earlier (i.e., average scores and Δ-values), we conduct a statistical analysis using a paired t-test, following the approach of Fayyaz et al. (2025), to assess whether the observed differences between evaluation settings are statistically significant. This test is applied across the various evaluation conditions described below.

We compute the t-test statistic z for a given evaluation metric e (fluency or coherence) and document type $d_{\text{type}} \in \{\text{gold}, t_{\text{mani}}\}$ as:

$$z = \frac{\overline{\delta_{e,d_{\text{type}}}}}{SE(\delta_{e,d_{\text{type}}})} \tag{5.1}$$

where $\overline{\delta_{e,d_{\text{type}}}}$ denotes the average difference between paired values, and $SE(\delta)$ is the standard error of those differences, defined as $SE = \frac{\sigma}{\sqrt{n}}$.[15] We set the significance level to $\alpha = 0.05$ per side.

Depending on the evaluation objective, $\overline{\delta_{e,d_{\text{type}}}}$ is defined differently:

(1) **Manipulation Effectiveness Test** within a fixed context length $c \in \{\texttt{2K}, \texttt{Full}\}$:

$$\overline{\delta_{e,d_{\text{type}}}} = \overline{\Delta^{c}_{e,d_{\text{type}}}} = \text{mean}\left(\Delta^{(i)}_{e,t_{\text{mani}}} | c\right) \tag{5.2}$$

where $i \in \{1, \ldots, |D^{c}_{d_{\text{type}}}|\}$. Each $\Delta^{(i)}_{e,t_{\text{mani}}}$ is the delta value computed for the i-th document, representing the score difference between a manipulated document and its corresponding gold document, as defined in Equation 3.22.

A *negative* t-statistic indicates that manipulated documents received *lower scores* than their respective gold counterparts, reflecting successful detection of the manipulation. Conversely, a *positive* t-statistic implies that manipulated documents were *overrated* compared to the gold versions, suggesting the model failed to recognize the injected issues.

(2) **Comparison across Context Lengths**, assessing whether long-context evaluations introduce significantly different behaviour:

$$\overline{\delta_{e,d_{\text{type}}}} = \overline{\Delta^{\texttt{2K}}_{e,t_{\text{mani}}} - \Delta^{\texttt{Full}}_{e,t_{\text{mani}}}} = \text{mean}\left(\Delta^{(i)}_{e,t_{\text{mani}}} | \texttt{2K} - \Delta^{(i)}_{e,t_{\text{mani}}} | \texttt{Full}\right) \tag{5.3}$$

where $i \in \{1, \ldots, |D^{c}_{d_{\text{type}}}|\}$. The differences are computed for matching documents across both context lengths.

A positive t-value suggests the model distinguishes less between manipulated and gold documents in the `Full` setting, possibly missing perturbations. A negative t-value indicates greater differences in the `Full` setting, implying harsher judgment of longer documents.

For the vanilla evaluation setup, we expect the t-statistic to be significant for both evaluation objectives, i.e., $z < -1.669$ or $z > 1.669$, using a critical value based on a degree of freedom $df = n - 1 = 64$ for $n = |D_{\text{gold}}| = 65$ paired samples in CoFluEval-LC per evaluation task. Here, a significant result would indicate (1) the effectiveness of our manipulations in short-context scenarios for $c = \texttt{2K}$ and (2) limitations in long-context evaluations.

In contrast, for our proposed method, we expect the t-statistic to fall within the *non-significant* range ($-1.669 < z < 1.669$) for (2), reflecting the method's ability to mitigate long-context evaluation issues and improve consistency in LLM-as-a-Judge setups.

[15] Here, σ denotes the standard deviation and n the number of paired samples.

5.3 Vanilla Baseline Evaluation

In conjunction with RQ1, we evaluate the vanilla baseline setup using Llama-3.3-70B-Instruct, Llama-3_3-Nemotron-Super-49B-v1[16], and Qwen3-32B[17]. We test the models on our proposed dataset, CoFluEval-LC, along with its 2,000-token variant, CoFluEval-SC, to simulate the short-context condition (cf. Section 3.1). Both datasets contain 65 documents per evaluation task (cf. Section 3.1 for more information on the datasets).

Effectiveness of Manipulations under Short Context We present the results of paired t-tests using $\overline{\delta_{e,d_{\text{type}}}}$ as defined in Equation 5.2, for $c = 2\text{K}$ in Figure 5.2a and Table A.12. The light blue band in the figure indicates the non-critical region (i.e., results not statistically significant). Building on these results, we evaluate the detectability of our manipulations across all models in the short-context setting. This step helps isolate whether failures to identify manipulations in longer documents are due to models' long-context limitations, rather than to weaknesses in the manipulations themselves.

Overall, all manipulations are found to be detectable in the short-context benchmark (CoFluEval-SC), with all models penalizing the manipulated documents effectively, as indicated by consistently negative t-statistics. The sole exception is Qwen3-32B-noThink, which fails to detect the *temporal inconsistency injection* [24] manipulation, slightly favoring the manipulated version, though not significantly. Unlike earlier prompt-engineering evaluations, where this model occasionally exhibited generation breakdowns, we observe no such issues in this 2K-token setting. Its reasoning-enabled variant, however, assigns scores such that the resulting Δ-values (relative to corresponding gold documents) are statistically significant ($z < -1.669$).

Among all models, Llama-3.3-70B-Instruct demonstrates the strongest average penalization ($\bar{z} = -6.452$), followed by Qwen3-32B-Think ($\bar{z} = -5.693$), while Qwen3-32B-noThink shows the weakest average signal ($\bar{z} = -3.681$).

Regarding specific manipulations, *typos* [41], *exchange content* [17], and *entity-to-term replacements* [32] are among the most easily detected, showing the highest average absolute t-statistics. Llama-3.3-70B-Instruct records the strongest individual penalization ($z = -12.646$ on [41]), while Llama-3_3-Nemotron-Super-49B-v1-noThink exhibits the most extreme t-statistic on [17]. Notably, Llama-3.3-70B-Instruct also shows heightened sensitivity to [17] and *shuffling word order* [43]. In contrast, Qwen3-32B-noThink most strongly penalizes [32], whereas its reasoning-enabled counterpart (Qwen3-32B-Think) places the most emphasis on [43].

Manipulations like [24] and *incorrect verb tenses* [42] prove more subtle across all models. While not entirely undetectable, they consistently elicit weaker penalization. Interestingly, [42] exhibits the lowest cross-model standard deviation ($\sigma = 0.992$), suggesting it is interpreted in a relatively uniform manner. In contrast, [41] displays high variability across models ($\sigma = 4.283$), pointing to inconsistent evaluative responses.

Both Llama-3_3-Nemotron-Super-49B-v1 variants perform reasonably well overall but show relative weaknesses on [41] and [42]. Given these manipulations are detectable by other models, this raises the question of whether the disruption intensity was insufficient for these two models. Despite using scoring scales in prompts to standardize evaluation, model-specific training biases may still lead to subjective interpretations, for instance, in how "distracting from reading and understanding" is perceived.

When breaking down results by evaluation category, manipulations targeting *fluency* (i.e., [43], [41], [42]) receive stronger average penalization than those targeting *coherence* ([24], [17], [32])

[16] tested in both reasoning and non-reasoning modes

[17] tested in both reasoning and non-reasoning modes

across most models, particularly for Llama-3.3-70B-Instruct ($\overline{z_F} = -8.257$, $\overline{z_C} = -4.647$). A notable exception is Llama-3_3-Nemotron-Super-49B-v1-noThink, which shows greater sensitivity to coherence-related manipulations ($\overline{z_C} = -5.531$) than fluency-related ones ($\overline{z_F} = -2.918$).

Finally, models operating in reasoning mode generally assign lower scores to manipulated documents than their non-reasoning counterparts, indicating that explicit reasoning improves evaluative accuracy. Nonetheless, they do not consistently outperform Llama-3.3-70B-Instruct, which, despite lacking explicit reasoning, exhibits robust and reliable detection performance.

	2K tokens		Full tokens		$\Delta_{\text{2K-Full}}$	
	Flu.	**Coh.**	**Flu.**	**Coh.**	**Flu.**	**Coh.**
Llama-3_3-Nemotron-Super-49B-v1 *	3.915	4.354	3.762	4.223	0.153	0.131
Llama-3_3-Nemotron-Super-49B-v1 †	4.092	4.331	3.900	4.308	0.192	0.023
Qwen3-32B *	3.677	3.615	3.730	3.738	-0.053	-0.123
Qwen3-32B †	3.985	4.069	4.038	4.123	-0.053	-0.054
Llama-3.3-70B-Instruct †	4.362	4.015	4.192	3.885	0.170	0.130

Table 5.1: Vanilla Evaluation: Average scores assigned to all gold documents in CoFluEval-SC and CoFluEval-LC across all models in the `2K`-token and `Full`-token settings. The Δ-values are computed as the difference of `2K` minus `Full` scores. Positive Δ-values indicate higher ratings in the `2K` setting, while negative values indicate higher ratings in the `Full` setting.

* no reasoning (noThink) † reasoning (Think)

Performance on Long-Context We evaluate the long-context capabilities of all models using the vanilla evaluation setup, where we compute the difference between the Δ-values (i.e., manipulated–gold score differences) obtained in the `2K` and `Full` document settings, as shown in Equation 5.3. A model that handles long context effectively should ideally assign similar scores to manipulated documents regardless of their length, resulting in statistically insignificant differences ($-1.669 < z < 1.669$) between the two conditions.

Figure 5.2c illustrates the resulting t-statistics z for all evaluation tasks across the tested models. A positive t-value indicates that the difference between manipulated and gold documents is larger in the `2K` setting than in the `Full` setting, suggesting that the model rates full documents more similarly to the gold reference, possibly because it is not detecting our perturbations. In contrast, a negative t-value indicates that the manipulated-gold difference is larger in the `Full` setting, implying a potential tendency to penalize longer documents more heavily.

Our findings reveal that long-context reasoning poses particular challenges for coherence evaluation, which is expected, as coherence goes beyond syntactic high-level features (as for fluency manipulations) and requires maintaining and recalling long-range dependencies.

For fluency manipulations, only Llama-3.3-70B-Instruct and Qwen3-32B-noThink show partially significant deviations. Manipulation [41] shows minimal variation across models, with small t-values indicating no substantial difference between short and long document evaluations. This is also reflected in the average t-statistics for fluency-related tasks, which range from $\overline{z_F} = -0.292$ for Llama-3_3-Nemotron-Super-49B-v1-noThink to $\overline{z_F} = 1.376$ for Qwen3-32B-noThink (cf. table A.13). These mostly positive average values suggest that, in general, full-document evaluations yield scores closer to the gold standard. However, this may indicate that some models fail to detect manipulations in longer documents, rather than demonstrating genuine long-context understanding.

In contrast, coherence manipulations produce significantly higher variance and more polarized results. Notably, Llama-3.3-70B-Instruct yields consistently negative t-statistics across all coherence tasks, indicating that manipulated–gold differences are larger in the `Full` setting. This could imply better detection of coherence violations in long contexts; however, we hypothesize that this may stem from a length bias, as Llama-3.3-70B-Instruct also rates gold documents lower in the `Full` condition (cf. Table 5.1). That is, while models like Qwen3-32B may overlook issues and rate long documents too indulgent, Llama-3.3-70B-Instruct might penalize longer documents regardless of their actual quality.

Overall, we find no consistent advantage for reasoning-capable models over their non-reasoning counterparts. This suggests that architectural or training enhancements intended for improved reasoning do not necessarily translate into better long-context evaluation capabilities.

```
...
- [SPELLING] "mpatriotism" instead of "patriotism"
- [SPELLING] "mpatriotism" instead of "patriotism"
- [SPELLING] "mpatriotism" instead of "patriotism"
- [SPELLING] "mpatriotism" instead of "patriotism"
- [SPELLING] "mpatriotism" instead of "patriotism"
- [SPELLING] "mpatriotism" instead of "patriotism"
...
```

Figure 5.1: Manipulation [41]: Excerpt from model-generated evaluation output for Qwen3-32B-noThink in the vanilla evaluation set up. The model repeatedly lists recurring issues, which causes it to reach the maximum generation length before assigning a final score to the document.

Additionally, when qualitative analysing the evaluation outputs for the models, we observe that the Qwen3-32B models exhibit generation issues when handling long-context inputs, particularly for fluency-related manipulations. These issues include loosing track in the reasoning process, repeating phrases or the same points multiple times, or exhaustively listing every detected issue. As a result, the models often exceed the maximum number of generation tokens. Qualitative examples of repetitive behaviour are shown in Figure A.28, while Figure 5.1 illustrates excessive issue listing.

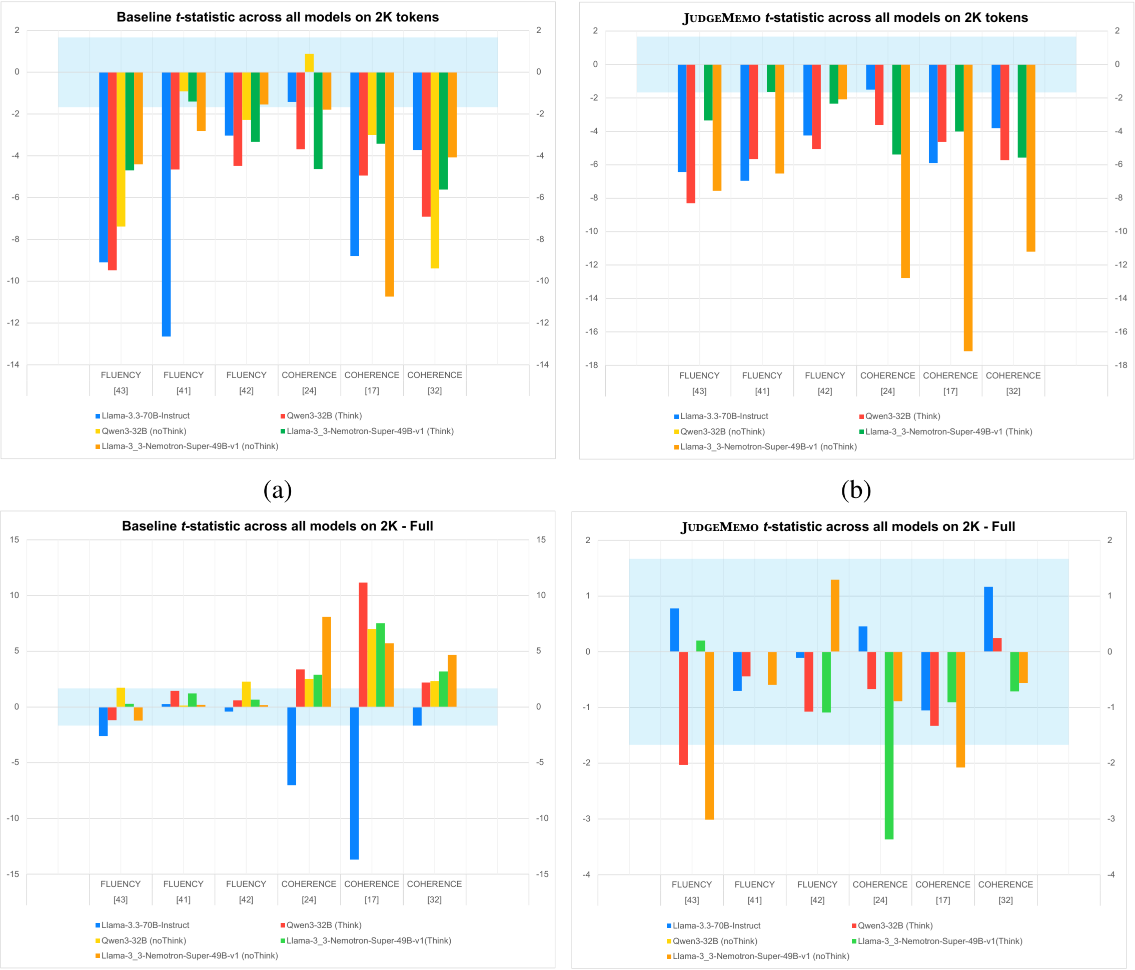
Llama-3_3-Nemotron-Super-49B-v1-Think, however, does not show this behaviour in the reasoning mode during our vanilla evaluations anymore. Without reasoning, it only struggles in the `Full` setting during the evaluation of typo-manipulated documents, showing the same behaviour as Qwen3-32B-Think, listing all occuring issues (cf. Figure A.29).

5.4 JudgeMemo Evaluation

To address RQ2 and mitigate challenges posed by long-context documents as shown in Section 5.3, we introduce and evaluate our JudgeMemo framework, which incorporates a human-like scanning behaviour combined with a structured memory mechanism.

For this purpose, we randomly sampled 30 documents from CoFluEval-LC to form a subset referred to as CoFluEval-LC $_{30}$ throughout this section. The same procedure was applied to the corresponding shortened documents in CoFluEval-SC. This sampling was performed to reduce computational resource requirements and accelerate inference. To ensure representativeness, we selected only those documents whose gold scores in the vanilla evaluation setup were close to the average across the full set of 65 documents evaluated by Llama-3.3-70B-Instruct.

In this experiment, we test our method in its baseline configuration: stride mode with $s = 0.1$



(a)

(b)

(c)

(d)

Figure 5.2: t-statistics for the vanilla evaluation setup (left) compared to JUDGEMEMO (ours, right) across all investigated models. The top two plots display t-statistics for the score differences (Δ) between gold and manipulated 2K documents, confirming the effectiveness of our manipulations. Ideally, z should fall outside the light blue area ($z < -1.669$) to indicate statistical significance. The bottom two plots show t-statistics for the difference in Δ-values between the 2K and Full document settings. These illustrate how JUDGEMEMO improves LLMs' ability to evaluate long-context inputs. Here, the ideal range is $-1.669 < z < 0$, within the light blue area, indicating non-significance and thus more stable long-context evaluation.

and without section summaries. All models evaluated in the vanilla setup (cf. Section 5.3) are tested again here using the `report only` setting, which serves as the baseline variant of our framework.

Effectiveness of Manipulations under Short Context To verify the robustness of our manipulations, we revisit their detectability in the short-context (`2K`) setting, independently of the applied evaluation design. This analysis serves to confirm that the manipulations are consistently identifiable even under constrained context lengths. As shown in Figure 5.2b and Table A.14, the t-statistics for all models (except Qwen3-32B-noThink) show substantial negative values across both fluency and coherence metrics, indicating strong penalization of manipulated documents compared to their gold counterparts. The exception, Qwen3-32B-noThink, fails to process final evaluation prompts correctly and does not produce any scores, neither in the `2K` nor in the `Full` condition. We provide further discussion of this failure in the long-context performance section.

Among all evaluated models, Llama-3_3-Nemotron-Super-49B-v1-noThink shows the strongest penalization, with an average t-statistic of $\overline{z} = -9.548$. It especially penalizes coherence-based manipulations such as *exchanging content* [17], *temporal inconsistencies* [24], and *entity-to-term replacements* [32]. Manipulation [17] results in the most pronounced degradation relative to gold documents, yielding a t-statistic of $z = -17.147$. These extreme drops in coherence-targeted manipulations appears to reflect a model-specific sensitivity, as the other models rate the same manipulation with much smaller variance. When excluding Llama-3_3-Nemotron-Super-49B-v1-noThink, the standard deviation of t-statistics across models remains low, ranging from $\sigma = 0.132$ for [24] to $\sigma = 0.206$ for [32].

The remaining three models show comparable penalization patterns across manipulation types. Qwen3-32B-Think and Llama-3.3-70B-Instruct penalize fluency-related manipulations more severely, with average t-statistics of $\overline{z}_C = -6.340 < \overline{z}_F = -4.660$ for Qwen3-32B-Think, and $\overline{z}_C = -5.884 < \overline{z}_F = -3.737$ for Llama-3.3-70B-Instruct. Similarly, Llama-3_3-Nemotron-Super-49B-v1-Think places greater emphasis on coherence-based manipulations, although the effect is less pronounced than in its non-reasoning counterpart.

In terms of statistical significance, Qwen3-32B-Think and Llama-3_3-Nemotron-Super-49B-v1-noThink produce consistently significant t-statistics across all manipulations, each averaging below $\overline{z} = -5.500$ across both metrics. Llama-3.3-70B-Instruct, by contrast, shows a non-significant result for manipulation [24], while Llama-3_3-Nemotron-Super-49B-v1-Think returns a non-significant result for [41] ($z = -1.649$). This aligns with its relatively weak penalization of fluency issues in the `2K` context.

Across all models, the most impactful manipulation is [17], which yields the lowest average t-statistic ($\overline{z} = -7.923$), suggesting strong model sensitivity to global content shifts. The least impactful manipulation is [42], with an average t-statistic of $\overline{z} = -3.436$.

Overall, our findings clearly demonstrate that the applied manipulations are highly effective in short-context settings, regardless of the evaluation framework. The few observed non-significant cases align with patterns seen in the vanilla evaluation setup, further validating the consistency of our design.

Performance on Long-Context We present our findings in Table A.15 and Figure 5.2d, which report the t-statistics across all models, computed using Equation 5.1 based on $\overline{\delta_{e,d_{\text{type}}}}$ as defined in Equation 5.3. The light blue region in the figure denotes the non-significant range ($-1.669 < z < 1.669$), indicating no statistically meaningful difference between the evaluations in the `2K` and `Full` conditions.

Compared to the vanilla evaluation setup (cf. Figure 5.2c), we observe a clear improvement in

	2K tokens		Full tokens		$\Delta_{\text{2K-Full}}$	
	Flu.	**Coh.**	**Flu.**	**Coh.**	**Flu.**	**Coh.**
Llama-3_3-Nemotron-Super-49B-v1 *	3.833	4.333	3.707	4.193	0.126	0.140
Llama-3_3-Nemotron-Super-49B-v1 †	4.033	4.400	3.767	4.233	0.266	0.167
Qwen3-32B *	—	—	—	—	—	—
Qwen3-32B †	4.008	4.117	3.742	4.042	-0.053	-0.054
Llama-3.3-70B-Instruct †	3.823	4.273	3.797	4.277	0.026	-0.004

Table 5.2: JUDGEMEMO Evaluation: Average scores assigned to all gold documents in COFLUEVAL-LC $_{30}$ and its corresponding short-context documents across all models in the `2K`-token and `Full`-token settings. The Δ-values are computed as for Table 5.1. Positive Δ-values indicate higher ratings in the `2K` setting, while negative values indicate higher ratings in the `Full` setting. Qwen3-32B-noThink did not generate valid outputs for this evaluation setup.

* no reasoning (noThink) † reasoning (Think)

long-context evaluation performance when using JUDGEMEMO. This suggests that our structured memory-enhanced prompting strategy, which simulates human evaluation behaviour, enhances the ability of models to process and rate long documents more consistently.

Notably, Llama-3.3-70B-Instruct exhibits non-significant t-statistics across all evaluation tasks, meaning it rates long documents in COFLUEVAL-LC $_{30}$ similarly to their corresponding short versions. This strongly supports the effectiveness of JUDGEMEMO in mitigating long-context degradation for this model.

For Qwen3-32B-Think, results are mixed. While it performs well on most tasks, it shows a significant deviation for the fluency manipulation [43], with $z = 3.366$. Unlike in the vanilla evaluation setup, where the results for this manipulation were non-significant, our method appears less effective in this specific case.

Among the individual manipulations, [24] remain the most challenging. For example, Llama-3_3-Nemotron-Super-49B-v1-Think exhibits lower Δ-values for `Full` documents than for their `2K` counterparts, suggesting that the model failed to detect the manipulation in the long context. This again points to a weakness in maintaining coherence over extended inputs.

Similarly, Llama-3_3-Nemotron-Super-49B-v1-noThink fails to reach non-significance on two tasks - [41] and [17] - both showing the same pattern: The manipulated-gold Δ-values are smaller in the `Full` setting, indicating a potential length-related evaluation mismatch. These deviations also may reflect difficulty in processing longer documents accurately.

Lastly, we observe that JUDGEMEMO reduces cross-model variance in t-statistics across evaluation tasks. For instance, standard deviations for tasks such as [41] and [17] are relatively low ($\sigma = 0.27$ and $\sigma = 0.45$, respectively), indicating more stable behaviour across models.

Overall, these results validate the effectiveness of JUDGEMEMO in improving long-context evaluation robustness and reducing model variance, especially for coherence-focused tasks where prior evaluations showed substantial instability.

Furthermore, in the qualitative analysis of model outputs, we again observe that Qwen3-32B-noThink fails to score documents across all experimental settings (cf. Figure 5.3). It consistently returns a default response indicating that it is awaiting further instructions, stating that it has not received the task. As shown in Figure 5.3, this response is produced instead of any evaluation output. We hypothesize that this behavior results from model-specific prompt issues, as it occurs not only in the `Full` setting, where input length could be a concern, but also in the short-context versions. This suggests that the model may require a specifically tuned prompt to handle the task correctly.

The shortcomings may also be connected to the fact that the non-reasoning version underperforms here, highlighting weaker capabilities compared to its reasoning-enabled counterpart.

```
<think>

</think>

Sure!  Please provide the text you'd like me to rate,
and let me know if you have any specific criteria or
guidelines in mind (e.g., grammar, clarity, coherence,
creativity, etc.).
```

Figure 5.3: Manipulation [41]: Excerpt from model-generated evaluation output for Qwen3-32B-noThink in the vanilla evaluation set up. The model repeatedly lists recurring issues, which causes it to reach the maximum generation length before assigning a final score to the document.

Across all experiments, and particularly under our proposed method, Llama-3.3-70B-Instruct demonstrates the most consistent generalization and long-context performance. It shows no signs of verbosity or other generation issues and consistently adheres to the provided instructions. For these reasons, we selected it as the model for conducting our ablation studies in Section 5.5.

5.5 JudgeMemo Ablation Studies

To better understand the components contributing to the effectiveness of our JUDGEMEMO framework, we conduct a series of ablation studies using Llama-3.3-70B-Instruct as the evaluation model (cf. Section 5.4). These experiments isolate and examine individual design decisions within the `Scanner` and `Judge` modules.

Specifically, we analyse (i) the impact of different scanning strategies, (ii) the role of section-level summaries and (iii) reporting strategies, including adding contextual signals for evaluation, and (iv) the overall importance of context-awareness during the judging process. Together, these ablations provide insight into which elements of JUDGEMEMO are most critical for improving long-context evaluation fidelity.

Scan Range Size We first investigate the effect of different scan range sizes in the section creation process of the `JMScanner`. We chose to look at $s \in \{1000, 2000, 3000\}$ tokens. For this experiment, we evaluate Llama-3.3-70B-Instruct in the `report only` setting, using *hard* scanning mode and omitting section-level summaries to isolate the impact of scan size. Figure A.30 presents the resulting average scores and corresponding Δ-values per evaluation task.

Overall, increasing the scan range tends to *slightly* improve gold document scores across both fluency and coherence metrics, with coherence scores rising from 4.227 at $s = 1000$ to 4.280 at $s = 3000$, and fluency from 3.767 to 3.843. This trend suggests that longer scan segments provide the model with a more comprehensive view of the document, leading to more favorable and consistent gold ratings.

When examining the Δ-values, we observe a *minimal* increase in penalization strength with larger scan ranges. For example, in evaluation task [43], the fluency Δ drops from -0.217 at $s = 1000$ to -0.310 at $s = 3000$, and the Δ_C for the same task falls from -0.364 to -0.468. A similar pattern is seen for [41], where the coherence gap remains consistently high across all scan sizes (> -0.9), but deepens slightly at $s = 3000$.

While these shifts are not dramatic, they indicate that larger scan ranges can marginally enhance manipulation detectability, especially in coherence-targeting tasks. However, the increasing standard deviations for both scores and Δ-values also suggest that higher scan ranges may introduce more variability in model responses.

Taken together, these results highlight a trade-off between context coverage and prediction stability, where moderate scan sizes ($s = 2000$) may provide a good balance for reliable section-level evaluation.

Influence of Scanning Modes As we differentiate between *hard* and *stride* scanning modes in the section creation process of the `Scanner`, this affects the contextual content of the input documents passed to the `Judge` for evaluation. For further details on the implementation and behaviour of these scanning modes, please refer to Section 4.3.1.

We analyse their effect under the `report only` setting, using the context-aware[18] configuration and disabling section-level summaries to isolate the impact of scanning strategy alone. For *stride* mode, we set the overlap ratio to $o = 0.1$, resulting in a 10% token overlap between consecutive scan segments, relative to the section size.

Figure 5.4a shows the paired t-statistics computed on the Δ-values between the `2K` and `Full` document settings for the `report only` configuration. In our analysis, we focus on the dark blue and dark orange pillars, which correspond to the setups without section summaries in *hard* ($o = 0.0$) and *stride* ($o = 0.1$) scanning modes, respectively.

The results indicate no consistent advantage for either scanning strategy across all evaluation tasks. For example, in task [43] (fluency), the *hard* mode yields a non-significant t-statistic of $z = 0.964$, while the *stride* mode performs similarly with $z = 0.779$. In task [24] (coherence), however, the hard mode shows a more pronounced deviation ($z_{0.0} = 2.528$ vs. $z_{0.1} = 0.456$). This suggests that scan overlap might help reduce inconsistencies when contextual reasoning is required, as opposed to relying solely on high-level syntactic analysis.

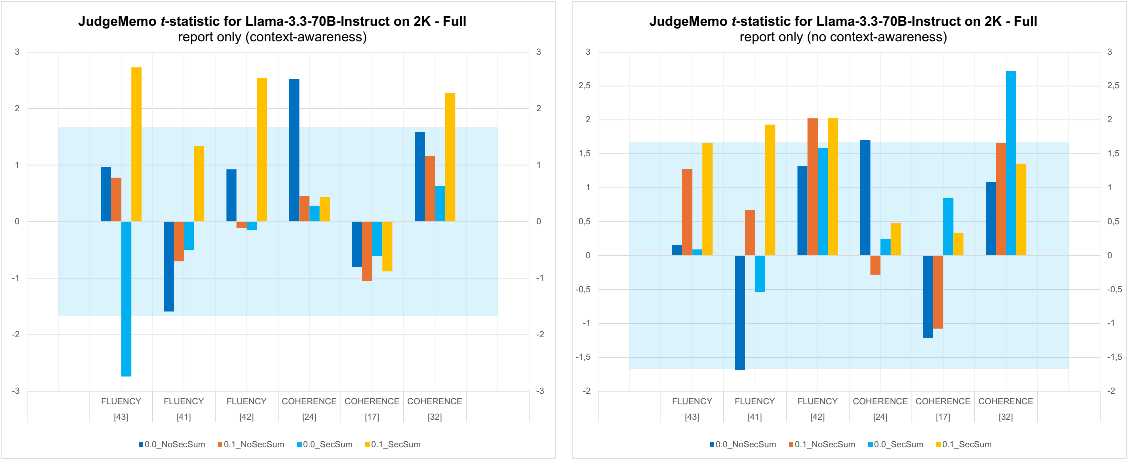
By contrast, the differences observed for the coherence-related task [32] remain within the non-significant range for both modes, again showing no clear preference for one scanning strategy. We hypothesize that this difference stems from the nature of the manipulations themselves: Detecting [24] likely requires tracking and reasoning over longer dependencies across the document, whereas in [32], the manipulated entities are more localized and thus easier to resolve within shorter spans.

Overall, these results suggest that while scan overlap (*stride* mode) can improve contextual continuity between segments, its benefits are highly task-dependent and not uniformly significant across all types of manipulations.

Influence of Section Summaries We now analyse the impact of including section summaries in the `report only` setup, again comparing *hard* ($o = 0.0$) and *stride* ($o = 0.1$) scanning modes. The relevant paired t-statistics are shown in Figure 5.4a, with light blue and light orange bars corresponding to runs with section summaries enabled.

Across tasks, we observe a more distinct effect of adding section summaries than of scan overlap alone. For instance, in task [43] (fluency), switching from no section summaries to section summaries yields a large shift: The *hard* mode result drops from a non-significant $z = 0.964$ to a strongly significant $z = -2.74$, while the *stride* mode jumps from $z = 0.779$ to $z = 2.731$. This

[18] Please note that *context-awareness* here refers to the prompting strategy and not the inclusion of prior document context within section evaluations.



(a) (b)

Figure 5.4: Ablation studies for JUDGEMEMO (ours) in the `report only` setting on Llama-3.3-70B-Instruct. Left plots use context-aware prompts; right plots use non-context-aware ones. t-statistics are computed over the difference in Δ-values between `2K` and `Full` documents. Values within the light blue area indicate non-significance, suggesting consistent performance across context lengths.

suggests that including summaries can amplify differences in Δ-values between `2K` and `Full` documents, possibly by introducing biases or misalignments in how context is aggregated.

In task [42] (fluency), we again see a shift from non-significance without summaries ($z_{0.0} = 0.925$, $z_{0.1} = -0.109$) to a significant deviation in the stride setting with summaries ($z_{0.1} = 2.546$). Similarly, coherence-related task [32] shows an increase in z-values with summaries, most notably under stride scanning ($z_{0.1} = 2.279$).

That said, the impact is not uniformly significant across all tasks. In coherence task [24], the addition of summaries appears to dampen the effect observed without them ($z_{0.0} = 2.528$ reduces to $z_{0.0} = 0.283$ with summaries). Likewise, task [17] remains in the non-significant range across all settings.

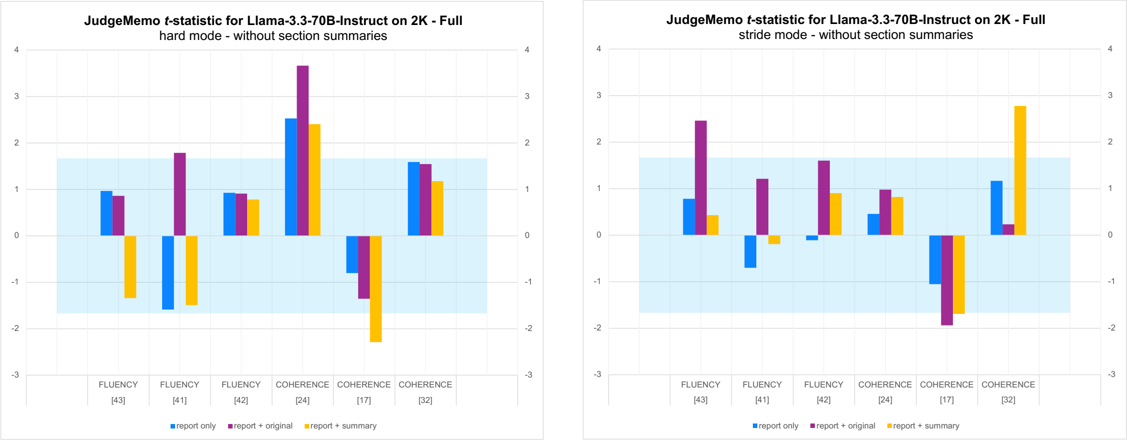
In sum, while section summaries can significantly alter model behaviour, sometimes increasing sensitivity to long-context differences, they do not universally improve evaluation quality. Their effectiveness appears tightly coupled to the nature of the manipulation and the scanning strategy used.

Influence of Reporting Strategies To assess the effect of different reporting strategies on the final evaluation in our method, we compare three configurations: (i) `report only`, where only the intermediate section evaluation report is used, (ii) `report + original`, which adds the full document text alongside the report, and (iii) `report + summary`, which pairs the report with a summary of the full document. We run evaluations for both *hard* scan mode ($o = 0.0$) and *stride* scan mode ($o = 0.1$), each without section summaries.

***Hard* Mode** As shown in Figure 5.5a, the `report only` setting yields the most stable performance, with all t-statistics staying within the non-significant range (except for [24], $z = 2.528$). In contrast, `report + original` introduces larger deviations, notably in [24] ($z = 3.665$) and [41] ($z = 1.787$), suggesting that reintroducing the full document text may harm evaluation fidelity. This supports our hypothesis that excessive prompt length or redundant content can dilute model judgement rather than reinforce it.

The `report + summary` setup performs similarly to `report only` but with greater variance. For instance, task [17] yields $z = -2.288$, indicating reduced stability. While summaries provide context continuity, they may obscure fine-grained cues needed to assess coherence-based manipulations accurately.

***Stride* Mode** We observe broadly consistent trends in Figure 5.5b. Again, the `report only` configuration shows the most stable performance, with all $|z| < 1.3$ across tasks. In contrast, `report + original` results in stronger deviations for several tasks, notably [43] ($z = 2.462$) and [42] ($z = 1.601$). Interestingly, `report + summary` leads to a large deviation in task [32] (coherence, $z = 2.777$), reinforcing our earlier observation that summaries might fail to preserve manipulation-relevant cues in long-range coherence tasks.



(a) *hard* mode, $o = 0.0$

(b) *stride* mode, $o = 0.1$

Figure 5.5: Ablation studies for JUDGEMEMO (ours) focusing on the reporting strategy used for final evaluation. t-statistics are computed over the difference in Δ-values between `2K` and `Full` documents. Values within the light blue area indicate non-significance, suggesting consistent performance across context lengths.

Overall, these results suggest that relying solely on the generated report from section-level judgements (`report only`) provides the most stable and context-length-robust setup across both scanning strategies for JUDGEMEMO. This setup also aligns well with our goal of efficient and faithful long-context evaluation.

Further results involving different scan modes and the inclusion of section summaries are presented in Figure 5.4 and Figure A.31.

Impact of Context-Aware Prompting Finally, we assess the influence of contextual phrasing in section-level evaluation prompts. In our default setup, we insert the following instruction into the prompt:

> *IMPORTANT: The text provided is a section extracted from a longer document. It may be preceded or followed by other parts not shown. Evaluate the section on its own terms, while allowing for the possibility that some context may lie outside the visible excerpt.*

We hypothesized that this phrasing may lead the model to overlook coherence-related issues by assuming that discontinuities might be resolved by unseen context. To test this, we reran all evaluations under the `report only` setting using `stride` mode ($o = 0.1$) on Llama-3.3-70B-Instruct, comparing the default (context-aware) prompts to a version without the contextual instruction.

As shown in Table 5.3, removing the contextual phrasing consistently resulted in more negative t-

statistics across coherence-targeting manipulations in both the `2K` and `Full` settings. For instance, in task [24], the `2K` setting yields $z = -3.084$ without contextual phrasing compared to $z = -1.505$ with it, indicating stronger sensitivity to the manipulation when the model is forced to evaluate the section in isolation. Similar trends hold for the evaluations on manipualtions [32] and [17].

Moreover, the average coherence scores for manipulated documents are slightly lower without contextual phrasing (e.g., 4.158 vs. 4.267 for task [24] in 2K), further confirming that the contextual hint softens judgement severity.

These findings suggest that, while the contextual phrasing may be intuitively helpful for grounding judgements, it introduces ambiguity in the evaluation of manipulated content, particularly for coherence. We therefore recommend omitting such phrasing when the goal is to assess coherence at the section level with high fidelity (cf. Figure 5.4 and Figure A.31).

		2K		**Full**	
$Mani_{\text{ID}}$	**Metric**	z **(no ctxt)**	z **(ctxt)**	z **(no ctxt)**	z **(ctxt)**
[24]	Coherence	-3.084	-1.505	-4.604	-3.914
[17]	Coherence	-5.656	-5.899	-7.441	-7.704
[32]	Coherence	-3.430	-3.806	-5.585	-5.559
$Mani_{\text{ID}}$	**Metric**	$\overline{Score}$ **(no ctxt)**	$\overline{Score}$ **(ctxt)**	$\overline{Score}$ **(no ctxt)**	$\overline{Score}$ **(ctxt)**
[24]	Coherence	4.158	4.267	4.133	4.167
[17]	Coherence	3.992	3.992	3.997	3.970
[32]	Coherence	4.002	4.042	3.878	3.890

Table 5.3: Paired t-statistics and average scores for Llama-3.3-70B-Instruct using `report only`, *stride* mode ($o = 0.1$) on CoFluEval-LC $_{30}$. Removing contextual phrasing results in stronger penalization of coherence manipulations.
ctxt ... context

5.6 Key Insights and Takeaways

Our experimental results yield several key insights into the limitations of LLM-as-a-Judge setups for long-context documents, as well as the effectiveness of our proposed JudgeMemo framework in mitigating these challenges.

(1) LLMs Struggle with Long-Context Evaluations Across all tested models and evaluation tasks, we observe a consistent degradation in performance when transitioning from short-context (2K-token) documents to full-length ones. In the vanilla evaluation setup, even state-of-the-art models like Llama-3.3-70B-Instruct and Qwen3-32B-Think exhibit statistically significant differences in their assessments, especially on coherence-targeted manipulations. This confirms our hypothesis (cf. Chapter 1) and aligns with prior findings on the long-context limitations of LLMs (cf. Section 2.5). Notably, these issues are not confined to a specific architecture or reasoning strategy: even models explicitly trained for enhanced reasoning (Qwen3-32B-Think, Llama-3_3-Nemotron-Super-49B-v1-Think) fail to consistently overcome these challenges.

(2) Coherence is More Susceptible Than Fluency Our analysis shows that manipulations targeting coherence, such as *entity-to-term replacements* or *temporal inconsistencies*, are more frequently undetected in long-context scenarios compared to those affecting fluency (e.g., *typos*, *shuffeling word order*). This is especially evident in the higher t-statistic variance and greater number of significant deviations in coherence-related tasks. Since coherence often relies on long-range dependencies and document-level structure, this reinforces the need for more robust memory mechanisms or evaluation strategies when assessing global quality aspects.

(3) The JudgeMemo Framework Mitigates Long-Context Issues Our proposed framework, JUDGEMEMO, substantially improves the consistency of LLM evaluations across context lengths. In contrast to the vanilla setup, t-statistics for most tasks fall within the non-significant range ($-1.669 < z < 1.669$), indicating that manipulated-gold differences are stable across `2K` and `Full` document settings. Importantly, this improvement is achieved without sacrificing manipulation detectability: Short-context evaluations under JUDGEMEMO remain robust, as confirmed by consistently negative and significant t-values.

(4) Design Choices in JUDGEMEMO Matter Through our ablation studies, we identify several factors contributing to JUDGEMEMO's effectiveness:

- **Scan Range**: Larger scan sizes (e.g., 3000 tokens) slightly improve manipulation detection but introduce more output variance. A size of 2000 tokens offers a balanced trade-off.
- **Scan Mode**: Stride scanning (with overlap) helps maintain context continuity, especially for coherence-related tasks, though its benefits are task-dependent.
- **Section Summaries**: While potentially helpful, summaries can introduce bias and increase variability. In many tasks, excluding summaries yields more stable results.
- **Reporting Strategy**: Using only the intermediate report (`report only`) outperforms more elaborate setups that include the full document or a summary alongside the report. This suggests that concise, structured inputs support more reliable scoring.
- **Context-aware Phrasing**: Surprisingly, prompting the model to consider unseen context (e.g., "this may be part of a longer document...") can soften judgement severity and reduce manipulation detection. For high-fidelity coherence evaluations, this phrasing should be omitted.

(5) Model-Specific Patterns Persist Despite the overall benefits of JUDGEMEMO, we still observe model-specific behaviours. For instance, Qwen3-32B-noThink fails in short- and long-context evaluation under our method, possibly due to input length confusion or prompt misinterpretation. Llama-3.3-70B-Instruct, on the other hand, demonstrates the most robust and stable performance across all tasks and configurations, making it a reliable choice for evaluation-heavy workflows.

6 Conclusion

In this thesis, we addressed a pressing challenge in the evolving landscape of language model evaluation: the reliability and effectiveness of LLMs when used as automated judges in long-context scenarios. While the LLM-as-a-Judge paradigm promises scalability and cost-efficiency over traditional human annotation, its applicability to long documents, ranging from narratives and essays to multi-turn conversations, remains severely limited. Despite architectural advances enabling models to process up to 128K tokens, our research confirms growing evidence that true comprehension and reasoning degrade significantly with increased input length. This limitation casts doubt on the validity of long-context evaluations performed by current LLMs.

To address this, we proposed JUDGEMEMO, a novel memory-augmented evaluation pipeline inspired by human strategies such as note-taking, iterative judgement, and externalized memory use. Our approach divides long documents into smaller, coherent units, processes them through an LLM to generate intermediate memory in the form of structured section-wise reports, and reintroduces this memory to guide final quality assessments. Thereby, JUDGEMEMO alleviates the need for the model to internally retain all contextual information during a single forward pass, thus reducing context loss and enhancing overall judgement consistency.

First, we introduced COFLUEVAL-LC, a new dataset specifically tailored for testing the evaluation abilities of LLMs under long-context constraints. This dataset is composed of 65 carefully curated documents sourced from Project Gutenberg, all ranging between 8,000 and 16,000 tokens. These documents were manipulated using rule-based perturbation techniques targeting two core evaluation metrics: fluency and coherence. Our manipulations were designed with varying scope (paragraph, section, chapter, document-level) and granularity (character vs. token-level) to simulate realistic degradation while maintaining control over content distortion. This systematic design enabled us to conduct a fine-grained analysis of LLM performance across diverse manipulation types.

In our experiments, we compared two major setups: (1) a vanilla LLM-as-a-Judge pipeline that evaluates documents as they are in a single forward pass, and (2) the proposed JUDGEMEMO framework that incorporates intermediate analysis of smaller sections and memory. Our findings highlight several important insights:

Long-Context Degradation Is Significant and Measurable (RQ1): Standard LLM evaluation setups consistently show a decline in performance as input length increases, often failing to detect subtle yet meaningful degradations in fluency and coherence.

JudgeMemo Improves Evaluation Performance and Robustness (RQ2): Integrating structured memory leads to more stable fluency and coherence judgements over longer documents. By externalizing memory into structured section-wise reports, the model can maintain contextual awareness and avoid misattribution errors.

Memory Component Ablation Confirms Architectural Contribution: Ablation studies show that removing or simplifying individual components of JUDGEMEMO leads to measurable declines in evaluation performance. This suggests that the pipeline's modularity and structure play a key role in its success.

Structured Prompts and Output Formats Improve Parsability and Interpretability: Our prompt design contributed to improved response consistency and reduced output variance. By explicitly defining evaluation criteria and scoring rubrics, we minimized metric confusion and allowed for clearer downstream analysis of the model's decisions.

Beyond performance metrics, the JUDGEMEMO architecture offers qualitative benefits as well. First, it aligns more closely with human evaluative strategies, which rely on note-taking, hierarchical processing, and multi-stage reasoning. Second, its modular design lends itself to extensibility. Third, it offers greater transparency and interpretability, as intermediate section-scorings and memory-entries can be inspected, edited, or audited by human users, a property of growing importance in responsible AI deployment.

In conclusion, JUDGEMEMO demonstrates that structured memory integration is not just an architectural enhancement. It is a necessary evolution for building LLM-based evaluators that are robust, interpretable, and aligned with real-world demands. As LLMs are increasingly used in evaluation, governance, and oversight roles, our findings offer a pathway toward more trustworthy and scalable assessment systems.

7 Limitations and Future Work

While our findings contribute meaningful insights, several limitations remain due to the scope of this work and the experimental constraints, which point toward promising directions for future research.

Dataset Scope and Representativeness Our study is based on a carefully curated dataset, CoFluEval-LC, composed of narrative documents from the Project Gutenberg web corpus. These documents are publicly available and suitable for research use. However, a significant portion of them are over 50 years old (cf. Decade Distribution in Figure 3.1a), largely because copyright restrictions limit access to more recent literary works. As a result, the language, stylistic choices, and cultural references in these texts may not reflect contemporary writing practices or societal norms.

This temporal mismatch introduces a potential domain gap between the documents in CoFluEval-LC and the data on which current LLMs were pre-trained. LLMs like LLaMA (Grattafiori et al., 2024) or GPT-4 (OpenAI et al., 2024) were primarily trained on large-scale web, book, and code corpora. These sources emphasize recent language usage and broader topical diversity. Consequently, the models might not align well with the older texts in our dataset, which may affect their evaluation performance due to lexical, syntactic, and stylistic discrepancies, i.e., a form of domain shift or dataset bias.

Additionally, our dataset focuses solely on narrative texts, such as fiction, romance, historical texts, and novels. This narrow domain, while appropriate for exploring long-context evaluation within a coherent genre, limits the generalizability of our results. Many long-form documents encountered in real-world applications, such as legal rulings, scientific publications, technical manuals, and financial reports, differ significantly in structure, style, and purpose. Future work should assess LLM evaluation performance in these other domains to determine whether the challenges identified in narrative evaluation generalize to other forms of long-context reasoning, particularly in domains that are more likely to reflect the data distribution of modern LLMs, as they often feature more recent and topical content more representative of typical pretraining corpora.

Model Diversity and Benchmark Expansion We only explored how the selected models (Llama-3.3-70B-Instruct, Llama-3_3-Nemotron-Super-49B-v1 and Qwen3-32B) behave on a subset of 30 documents from our dataset for our framework. Due to time and computational constraints, we did not extend this analysis to the full CoFluEval-LC dataset. Moreover, we did not include commercial models such as GPT-4 (OpenAI et al., 2024) or Claude 3 (Anthropic, 2024), which have shown superior performance over open-source models in long-context tasks according to recent studies (Liu et al., 2024b; Modarressi et al., 2025a). These models often support larger context windows (e.g., up to 200k tokens) and may have better architectural or training adaptations for handling long sequences.

A natural extension of this work would be to build a broader benchmark based on CoFluEval-LC, adapted for cross-model evaluation. This would allow for a more comprehensive comparison across open-source and proprietary models and offer insight into how prompt sensitivity, memory mechanisms, and evaluation robustness scale across architectures.

Context Length and Manipulation Diversity With the increasing availability of LLMs that support extended context windows (e.g., GPT-4 Turbo, Claude 3.5), another future direction is to extend the context length of documents in CoFluEval-LC. This would allow researchers to examine how evaluation performance scales with even longer input documents (e.g., 8K, 32K, or 100K tokens), and how various strategies, such as chunking, summarization, or retrieval-based augmentation, interact with these extended contexts.

From our experimental analysis in Chapter 5, we also observed that not all manipulations were equally challenging to detect. Some manipulation types may have introduced too strong or too obvious disruptions, while others were too subtle. Furthermore, several manipulation types, such as awkward phrasing, passive voice misuse, or redundancy, were excluded from this work for scope reasons but offer valuable potential for increasing the challenge and realism of future benchmarks. Incorporating a more diverse and linguistically nuanced set of manipulations would help stress-test LLM evaluation fidelity across a broader range of issues.

Memory-Based Evaluation Limitations While our proposed method yields clear improvements over standard long-context evaluation setups, several important limitations must be acknowledged. First, JudgeMemo relies on segmentation and summarization heuristics, which may introduce noise or bias into the memory representation. The quality of the intermediate notes directly affects the final evaluation, and imperfect segment boundaries or overly generic summaries could distort the assessment process. Additionally, while our manipulations are rule-based and interpretable, they do not capture the full complexity of real-world degradation in generated text, such as factual inconsistencies, subtle contradictions, or stylistic mismatches. These more naturalistic forms of degradation often arise in generative settings and require deeper semantic understanding to detect.

Furthermore, we did not evaluate our method on other benchmarks beyond our primary setup. This decision was due to two main factors: time constraints, and the significant adaptation effort required to apply our approach to new tasks. In particular, each benchmark would demand task-specific prompt engineering and careful design choices to align with its format and evaluation criteria; an effort that is nontrivial and time-consuming and therefore, out of our scope for this work. We leave a broader exploration of these applications to future work.

References

AI@Meta. 2024. Llama 3.3-70b model card.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2024. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14388–14411, Bangkok, Thailand. Association for Computational Linguistics.

Anthropic. 2024. Claude 3 model card. Accessed: 2024-07-18.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient training of language models to fill in the middle.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, and et al. 2025. Llama-nemotron: Efficient reasoning models.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv:2006.14799*.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Hoa Trang Dang. 2006. DUC 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, pages 48–55, Sydney, Australia. Association for Computational Linguistics.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Mohsen Fayyaz, Ali Modarressi, Hinrich Schuetze, and Nanyun Peng. 2025. Collapse of dense retrievers: Short, early, and literal biases outranking factual evidence.

Mingqi Gao, Jie Ruan, Renliang Sun, Xunjian Yin, Shiping Yang, and Xiaojun Wan. 2023. Human-like summarization evaluation with chatgpt. *arxiv:2304.02554*.

Gemini Team et al. 2025. Gemini: A family of highly capable multimodal models.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, and et al. 2024. The llama 3 herd of models.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2024. A survey on llm-as-a-judge. *preprint arXiv:2411.15594*.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Xinyu Hu, Mingqi Gao, Sen Hu, Yang Zhang, Yicheng Chen, Teng Xu, and Xiaojun Wan. 2024. Are LLM-based evaluators confusing NLG quality criteria? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9530–9570, Bangkok, Thailand. Association for Computational Linguistics.

Sameer Jain, Vaishakh Keshava, Swarnashree Mysore Sathyendra, Patrick Fernandes, Pengfei Liu, Graham Neubig, and Chunting Zhou. 2023. Multi-dimensional evaluation of text summarization with in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8487–8495, Toronto, Canada. Association for Computational Linguistics.

Pei Ke, Hao Zhou, Yankai Lin, Peng Li, Jie Zhou, Xiaoyan Zhu, and Minlie Huang. 2022. CTRL-Eval: An unsupervised reference-free metric for evaluating controlled text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2319, Dublin, Ireland. Association for Computational Linguistics.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Wojciech Kryscinski, Nazneen Fatema Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir R. Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization. *CoRR*, abs/2105.08209.

Alice Lai and Joel Tetreault. 2018. Discourse coherence in the wild: A dataset, evaluation and methods. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 214–223, Melbourne, Australia. Association for Computational Linguistics.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, Bangkok, Thailand. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. 2024a. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2024b. LooGLE: Can long-context language models understand long contexts? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16304–16333, Bangkok, Thailand. Association for Computational Linguistics.

Yen-Ting Lin and Yun-Nung Chen. 2023. LLM-Eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. In *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pages 47–58, Toronto, Canada. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Xiang Liu, Peijie Dong, Xuming Hu, and Xiaowen Chu. 2024b. LongGenBench: Long-context generation benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 865–883, Miami, Florida, USA. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Yixin Liu, Kejian Shi, Katherine He, Longtian Ye, Alexander Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. 2024c. On learning to summarize with large language models as references. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8647–8664, Mexico City, Mexico. Association for Computational Linguistics.

Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2024d. HD-eval: Aligning large language model evaluators through hierarchical criteria decomposition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7641–7660, Bangkok, Thailand. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Manu. 2022. Project gutenberg dataset. `https://huggingface.co/datasets/manu/project_gutenberg`.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. 2025a. Nolima: Long-context evaluation beyond literal matching.

Ali Modarressi, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2025b. Memllm: Finetuning llms to use an explicit read-write memory.

Peter Norvig. 2012. English letter frequency counts: Mayzner revisited or etaoin srhldcu. `https://norvig.com/mayzner.html`. Accessed: 2025-07-12.

OpenAI et al. 2024. Gpt-4 technical report.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models.

Project Gutenberg. 1971. Project Gutenberg. `https://www.gutenberg.org`.

StudySmarter. 2024. Coherence assessment. Online; accessed on 2025-03-14. Accessed: 2025-07-12.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. 2023. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14590–14604, Toronto, Canada. Association for Computational Linguistics.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang

Wang, Raluca Ada Popa, and Ion Stoica. 2024. Judgebench: A benchmark for evaluating llm-based judges.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2025. Scm: Enhancing large language model with self-controlled memory framework.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Wei Wang and Qing Li. 2024. Schrodinger's memory: Large language models.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. 2024. Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. SummIt: Iterative text summarization via ChatGPT. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10644–10657, Singapore. Association for Computational Linguistics.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731.

List of Figures

List of Tables

List of Prompts

A Supplementary Material

A.1 General Reference Materials

Label	Description
PERSON	People, including fictional
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
FAC	Buildings, airports, highways, bridges, etc.

Table A.1: Overview of the spaCy NER labels used in this work to detect named entities for document-level manipulations

taken from spaCy's GitHub Repository

Abbreviation	Full Model Name
LN-noT	Llama-3_3-Nemotron-Super-49B-v1-noThink
LN-T	Llama-3_3-Nemotron-Super-49B-v1-Think
Q-noT	Qwen3-32B-noThink
Q-T	Qwen3-32B-Think
LI	Llama-3.3-70B-Instruct

Table A.2: Abbreviations of model names as used throughout this thesis for improved clarity and simplicity in tables.

$\textbf{Mani}_{\textbf{ID}}$	**Manipulation Type**	**Short Description**
[11]	Swap Content	swapping text passages within the document
[12]	Remove Content	removing text passages from the document
[13]	Insert Content	inserting text passages from an external source in the document
[14]	Repeat Content	repeating text passages within the document
[17]	Exchange Content	replacing text passages in the document with external text passages
[21]	Swap Entities	cyclic swap of different occuring entity mentions within the text
[22]	Exchange Entities	replacement of entity mentions with external entities (not part of the text)
[24]	Temporal Inconsistencies	adding neutral phrases introducing concepts from the 21^{st} century
[31]	Entity-to-Term Pronoun	replacing entity mentions with adequate pronouns
[32]	Entity-to-Term Replacement	replacing entity mentions with neutral terms like “stuff” or “thing”
[41]	Typos	introducing spelling errors
[42]	Incorrect Verb Tenses	changing verbs and auxilary verbs from Past to Present and vice versa
[43]	Shuffeling Word Order	random ordering of words within a sentence
[44]	Punctuation Removal	removing punctuation characters from the text
[45]	Word Removal	removing words from the text

Table A.3: Overview of all implemented text manipulation types. Bolded IDs indicate the subset of manipulations that are currently included in the CoFluEval-LC dataset.

A.2 Gold Subset Statistics

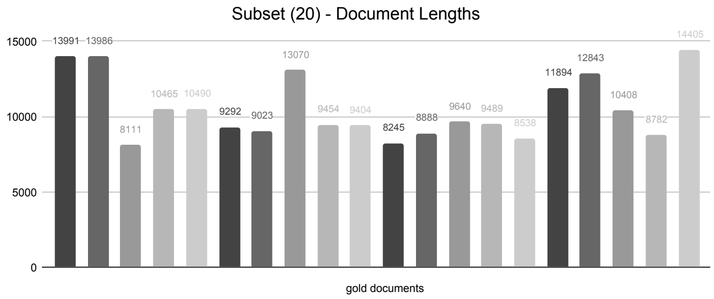


Figure A.1: Token distribution across the 20 gold documents selected for manipulation effectiveness tests and sanity checks. Token counts are based on whitespace tokenization.

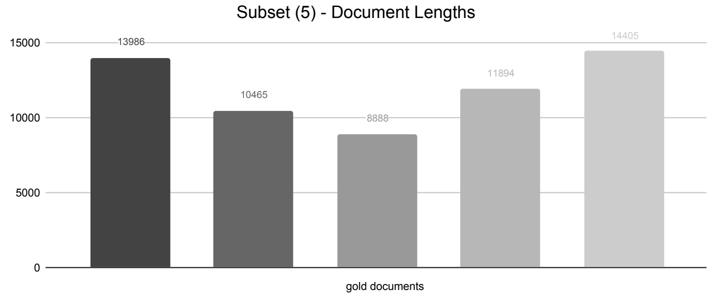


Figure A.2: Token distribution across the 5 gold documents selected for prompt engineering methodolody. Token counts are based on whitespace tokenization.

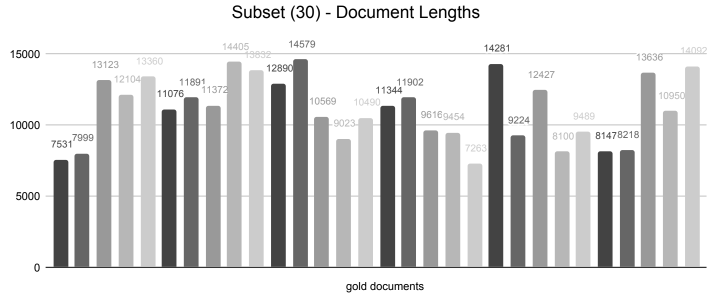


Figure A.3: Token distribution across the 30 gold documents selected for JUDGEMEMO framework experiments and ablation studies. Token counts are based on whitespace tokenization.

A.3 Additional Implemented Text Manipulations

This section first presents explanations for additional manipulations we implemented but deselected in the early stages of this work. It further provides illustrative examples of the manipulation types introduced in Section 3.3, helping to visualize how each manipulation alters the original text.

Additionally, we provide the full taxonomy of all manipulation types that were implemented and tested during the course of this thesis. Note that only manipulations [41], [42], [43], [17], [24], and [32] are currently included in CoFluEval-LC.

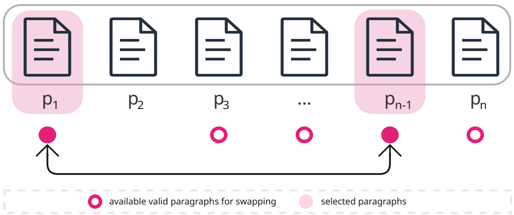


Figure A.4: Illustration of the paragraph-level swapping manipulation for $n_o = 1$

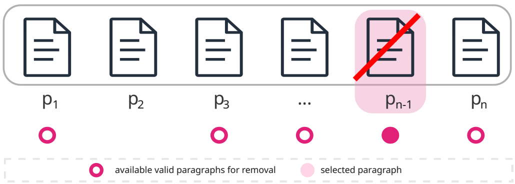


Figure A.5: Illustration of the paragraph-level removal manipulation for $n_o = 1$

Insert Content [13] To simulate the introduction of off-topic or contextually irrelevant material, we implement the *content insertion operation*. This manipulation assesses whether an LLM can identify and react to disruptions in thematic consistency and narrative flow caused by the sudden inclusion of unrelated or out-of-place segments.

Formally, let $d_{\text{insert}} = [q_1, q_2, \ldots, q_m]$ be a separate source of candidate paragraphs for insertion. Each q_j denotes a paragraph as defined above.

We identify a subset $\mathcal{J} \subset 1, \ldots, m$ of candidate insertions satisfying a minimum length criterion (e.g., 50 characters), and ...randomly sample n_o distinct indices $j_1, \ldots, j_{n_o} \subset \mathcal{J}$ without replacement, provided that $|\mathcal{J}| \geq n_o$. If there are insufficient valid insertion candidates, the operation is aborted.

Similarly, we sample n_o insertion points $k_1, \ldots, k_{n_o} \subset 0, \ldots, n$ in the original document $d^{(i)}_{\text{gold}}$, where each k_w denotes the index after which the insertion occurs.

We highlight that inserted segments originate from an external document and are intentionally not semantically aligned with their surrounding context, thereby introducing potential coherence-breaking elements. Also note that this operation leads to an increase in the total number of tokens in the document.

The manipulated document is then defined as:

$$d^{(i)}_{C,\text{insert}} = m_{C,\,\text{insert}}(d^{(i)}_{\text{gold}}) = [p_1, \ldots, p_{k_1}, q_{j_1}, p_{k_1+1}, \ldots, p_{k_{n_o}}, q_{j_{n_o}}, p_{k_{n_o}+1}, \ldots, p_n] \tag{A.1}$$

Given $n_o = 1$, we would insert one paragraph taken from d_{insert} after a randomly chosen paragraph in the gold document $d^{(i)}_{\text{gold}}$ (cf. Figure A.6).

The resulting paragraph sequence is reassembled into a single textual body, preserving formatting via the original newline delimiters. Each insertion point is represented as a character-level position immediately following a selected paragraph in the original document. The inserted span begins and ends at this same position, reflecting a pure addition.

Repeat Content [14] We apply the *content repetition operation* to assess how sensitive LLMs are to internal redundancies and textual repetition. This manipulation duplicates selected seg-

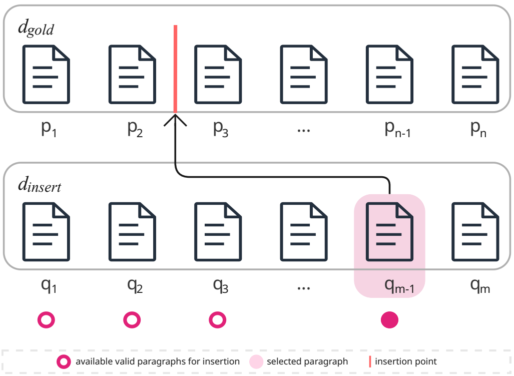


Figure A.6: Illustration of the paragraph-level insertion manipulation for $n_o = 1$

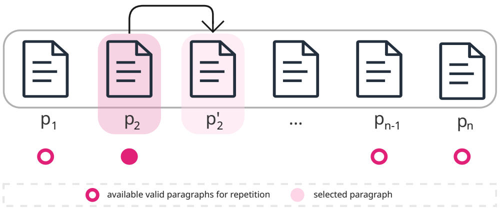


Figure A.7: Illustration of the paragraph-level repetition manipulation for $n_o = 1$ and $f_{\text{repeat}} = 1$

ments of text directly after themselves, introducing redundancy without modifying the document's semantic content.

Unlike insertions or exchanges, this operation does not incorporate external information but instead amplifies internal content, potentially leading to local coherence issues or inflated emphasis.

The number of repetitions is governed by the repetition factor $f_{\text{repeat}} \geq 1$, allowing fine-grained control over the degree of redundancy introduced. Figure A.7 shows an example of paragraph-level repetition with a repetition factor $f_{\text{repeat}} = 1$.

We begin by identifying a subset of valid paragraph indices $\mathcal{I} \subset \{1, \dots, n\}$ based on a minimum length requirement (e.g., 50 characters). We uniformly select n_o distinct paragraph indices $\{j_1, \dots, j_{n_o}\}$ from the valid set $\mathcal{I}$, where each j_k denotes the index of a paragraph chosen for repetition, with $k = 1, \dots, n_o$. Each selected paragraph p_{j_k} is duplicated f_{repeat} times directly after its original position, yielding:

$$d^{(i)}_{C,\text{ repeat}} = m_{C,\text{ repeat}}(d^{(i)}_{\text{gold}}) = [p_1, \dots, p_{j_k}, \underbrace{p_{j_k}, \dots, p_{j_k}}_{f_{\text{repeat}}\text{ times}}, p_{j_k+1}, \dots, p_n] \tag{A.2}$$

This operation is applied independently for each of the n_o selected paragraphs, with insertions tracked and performed in order of increasing paragraph indices to preserve correct offset handling.

The resulting paragraph sequence is reassembled using preserved delimiters between paragraphs.

Entity-to-Pronoun Replacement [31] By implementing the *entity-to-pronoun replacement* manipulation, we again substitute named entities in the text, but this time, we use pronouns instead of replacement terms to obscure reference chains and weaken long-range coherence. Specifically, selected entity mentions are substituted with contextually appropriate pronouns, depending on their position in the sentence. These types are chosen because they are typically referred to with the pronoun *it* in English, which ensures grammatical consistency and avoids complications associated with gendered or plural pronoun resolution.

While the replacements preserve surface fluency, they remove entity specificity, making it more difficult to resolve coreference and maintain global coherence.

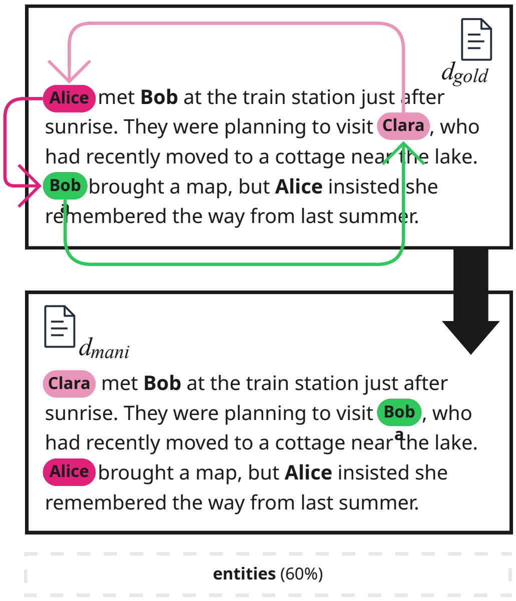
Formally, given Definitions 3.5 and 3.7, we sample a replacement proportion $r \in [start, end]$ and compute the number of entities to replace n_o as specified in Definition 3.15.

In the next step, a subset $\mathcal{E}_{\text{sub}} \subset \mathcal{E}$ of size n_o is sampled uniformly at random. Each selected entity

$ent_j \in \mathcal{E}_{\text{sub}}$ is then replaced with the pronoun *it* or *It*, depending on sentence position, to keep the manipulation subtle and avoid introducing confounding factors such as typos.

$$d^{(i)}_{C,\text{ ent-pronoun}} = m_{C,\text{ ent-pronoun}}(d^{(i)}_{\text{gold}}) = \texttt{replace}(ent_j \mapsto pron_j) \quad \forall ent_j \in \mathcal{E}_{\text{sub}} \tag{A.3}$$

We illustrate a simplified example in Figure A.9. Detected entities are marked as bold. In this example, we select $n_o = 3$ entities of $\mathcal{E}_{\text{sub}} = \{GrandCentralStation, NewYorkCity, OpenAI\}$ (highlighted) and exchange them with the pronoun *it*.



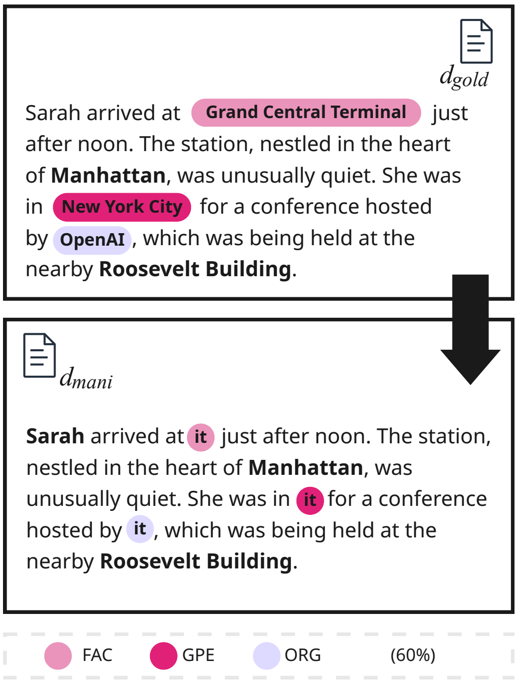
d_{gold}

Alice met **Bob** at the train station just after sunrise. They were planning to visit **Clara**, who had recently moved to a cottage near the lake. **Bob** brought a map, but **Alice** insisted she remembered the way from last summer.

d_{mani}

Clara met **Bob** at the train station just after sunrise. They were planning to visit **Bob**, who had recently moved to a cottage near the lake. **Alice** brought a map, but **Alice** insisted she remembered the way from last summer.

Figure A.8: Illustration of a cyclic swap between named entities of type PERSON for $n_o = 3$



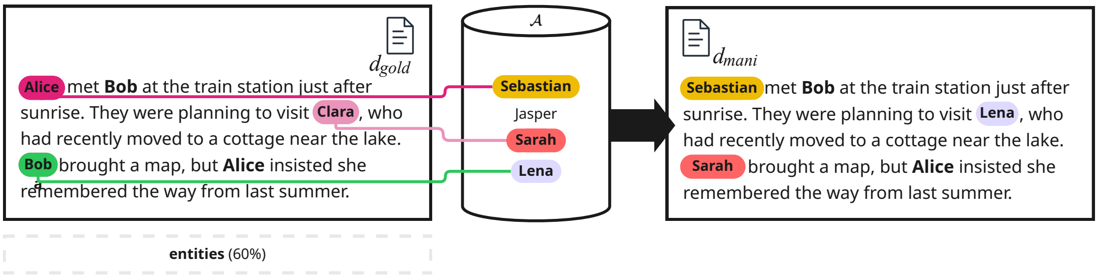
d_{gold}

Sarah arrived at **Grand Central Terminal** just after noon. The station, nestled in the heart of **Manhattan**, was unusually quiet. She was in **New York City** for a conference hosted by **OpenAI**, which was being held at the nearby **Roosevelt Building**.

d_{mani}

Sarah arrived at **it** just after noon. The station, nestled in the heart of **Manhattan**, was unusually quiet. She was in **it** for a conference hosted by **it**, which was being held at the nearby **Roosevelt Building**.

Figure A.9: Illustration of named entity replacement with pronouns for $n_o = 3$



d_{gold}

Alice met **Bob** at the train station just after sunrise. They were planning to visit **Clara**, who had recently moved to a cottage near the lake. **Bob** brought a map, but **Alice** insisted she remembered the way from last summer.

d_{mani}

Sebastian met **Bob** at the train station just after sunrise. They were planning to visit **Lena**, who had recently moved to a cottage near the lake. **Sarah** brought a map, but **Alice** insisted she remembered the way from last summer.

Figure A.10: Illustration of replacing known entities of type PERSON with selected names from a pool of alternative entities $\mathcal{A}$ for $n_o = 3$

Punctuation Removal [44] The *punctuation removal* operation aims to affect the fluency of a document by selectively eliminating punctuation characters, which serve as critical cues for sentence structure and meaning.

Formally, let $d_{\text{gold}}^{(i)}$ be defined as in Definition 3.5. Given the manipulation intensity parameter $r \in [start, end]$ for the ratio-based setting, the number of punctuation characters to be removed, n_o, is computed as:

$$n_o = \text{round}\left(r \cdot \frac{|\mathcal{P}|}{100}\right) \tag{A.4}$$

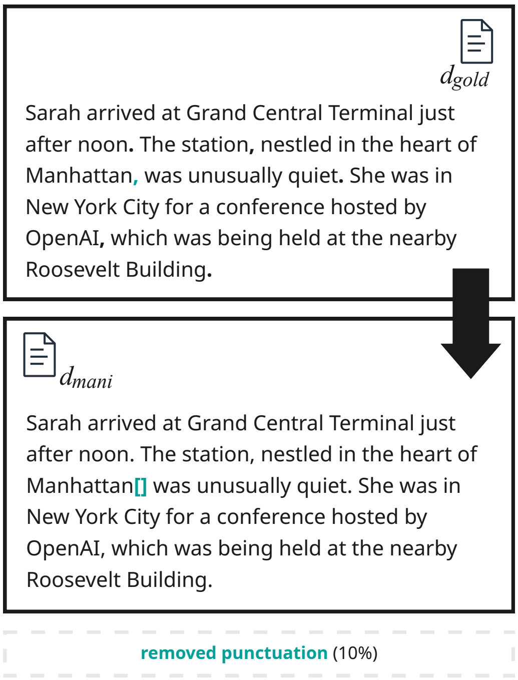
with $\mathcal{P} \subseteq \{char_j \mid char_j \in \texttt{string.punctuation}\}$ denoting the sequence of punctuation characters[19] present in $d_{\text{gold}}^{(i)}$.

The manipulation function $m_{F,\,\text{punct-removal}}(d_{\text{gold}}^{(i)})$ randomly selects n_o punctuation characters and removes them to create the manipulated document $d_{\text{mani}}^{\text{punct-removal}}$, expressed formally as:

$$d_{F,\,\text{punct-removal}}^{(i)} = m_{F,\,\text{punct-removal}}(d_{\text{gold}}^{(i)}) = \texttt{remove}(char_j), \quad \forall char_j \in \mathcal{P}_{sub} \tag{A.5}$$

where $\mathcal{P}_{sub} \subseteq \mathcal{P}$ is the subset of punctuation characters chosen for removal.

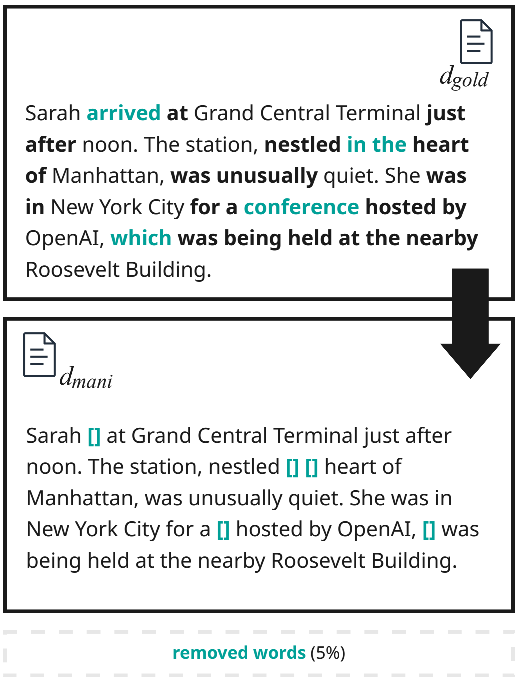
This operation reduces syntactic clarity and fluency by eliminating punctuation marks while keeping lexical content intact. Although primarily affecting fluency, it may also have secondary effects on coherence by obscuring sentence and clause boundaries. See Figure A.11 for an illustration of the manipulation's effect.



Sarah arrived at Grand Central Terminal just after noon. The station, nestled in the heart of Manhattan, was unusually quiet. She was in New York City for a conference hosted by OpenAI, which was being held at the nearby Roosevelt Building.

Sarah arrived at Grand Central Terminal just after noon. The station, nestled in the heart of Manhattan[] was unusually quiet. She was in New York City for a conference hosted by OpenAI, which was being held at the nearby Roosevelt Building.

Figure A.11: Illustration of punctuation removal manipulation for $n_o = 3$



Sarah **arrived** **at** Grand Central Terminal **just** **after** noon. The station, **nestled** **in the** **heart** **of** Manhattan, **was unusually** quiet. She **was** **in** New York City **for a** **conference** **hosted by** OpenAI, **which** **was being held at the nearby** Roosevelt Building.

Sarah [] at Grand Central Terminal just after noon. The station, nestled [] [] heart of Manhattan, was unusually quiet. She was in New York City for a [] hosted by OpenAI, [] was being held at the nearby Roosevelt Building.

Figure A.12: Illustration of word removal manipulation for $n_o = 3$

Word Removal [45] *Word Removal* manipulations reduce the lexical content of a document by randomly deleting a specified number of words composed exclusively of alphanumeric characters.

This operation can significantly impact both the fluency and the semantic coherence of the text, as the removal of words breaks the continuity of information and readability flow and may create gaps in meaning.

[19] We use the `string.punctuation` module as a reference here.

Let $d_{\text{gold}}^{(i)}$ be defined as a sequence of tokens (cf. Equation 3.5). Given the manipulation intensity parameter $r \in [start, end]$ for the ratio-based setting, the number of words to remove, denoted n_o, is computed as in Definition 3.2.

The manipulation function $m_{F,\text{word-removal}}(d_{\text{gold}}^{(i)})$ removes n_o randomly selected tokens in $d_{\text{gold}}^{(i)}$ to produce the manipulated document $d_{F,\text{ word-removal}}^{(i)}$, expressed as:

$$d_{F,\text{ word-removal}}^{(i)} = m_{F,\text{ word-removal}}(d_{\text{gold}}^{(i)}) = \texttt{remove}(tok_j), \quad \forall tok_j \in \mathcal{I}_{sub} \tag{A.6}$$

where $\mathcal{I}_{sub} \subseteq d_{\text{gold}}^{(i)}$ is the subset of tokens removed.

We give a high-level example for this manipulation in Figure A.12.

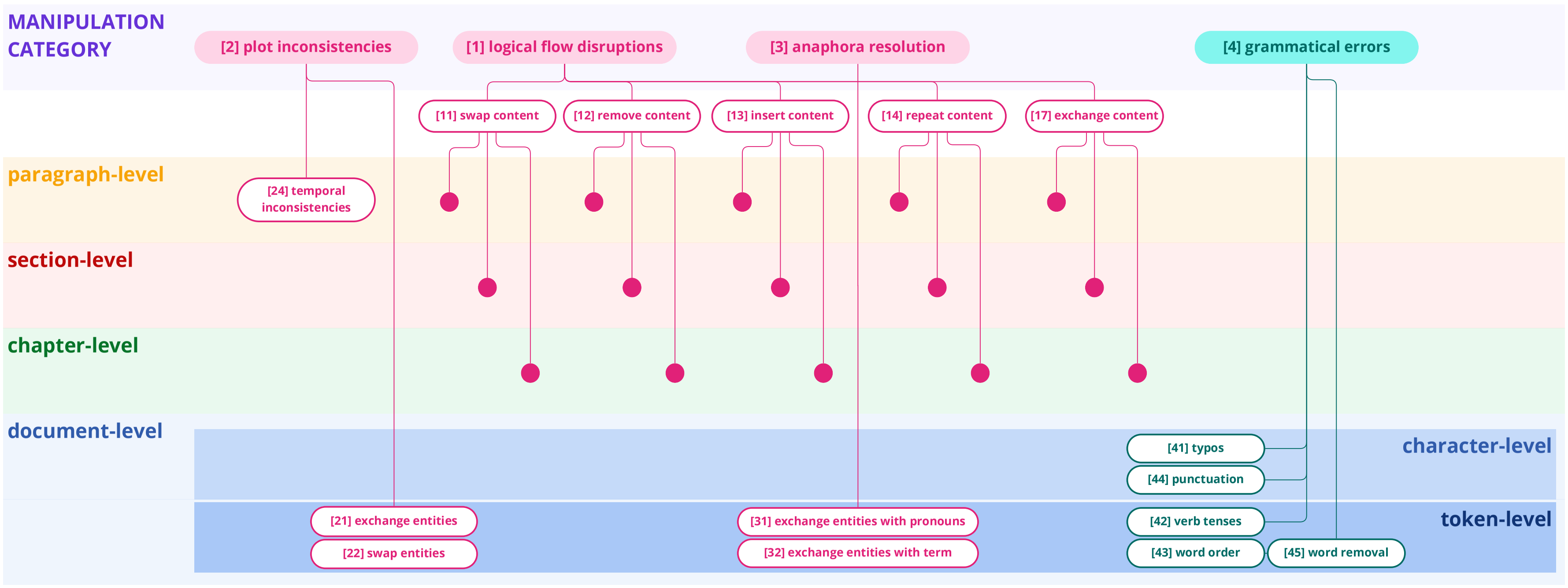


Figure A.13: Overview of the implemented text manipulations, categorized by structural scope and level of granularity. Manipulations targeting coherence are highlighted in pink, while those affecting fluency are highlighted in green.

A.4 Effectiveness of Manipulations

We give give some insights for the initial screening and filtering process of manipulations in this section. Further, we provide the full set of heatmaps from testing referenced in Section 3.4, illustrating the effects of each tested manipulation. The results are organized by whether the manipulation was selected for the final dataset or excluded during earlier stages of evaluation.

Initial Screening and Filtering

We apply all manipulations at a single, reasonably chosen intensity across the selected documents. These intensities were selected heuristically to enable a broad initial screening. Based on these tests, we narrow down the set of manipulations for further study. At this stage, we also conduct manual inspections of manipulated documents and their corresponding model-generated scores to better understand the effects of each manipulation and how they influence model behavior.

We initially apply all suitable coherence manipulations across all structural levels (paragraph, section, chapter) to evaluate their effects. However, chapter-level manipulations are impractical, as many gold documents lack a reliable chapter structure (Section 3.1.4), which would exclude a substantial portion of the dataset. Similarly, section-level manipulations are unsuitable for our setup: in our implementation, sections span at least 3000 characters ($\approx$ 500 to 600 tokens; cf. Section 3.2.3). In shorter documents (1000–3000 tokens), this would affect up to half the content, making the manipulation too obvious and thus compromising subtlety. Consequently, manipulations involving structural granularity will be applied exclusively at the paragraph level.

Several coherence manipulations were excluded at this stage due to practical limitations. *Insert Content* and *Repeat Content* proved too disruptive or inconsistent across document lengths, compromising subtlety and comparability.

For fluency-focused manipulations, we first considered *Punctuation Removal*, but due to the sparsity of punctuation marks, its impact was limited. Instead, we opted for inserting more substantial typographic errors. We also tested *Word Removal*, which slightly reduces document length. However, to maintain consistent length across conditions, we preferred *Word Order Shuffling*, which preserves length while still affecting fluency.

Mani$_{\text{ID}}$	**structural dimension**	**Metric**	$\overline{\textbf{Score}}_{\textbf{1K}}$	$\overline{\textbf{Score}}_{\textbf{2K}}$	$\Delta_{\textbf{1K}}$	$\Delta_{\textbf{2K}}$
[13]	paragraph	Fluency	4.650	4.500	-0.050	-0.100
		Coherence	4.650	4.750	-0.200	-0.100
	section	Fluency	4.600	4.450	-0.100	-0.150
		Coherence	4.350	4.600	-0.500	-0.250
	chapter	Fluency	4.600	4.650	-0.100	+0.050
		Coherence	3.800	4.050	-1.050	-0.800
[14]	paragraph	Fluency	4.600	4.700	-0.100	+0.100
		Coherence	4.800	4.750	-0.050	-0.100
	section	Fluency	4.550	4.700	-0.150	+0.100
		Coherence	4.550	4.900	-0.300	+0.050
	chapter	Fluency	4.700	4.750	0.000	+0.150
		Coherence	4.350	4.350	-0.500	-0.500

Table A.4: Initial test results for deselected manipulations with $n_o = 2$ (count-based) and $f_{\text{repeat}} = 1$. Δ-values are computed with respect to the corresponding gold score in Table 3.4.

Mani$_{\text{ID}}$	structural dimension	Metric	$\overline{\text{Score}}_{1\text{K}}$	$\overline{\text{Score}}_{2\text{K}}$	$\Delta_{1\text{K}}$	$\Delta_{2\text{K}}$
[44]	document	Fluency	4.700	4.450	0.000	-0.150
		Coherence	4.850	4.750	+0.050	0.000
[45]	document	Fluency	4.150	4.000	-0.550	-0.600
		Coherence	4.800	4.650	-0.050	-0.200

Table A.5: Initial test results for deselected manipulations with $n_o = 50$ (count-based) for word removal and $n_o \in [10; 20]$ for punctuation removal. n_o is doubled in `2K` setting for both manipulations. Δ-values are computed with respect to the corresponding gold score in Table 3.4.

As a result of these limitations, we excluded the above manipulations from further experiments. Initial results are reported in Table A.4 and Table A.5.

Heatmaps for Selected Manipulations

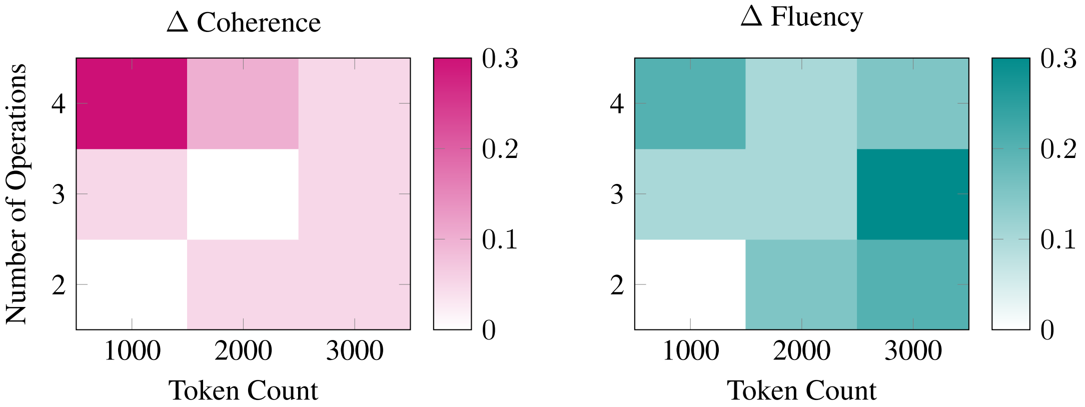


Figure A.14: Heatmaps for **Temporal Inconsistencies** [24]: Count-based analysis at paragraph-level showing the absolute change (Δ) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and number of operations.

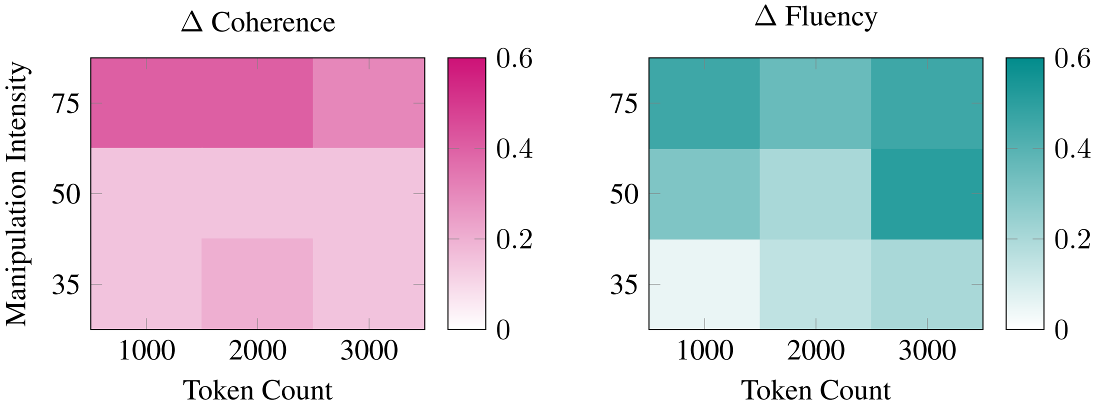


Figure A.15: Heatmaps for **Entity-to-Term Replacement** [32]: Ratio-based analysis at the paragraph level showing the absolute change (∆) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and manipulation intensities (in percent, %).

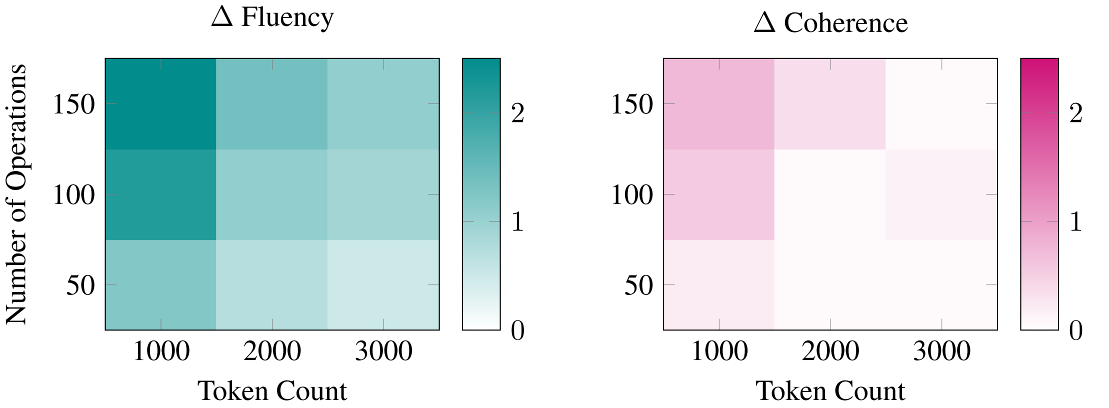


Figure A.16: Heatmaps for **Incorrect Verb Tenses** [42]: Count-based analysis at paragraph-level showing the absolute change (∆) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and number of operations.

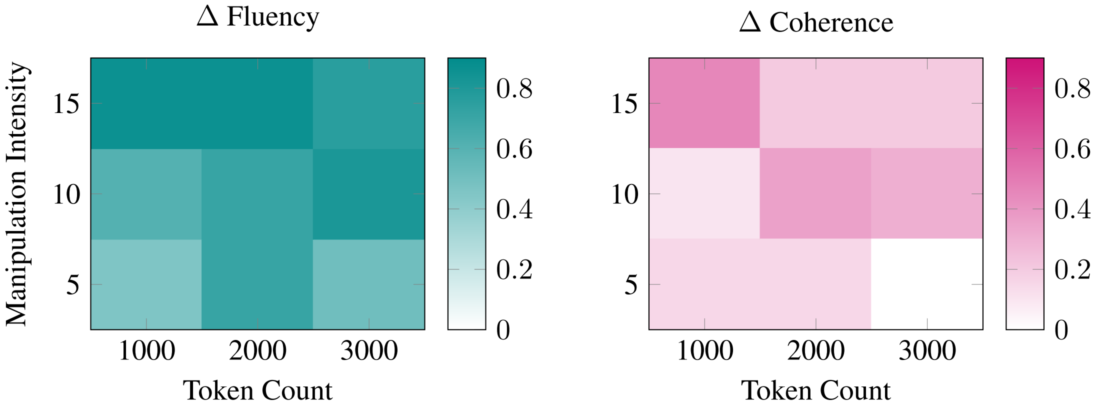


Figure A.17: Heatmaps for **Shuffling Word Order** [43]: Ratio-based analysis at the paragraph level showing the absolute change (∆) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and manipulation intensities (in percent, %).

Heatmaps and for Deselected Manipulations

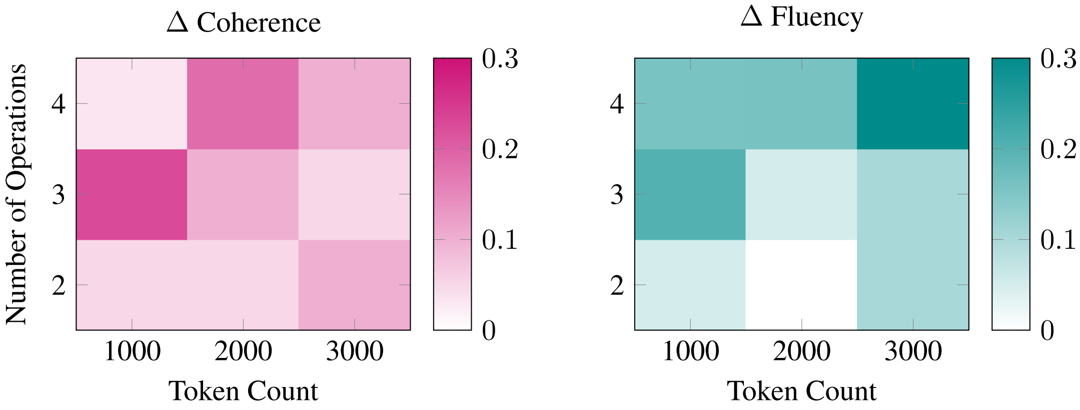


Figure A.18: Heatmaps for **Swap Content** [11]: Count-based analysis at paragraph-level showing the absolute change (Δ) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and number of operations.

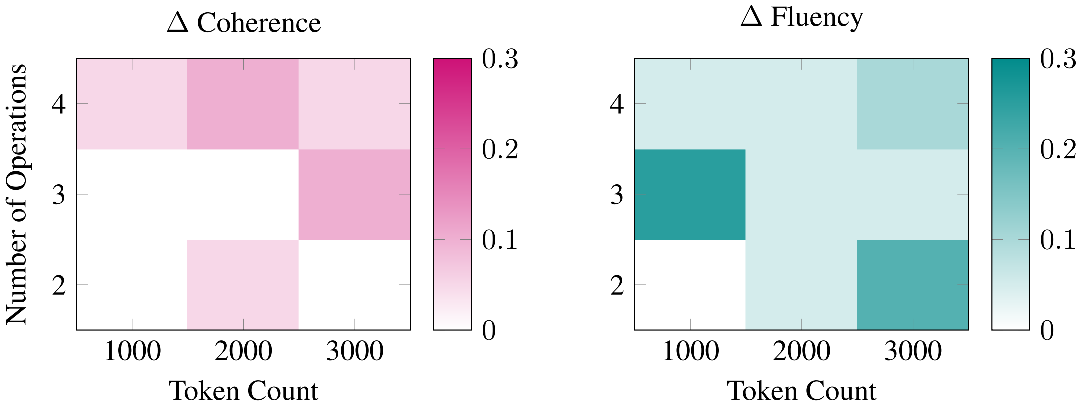


Figure A.19: Heatmaps for **Remove Content** [12]: Count-based analysis at paragraph-level showing the absolute change (Δ) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and number of operations.

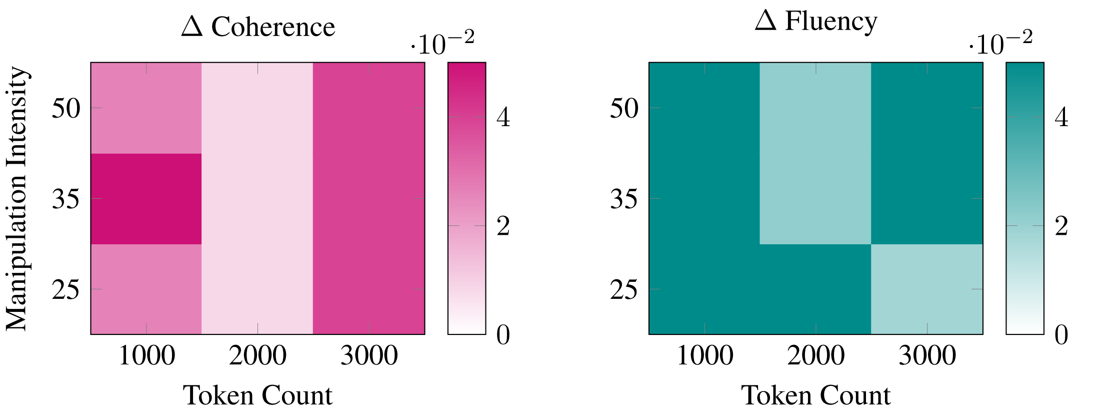


Figure A.20: Heatmaps for **Swap Entities** [21]: Ratio-based analysis at the paragraph level showing the absolute change (Δ) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and manipulation intensities (in percent, %).

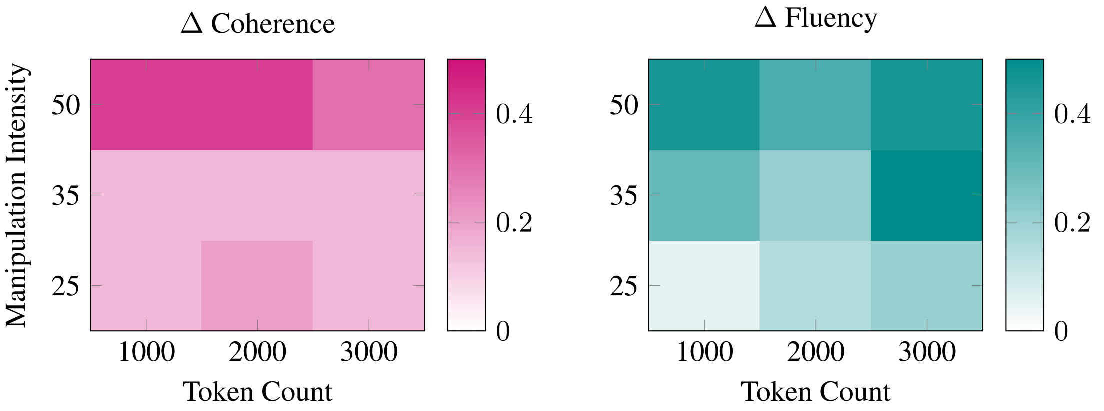


Figure A.21: Heatmaps for **Exchange Entities** [22]: Ratio-based analysis at the paragraph level showing the absolute change (Δ) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and manipulation intensities (in percent, %).

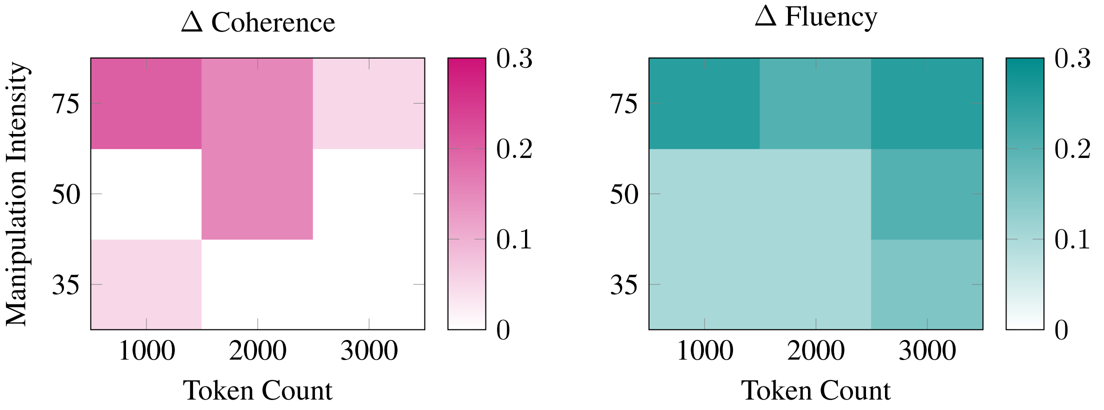
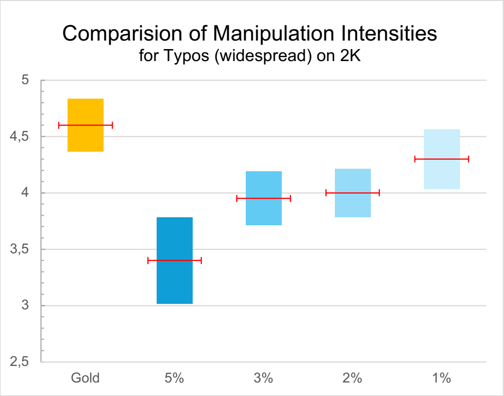
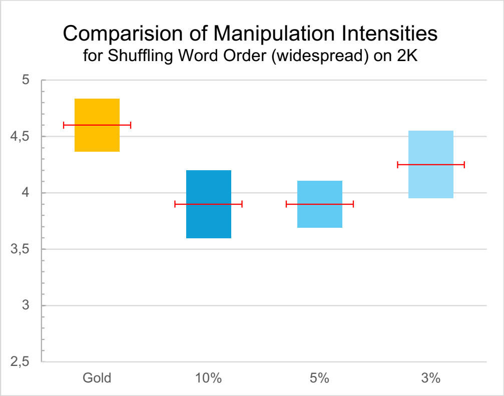


Figure A.22: Heatmaps for **Entity-to-Pronoun Replacement** [31]: Ratio-based analysis at the paragraph level showing the absolute change (Δ) in fluency and coherence scores relative to the corresponding gold document scores (cf. Table 3.4). Results are displayed across document lengths and manipulation intensities (in percent, %).

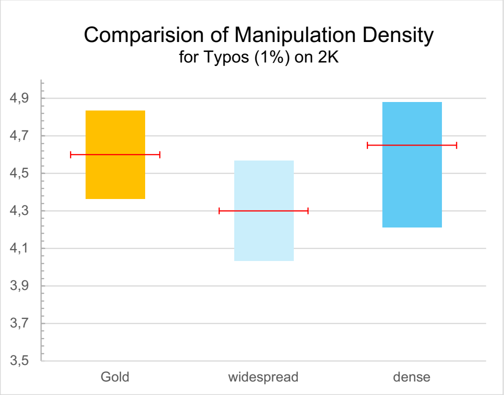
Validation and Sanity Check Results



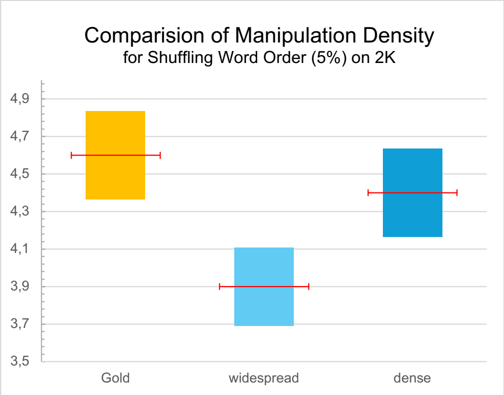
(a) Effect of varying *typo* manipulation intensities on LLM evaluations. Results are benchmarked against the gold scores for the respective documents.



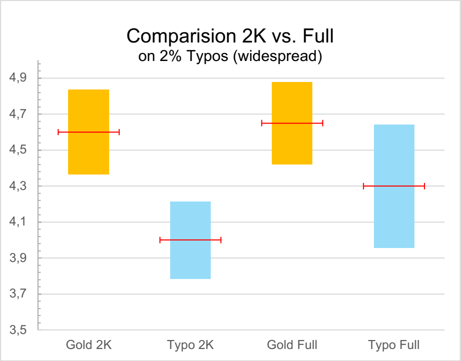
(b) Effect of varying *shuffeling word order* manipulation intensities on LLM evaluations. Results are benchmarked against the gold scores for the respective documents.



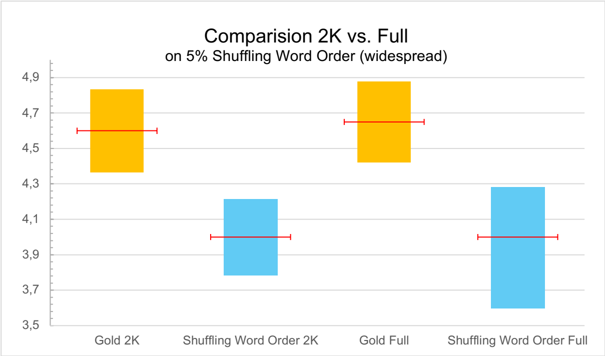
(a) Effect of varying *typo* manipulation densities on LLM evaluations for a fixed intensity. Results are benchmarked against the gold scores for the respective documents.



(b) Effect of varying *shuffeling word order* manipulation densities on LLM evaluations for a fixed intensity. Results are benchmarked against the gold scores for the respective documents.



(a) Comparision of confidence intervals for the `2K` and the `Full` setting for *typos*.



(b) Comparision of confidence intervals for the `2K` and the `Full` setting for *shuffeling word order*.

A.5 Implementation Details

Python Version and Packages Both, the vanilla baseline and the JUDGEMEMO pipeline, are implemented in Python 3.11.11[20] using the vLLM library[21] for efficient interaction with large language models.

Models We utilize several pretrained LLMs, including Llama-3.3-70B-Instruct (Llama-Instruct), Llama-3_3-Nemotron-Super-49B-v1 (Llama-Nemotron), and Qwen3-32B, which are loaded via the Huggingface Hub.[22] For inference, we follow the configuration recommendations for each model as summarized in Table A.6.

Parameter	Llama-Instruct	Llama-Nemotron	Qwen3-32B
max_tokens	1024	1024 / 2048*	1024 / 2048*
temperature	0.0	0.0 / 0.6*	0.7 / 0.6*
top_p	1.0	1.0 / 0.95*	1.0 / 0.95*
quantization	fp8	fp8	fp8
other	—	—	top_k: 20

Table A.6: Model Generation Configurations

*Values after the slash apply in reasoning mode.

Computational Resources All experiments and model evaluations in this thesis were conducted on high-performance computing (HPC) clusters using Slurm as the job scheduler. Specifically, we used nodes equipped with NVIDIA H100 and A100 GPUs. For all runs, we set `n_gpus = 2`, allowing vLLM to distribute model weights across two GPUs via tensor parallelism.

We provide detailed instructions on accessing our data, scripts, and results in Appendix C.

A.6 Prompt Engineering Findings

This section provides additional insights from the prompt engineering process described in Section 4.1, including an analysis of generation issues for each model, as well as plots showing the average gold scores and average manipulation deltas across all tested models.

[20] See Python 3.11.11 on python.org

[21] See vLLM Documentation: https://docs.vllm.ai/en/latest/

[22] See Huggingface Hub Documentation: https://huggingface.co/docs/hub/index

	Generation Issues Gold				Generation Issues Manipulated			
Prompt	2000 F	Full C	2000 F	Full C	2000 F	Full C	2000 F	Full C
v1	0	0	0	0	2	2	1	2
v2	0	0	0	0	1	1	1	1
v3	0	0	0	0	0	0	3	3
v4-2	0	0	0	0	1	1	1	1
v5-1	0	0	0	0	0	0	1	1
v5-2	0	0	0	0	0	0	5	6
v5-3	0	0	0	0	1	1	5	5
v5-4	0	0	0	0	0	0	3	3
v5-5	0	0	0	0	1	1	4	4
v5-6	0	0	0	0	0	0	2	2
v5-7	0	0	0	0	1	1	2	2
v5-8	1	1	0	0	4	2	2	0
v6-1	0	0	0	0	2	2	2	2
v6-2	0	0	0	0	0	0	0	0
v6-3	0	0	0	0	0	0	1	1
v6-4	0	0	0	0	0	0	2	0

Table A.7: Number of generation issues per prompt, metric and document length setting for **Llama-3_3-Nemotron-Super-49B-v1-Think**

	Generation Issues Gold				Generation Issues Manipulated			
Prompt	2000 F	Full C	2000 F	Full C	2000 F	Full C	2000 F	Full C
v1	0	0	0	0	0	0	0	0
v2	2	2	2	2	20	10	16	6
v3	0	0	1	1	12	9	5	1
v4-2	2	2	2	1	14	19	2	15
v5-1	0	0	0	0	1	1	0	0
v5-2	0	0	0	0	0	0	0	0
v5-3	0	0	0	0	0	0	0	0
v5-4	0	0	0	0	1	1	0	0
v5-5	0	0	0	0	1	1	0	0
v5-6	0	0	0	0	0	0	0	0
v5-7	0	0	0	0	0	0	1	1
v5-8	0	0	0	0	0	0	2	0
v6-1	0	0	0	0	0	0	1	1
v6-2	0	0	0	0	0	0	0	0
v6-3	0	0	0	0	0	0	1	1
v6-4	0	0	0	0	0	0	0	0

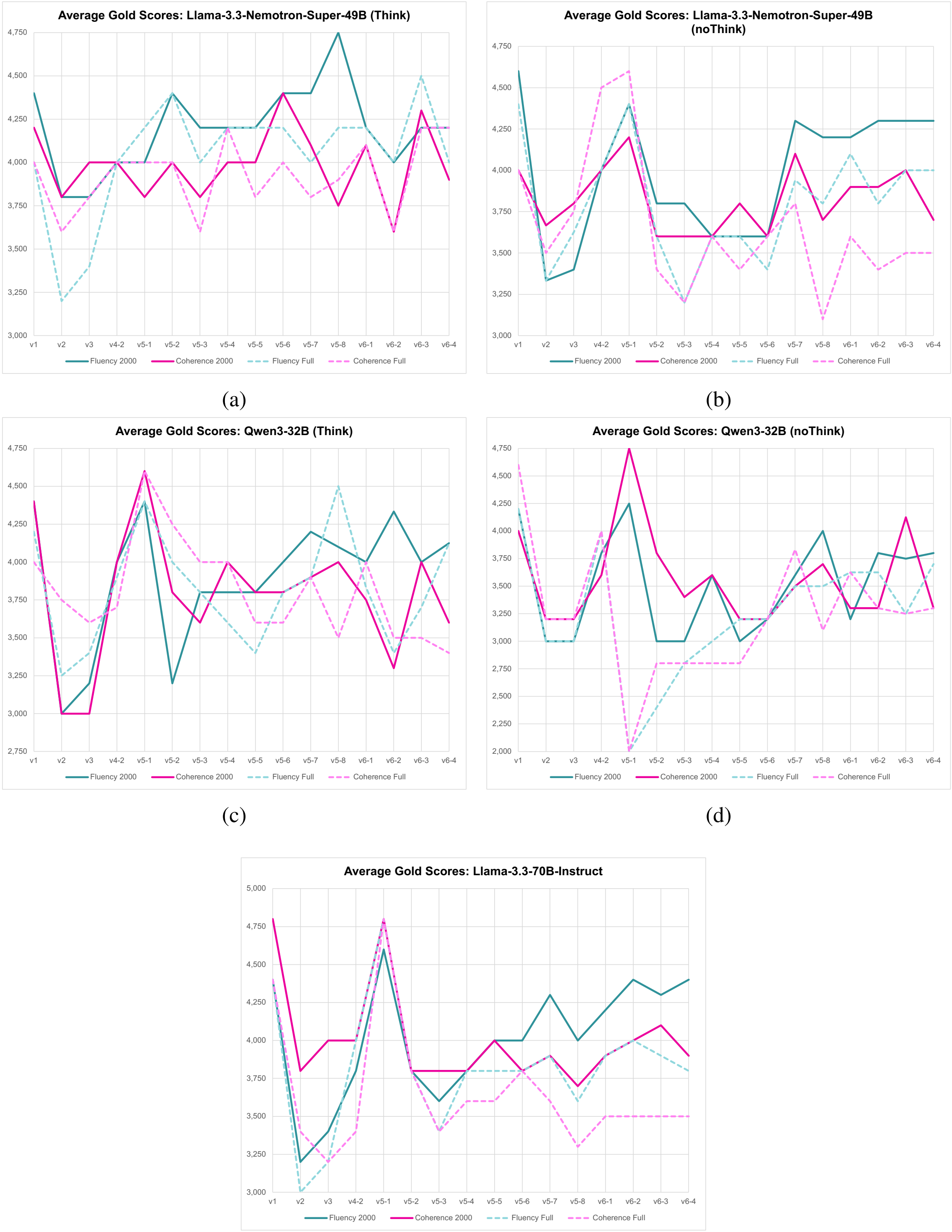
Table A.8: Number of generation issues per prompt, metric and document length setting for **Llama-3_3-Nemotron-Super-49B-v1-noThink**

Prompt	Generation Issues Gold				Generation Issues Manipulated			
	2000 F	Full C	2000 F	Full C	2000 F	Full C	2000 F	Full C
v1	0	0	0	0	0	0	1	1
v2	0	0	1	1	2	2	10	10
v3	0	0	0	0	0	0	3	3
v4-2	0	0	0	0	4	4	3	3
v5-1	0	0	1	1	1	1	4	4
v5-2	0	0	1	1	1	1	4	
v5-3	0	0	0	0	3	3	3	3
v5-4	0	0	0	0	0	0	2	2
v5-5	0	0	0	0	3	3	3	3
v5-6	0	0	0	0	3	3	1	1
v5-7	0	0	0	0	7	7	8	7
v5-8	0	0	1	0	8	0	13	1
v6-1	1	1	2	2	9	9	3	3
v6-2	2	0	0	0	5	0	5	1
v6-3	0	0	0	0	5	5	3	3
v6-4	1	0	1	0	5	0	6	0

Table A.9: Number of generation issues per prompt, metric and document length setting for **Qwen3-32B-Think**

Prompt	Generation Issues Gold				Generation Issues Manipulated			
	2000 F	Full C	2000 F	Full C	2000 F	Full C	2000 F	Full C
v1	0	0	0	0	0	0	0	0
v2	0	0	0	0	5	5	5	5
v3	0	0	0	0	1	1	5	5
v4-2	0	0	0	0	0	0	2	2
v5-1	0	0	0	0	0	0	0	0
v5-2	0	0	0	0	0	0	1	1
v5-3	0	0	0	0	0	0	5	5
v5-4	0	0	0	0	4	4	5	5
v5-5	0	0	0	0	1	1	4	4
v5-6	0	0	0	0	2	2	2	2
v5-7	0	0	2	2	5	5	12	12
v5-8	0	0	1	0	4	0	14	0
v6-1	0	0	1	1	2	2	8	8
v6-2	0	0	1	0	3	0	10	0
v6-3	1	1	1	1	3	3	14	14
v6-4	0	0	0	0	0	0	7	0

Table A.10: Number of generation issues per prompt, metric and document length setting for **Qwen3-32B-noThink**



Average Gold Scores: Llama-3.3-Nemotron-Super-49B (Think)

(a)

Average Gold Scores: Llama-3.3-Nemotron-Super-49B (noThink)

(b)

Average Gold Scores: Qwen3-32B (Think)

(c)

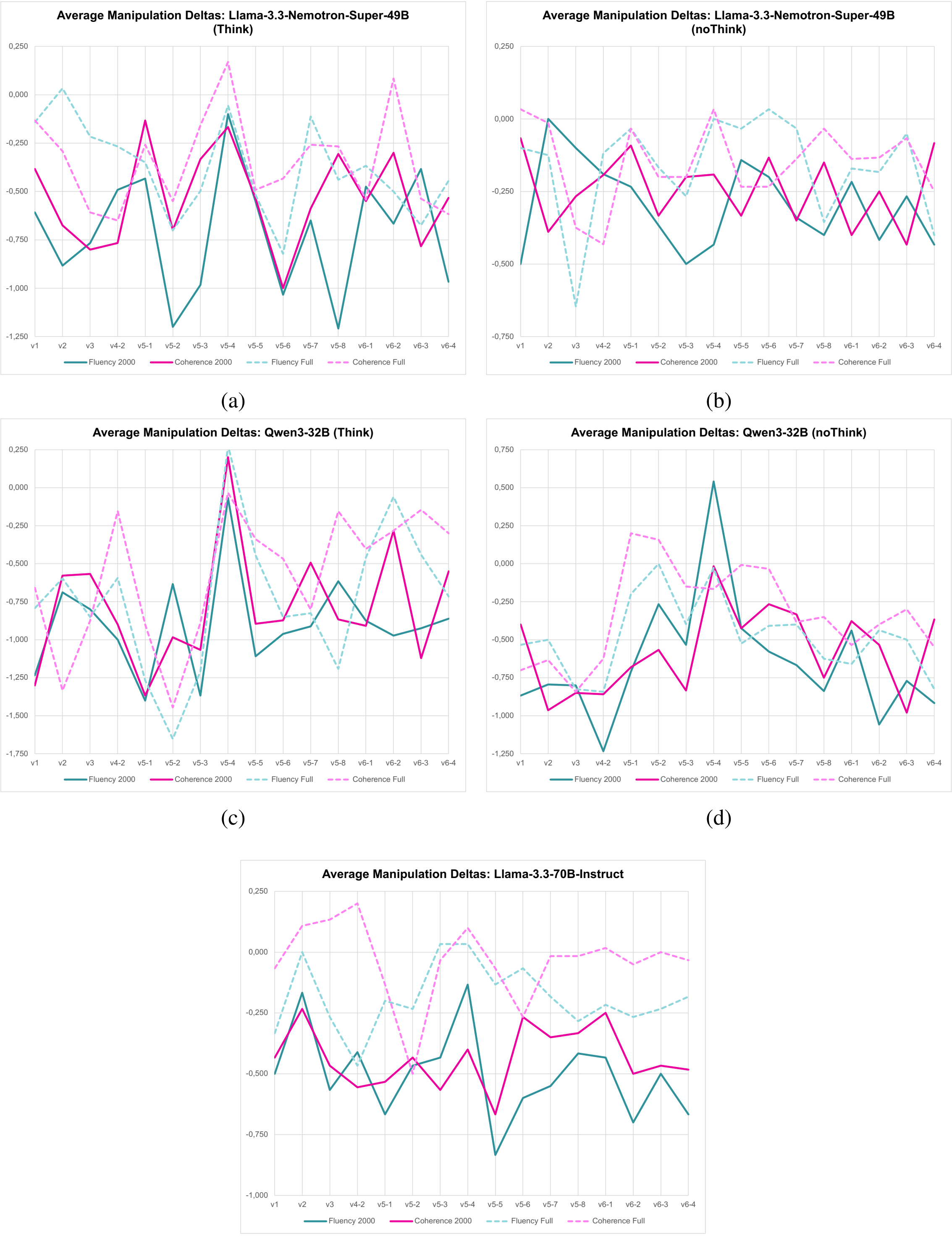
Average Gold Scores: Qwen3-32B (noThink)

(d)

Average Gold Scores: Llama-3.3-70B-Instruct

(e)

Figure A.26: Average gold scores Score(d_{gold}, e) for each evaluation metric e. Each subfigure reports the average score for one evaluating models across all prompt versions for fluency and coherence on both the 2K and Full document settings.



(a)

(b)

(c)

(d)

(e)

Figure A.27: Average manipulation deltas (cf. Equation 3.22) for each evaluation metric e, computed across all manipulation types t. Each subfigure reports the average delta for one evaluating model across all prompt versions for fluency and coherence on both the 2K and Full document settings.

	Generation Issues Gold				**Generation Issues Manipulated**			
Prompt	`2000`	`Full`	`2000`	`Full`	`2000`	`Full`	`2000`	`Full`
	F	C	F	C	F	C	F	C
v1	0	0	0	0	0	0	0	0
v2	0	0	0	0	0	0	1	1
v3	0	0	0	0	0	0	0	0
v4-2	0	0	0	0	2	2	0	0
v5-1	0	0	0	0	0	0	0	0
v5-2	0	0	0	0	0	0	0	0
v5-3	0	0	0	0	0	0	0	0
v5-4	0	0	0	0	0	0	0	0
v5-5	0	0	0	0	0	0	0	0
v5-6	0	0	0	0	0	0	0	0
v5-7	0	0	0	0	0	0	0	0
v5-8	0	0	0	0	0	0	0	0
v6-1	0	0	0	0	0	0	0	0
v6-2	0	0	0	0	0	0	0	0
v6-3	0	0	0	0	0	0	0	0
v6-4	0	0	0	0	0	0	0	0

Table A.11: Number of generation issues per prompt, metric and document length setting for **Llama-3.3-70B-Instruct**

A.7 Additional Experimental Results

This section complements Chapter 5 with further qualitative examples and ablation study results.

Qualitative Examples

```
...
- [SPELLING] "neber" (should be "never")
- [SPELLING] "worl" (should be "work")
- [SPELLING] "fo" (should be "to")
- [SPELLING] "xre" (should be "are")
- [SPELLING] "cchad" (should be "had")
- [SPELLING] "lf" (should be "of")
...
```

Figure A.28: Manipulation [41]: Excerpt from model-generated evaluation output for Qwen3-32B-Think in the vanilla evaluation set up. The model lists all occuring issues, which causes it to reach the maximum generation length before assigning a final score to the document.

```
...
Evaluation Form:
1) Fluency Issues:
- [SPELLING] Multiple minor spelling errors (e.g.,
"thwir", "myn", "sdlf-conscious", "txd", "myn", "wz",
"xomplacency", "fhe", "myn", "fhe", "myn", "fhe", "myn",
"fhe", "myn", "fhe", "myn", "fhe", "myn", "fhe", "myn",
"fhe", "myn", "fhe", "myn", "fhe", "myn", "fhe", "myn",
"fhe", "myn", "fhe", "myn", "fhe", "myn", "fhe", "myn",
"fhe", "myn", "fhe", "myn", "fhe", "myn", "fhe", "myn",
"fhe", "myn", "fhe", "myn", "fhe", "myn", "fhe", "myn",
"fhe", "myn", "fhe", "myn", "fhe", "myn", "fhe",
...
```

Figure A.29: Manipulation [41]: Excerpt from model-generated evaluation output for Llama-3_3-Nemotron-Super-49B-v1-noThink in the vanilla evaluation set up. The model repeatedly lists recurring issues, which causes it to reach the maximum generation length before assigning a final score to the document.

Average Scores and Stats

		Average Scores						
							t-statistics	
Task	**Metric**	**LN-noT**	**LN-T**	**Q-noT**	**Q-T**	**LI**	**Avg.**	**Stdev**
gold	Fluency	3.915	4.092	3.677	3.985	4.362		
gold	Coherence	4.354	4.331	3.615	4.069	4.015		
[43]	Fluency	3.738	3.777	3.062	3.146	3.969	-7.010	2.132
[41]	Fluency	3.836	4.000	3.623	3.664	3.715	-4.484	4.283
[42]	Fluency	3.862	3.908	3.546	3.654	4.254	-2.935	0.992
[24]	Coherence	4.282	4.015	3.669	3.823	3.969	-2.131	1.916
[17]	Coherence	3.877	4.115	3.408	3.708	3.446	-6.178	3.059
[32]	Coherence	4.131	3.931	3.000	3.408	3.877	-5.942	2.066
Avg. Fluency t-statistics		-2.918	-3.143	-3.526	-6.205	-8.257		
Avg. Coherence t-statistics		-5.531	-4.557	-3.835	-5.182	-4.647		
Avg. Overall t-statistics		-4.225	-3.850	-3.681	-5.693	-6.452		

Table A.12: Summary of average model scores and t-statistics on CoFluEval-SC (`2K` setting; cf. Equation 5.2) for the vanilla baseline evaluation. Scores for each task and metric are shown alongside gold references. The table also includes standard deviations and average t-statistics for each model.

model abbreviations as in Table A.2

		t-statistics						
Task	**Metric**	**LN-noT**	**LN-T**	**Q-noT**	**Q-T**	**LI**	**Avg.**	**Stdev**
[43]	Fluency	-1.219	0.280	1.723	-1.176	-2.600	-0.598	1.476
[41]	Fluency	0.179	1.224	0.131	1.449	0.268	0.650	0.567
[42]	Fluency	0.164	0.653	2.275	0.600	-0.423	0.654	0.898
[24]	Coherence	8.083	2.881	2.519	3.371	-7.014	1.968	4.923
[17]	Coherence	5.726	7.520	7.000	11.155	-13.700	3.540	8.807
[32]	Coherence	4.667	3.192	2.330	2.189	-1.675	2.141	2.102
Avg. Fluency t-statistics		-0.292	0.719	1.376	0.291	-0.918		
Avg. Coherence t-statistics		6.159	4.531	3.950	5.572	-7.463		
Avg. Overall t-statistics		2.933	2.625	2.663	2.931	-4.191		

Table A.13: Summary of t-statistics on CoFluEval-SC (`2K-Full` setting; cf. Equation 5.3) for the vanilla baseline evaluation.

model abbreviations as in Table A.2

		Average Scores						
Task	**Metric**	**LN-noT**	**LN-T**	**Q-noT**	**Q-T**	**LI**	***t*-statistics** **Avg.**	**Stdev**
gold	Fluency	3.833	4.033	—	4.008	3.950		
gold	Coherence	4.333	4.400	—	4.117	4.350		
[43]	Fluency	3.442	3.667	—	3.067	3.683	-6.410	1.890
[41]	Fluency	3.558	3.833	—	3.517	3.600	-5.200	2.103
[42]	Fluency	3.742	3.783	—	3.617	3.750	-3.436	1.257
[24]	Coherence	4.133	3.900	—	3.858	4.267	-5.821	4.243
[17]	Coherence	3.917	4.117	—	3.658	3.992	-7.923	5.369
[32]	Coherence	4.108	3.867	—	3.467	4.042	-6.576	2.772
Avg. Fluency *t*-statistics		-5-390	-2.446	—	-6.340	-5.884		
Avg. Coherence *t*-statistics		-13.706	-4.990	—	-4.660	-3.737		
Avg. Overall *t*-statistics		-9.548	-3.718	—	-5.500	-4.810		

Table A.14: Summary of average model scores and t-statistics on CoFluEval-SC (2K setting; cf. Equation 5.2) for JudgeMemo. Scores for each task and metric are shown alongside gold references. The table also includes standard deviations and average t-statistics for each model.

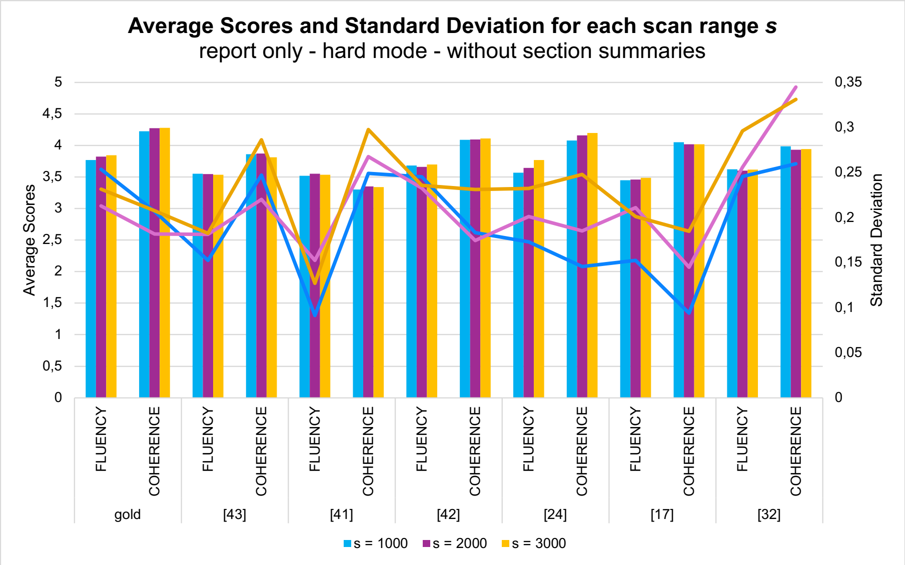
model abbreviations as in Table A.2

		t-statistics						
Task	**Metric**	**LN-noT**	**LN-T**	**Q-noT**	**Q-T**	**LI**	**Avg.**	**Stdev**
[43]	Fluency	-3.014	0.203	—	-2.032	0.779	-1.016	1.560
[41]	Fluency	-0.595	0.000	—	-0.440	-0.703	-0.434	0.268
[42]	Fluency	1.294	-1.091	—	-1.076	-0.109	-0.245	0.974
[24]	Coherence	-0.886	-3.366	—	-0.670	0.456	-1.117	1.395
[17]	Coherence	-2.075	-0.905	—	-1.330	-1.051	-1.340	0.451
[32]	Coherence	-0.560	-0.710	—	0.247	1.165	0.036	0.747
Avg. Fluency *t*-statistics		-0.772	-0.296	—	-1.183	-0.011		
Avg. Coherence *t*-statistics		-1.174	-1.660	—	-0.584	0.190		
Avg. Overall *t*-statistics		-0.973	-0.978	—	-0.884	0.090		

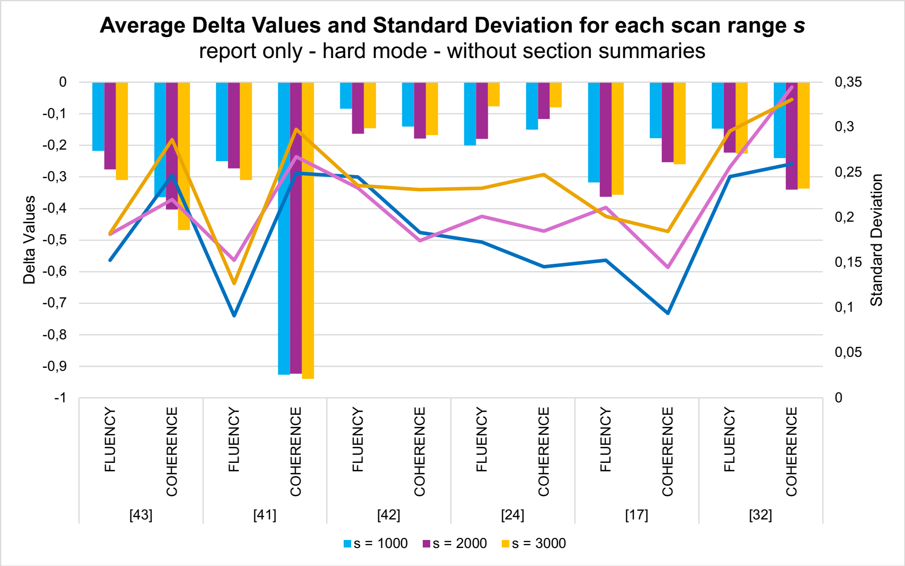
Table A.15: Summary of t-statistics on CoFluEval-SC (2K-Full setting; cf. Equation 5.3) for JudgeMemo.

model abbreviations as in Table A.2

Insights from Ablation Studies

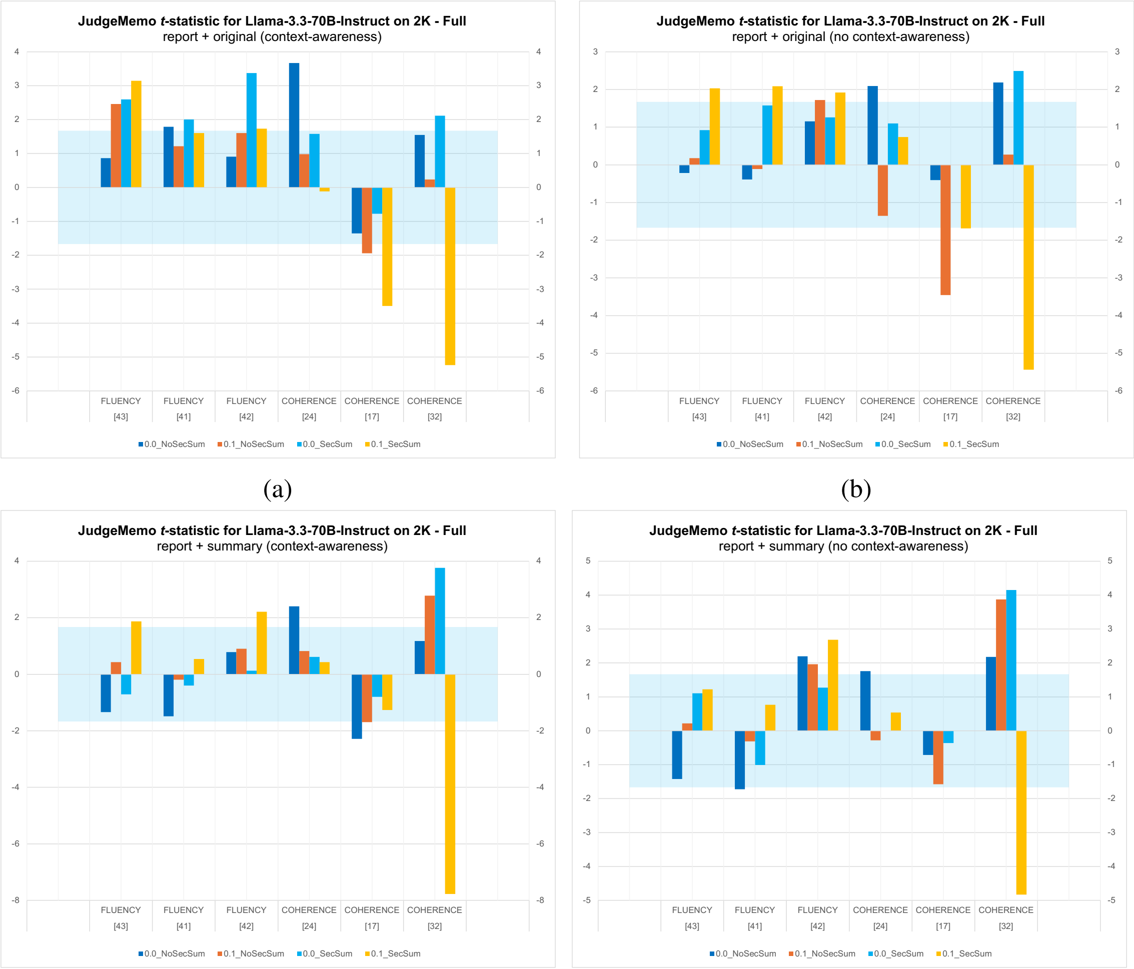


(a) Average scores and corresponding standard deviations σ for CoFluEval-LC $_{30}$.



(b) Average Δ-values and corresponding standard deviations σ for CoFluEval-LC $_{30}$.

Figure A.30: Scan range ablation studies with $s \in 1000, 2000, 3000$ for JudgeMemo (ours) in the `report only` setting, using *hard* mode with $o = 0.0$ on Llama-3.3-70B-Instruct, without section summaries.



(a)

(b)

(c)

(d)

Figure A.31: Ablation studies for JUDGEMEMO (ours) in the `report + original` (top), and `report + summary` (bottom) settings on Llama-3.3-70B-Instruct. Left plots use context-aware prompts; right plots use non-context-aware ones. t-statistics are computed over the difference in Δ-values between `2K` and `Full` documents. Values within the light blue area indicate non-significance, suggesting consistent performance across context lengths.

B Prompts

Prompts used in Vanilla Evaluation Setup

Below, we highlight the most important prompts developed during the prompt engineering process as outlined in Section 4.1. The other prompts mentioned in Section 4.1 are available in our GitHub repository[23]. All prompts were carefully crafted to optimize performance across all models, reflecting iterative refinements based on preliminary results.

System Evaluation

You are a human annotator that rates the quality of texts.

Prompt B.1: System prompt consistently used across all models and experiments to instruct the evalutator's role.

Prompt v1

Imagine you are a human annotator now. You will evaluate the quality of a given written story. Please follow these steps:
1. Carefully read the story, and be aware of the information it contains.
2. Rate the story on two dimensions: fluency, and coherence.
You should rate on a scale from 1 (worst) to 5 (best).
Definitions are as follows:
Fluency: This rating measures the quality of individual sentences, whether they are well-written and grammatically correct. Consider the quality of individual sentences.
Coherence: The rating measures the quality of all sentences collectively. The text should be well-structured and well-organized, it should build from sentence to sentence to form a coherent body of information about the text. Consider the quality of the text as a whole.
Explain the quality of the text briefly in a few lines (100-200 tokens) regarding fluency and coherence.
Afterwards, return the final scores (1 to 5) in new lines starting with each aspect as following:
FINAL Coherence Score: [SCORE]
FINAL Fluency Score: [SCORE]

The story text is given below:
Story: {Story}

Explain the quality of the text briefly in a few lines (100-200 tokens) regarding fluency and coherence. Afterwards, return the final scores (1 to 5) in new lines starting with each aspect as following:
FINAL Coherence Score: [SCORE]
FINAL Fluency Score: [SCORE]

Prompt B.2: Prompt `v1`: Baseline user prompt employed for initial manipulation tests and sanity checks, guiding the model in evaluating fluency and coherence of generated texts.

[23] https://github.com/hkleiner/JudgeMemo

Prompt v6-3

You will be given a human-written text. Your task is to rate the text according to the evaluation criterion on a Likert scale from 1 to 5.
You are allowed to use half-points: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. Make sure you read and understand these instructions carefully.

# Task Description:
1. Before producing your output, follow these internal steps:
    a) For each metric, detect problems in the text that lead to a point deduction.
    b) For each metric, summarize the detected problems.
2. For each metric, give **the most serious problems** in the text that justify point deductions. Use bullet points only.
3. Label each issue with a category that describes the issue best, e.g.
- for *fluency*: [GRAMMAR], [SPELLING], [SYNTAX], [LEXICON], or
- for *coherence*: [LOGIC], [STRUCTURE], [CLARITY], [TRANSITION].
These are examples - you may create your own label if it better fits the issue. Each label should reflect the type of problem that best describes the issue.
4. For each metric, give a score between 1 and 5. You are allowed to use half-points. You should refer to the requested metrics criteria and corresponding accuracy scales.
5. It is forbidden to generate any other opening, closing, and explanations.
6. It is forbidden to give corrections for detected issues.

# Evaluation Criterion and Metric Accuracy Scale:
## FLUENCY
Definition: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
Scale:
- Score 5: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding.
- Score 4: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability.
- Score 3: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways.
- Score 2: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort.
- Score 1: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

## COHERENCE
Definition: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
Scale:
- Score 5: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding.
- Score 4: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text.
- Score 3: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience.
- Score 2: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort.
- Score 1: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

The text to evaluate is given below:
Text: {Story}

You must output only and exactly the following format:
Evaluation Form:
1) Fluency Issues: - [LABEL] [ISSUE] (only name short bullet points)
2) Coherence Issues: - [LABEL] [ISSUE] (only name short bullet points)
3) FINAL Coherence Score: [SCORE] (between 1 and 5; half-points are allowed)
4) FINAL Fluency Score: [SCORE] (between 1 and 5; half-points are allowed)

Your assessment of the text:

Prompt B.3: Prompt `v6-3`: This prompt incorporates detailed definitions for fluency and coherence.

Prompts used in JudgeMemo framework

Our JudgeMemo framework supports multiple prompt configurations, depending on the chosen evaluation setup. The exact prompts vary based on the selected scanning mode, whether section and/or full document summarization is enabled, and the final evaluation strategy - i.e., whether only the section-level report is used, or if it is supplemented with a full-document summary or the full document itself.

For reproducibility and to support future research, we include all prompts used in the JudgeMemo evaluation pipeline. These prompt templates are also available in our GitHub repository[24].

System Summarization

You are a helpful assistant that summarizes text clearly and concisely. Focus on the most important points. Avoid repeating content. Maintain the original meaning without adding new information. Use plain language.

Prompt B.4: System prompt used for section and full document sumamrization tasks in our JudgeMemo framework.

Prompt Document Summarization

Document: {Story}

Summarize the above text in approximately 512 tokens (25 to 35 sentences).

Summary:

Prompt B.5: Summarization prompt used for full document summarization as an optional part of our JudgeMemo pipeline.

Prompt Section Summarization

Section: {Story}

Summarize the above section in four to five sentences.

Summary:

Prompt B.6: Summarization prompt used for section summarization as an optional part of our JudgeMemo pipeline.

[24] https://github.com/hkleiner/JudgeMemo

Prompt Section Evaluation (hard; no summary)

```
You will be given a human-written text. Your task is to rate the text according to the evaluation criterion on a Likert scale from 1 to 5.
You are allowed to use half-points: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. Make sure you read and understand these instructions carefully.

# Task Description:
1. Before producing your output, follow these internal steps:
        a) For each metric, detect problems in the text that lead to a point deduction.
        b) For each metric, summarize the detected problems.
2. For each metric, give **the most serious problems** in the text that justify point deductions. Use bullet points only.
3. Label each issue with a category that describes the issue best, e.g.
- for *fluency*: [GRAMMAR], [SPELLING], [SYNTAX], [LEXICON], or
- for *coherence*: [LOGIC], [STRUCTURE], [CLARITY], [TRANSITION].
These are examples - you may create your own label if it better fits the issue. Each label should reflect the type of problem that best describes the issue.
4. For each metric, give a score between 1 and 5. You are allowed to use half-points. You should refer to the requested metrics criteria and corresponding accuracy scales.
5. It is forbidden to generate any other opening, closing, and explanations.
6. It is forbidden to give corrections for detected issues.

# Evaluation Criterion and Metric Accuracy Scale:
## FLUENCY
**Definition**: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
*Scale*:
- Score 5: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding.
- Score 4: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability.
- Score 3: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways.
- Score 2: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort.
- Score 1: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

## COHERENCE
**Definition**: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
*Scale*:
- Score 5: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding.
- Score 4: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text.
- Score 3: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience.
- Score 2: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort.
- Score 1: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

**IMPORTANT**: The text provided is a section extracted from a longer document. It may be preceded or followed by other parts not shown. Evaluate the section on its own terms, while allowing for the possibility that some context may lie outside the visible excerpt.

The text to evaluate is given below:
Current Section to Evaluate: {Story}

You must output only and exactly the following format:
Evaluation Form:
1) Fluency Issues: - [LABEL] [ISSUE] (only name short bullet points)
2) Coherence Issues: - [LABEL] [ISSUE] (only name short bullet points)
3) FINAL Coherence Score: [SCORE] (between 1 and 5; half-points are allowed)
4) FINAL Fluency Score: [SCORE] (between 1 and 5; half-points are allowed)

Your assessment of the text:
```

Prompt B.7: Section evaluation prompt based on prompt version `v6-3` (cf. Prompt B.3) used for section evaluations in hard scanning mode without section summaries as a fundamental part of our JUDGEMEMO pipeline.

Prompt Section Evaluation (hard; with summary)

```
You will be given a human-written text. Your task is to rate the text according to the evaluation criterion on a Likert scale from 1 to 5.
You are allowed to use half-points: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. Make sure you read and understand these instructions carefully.

# Task Description:
1. Before producing your output, follow these internal steps:
        a) For each metric, detect problems in the text that lead to a point deduction.
        b) For each metric, summarize the detected problems.
2. For each metric, give **the most serious problems** in the text that justify point deductions. Use bullet points only.
3. Label each issue with a category that describes the issue best, e.g.
- for *fluency*: [GRAMMAR], [SPELLING], [SYNTAX], [LEXICON], or
- for *coherence*: [LOGIC], [STRUCTURE], [CLARITY], [TRANSITION].
These are examples - you may create your own label if it better fits the issue. Each label should reflect the type of problem that best describes the issue.
4. For each metric, give a score between 1 and 5. You are allowed to use half-points. You should refer to the requested metrics criteria and corresponding accuracy scales.
5. It is forbidden to generate any other opening, closing, and explanations.
6. It is forbidden to give corrections for detected issues.

# Evaluation Criterion and Metric Accuracy Scale:
## FLUENCY
**Definition**: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
*Scale*:
- Score 5: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding.
- Score 4: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability.
- Score 3: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways.
- Score 2: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort.
- Score 1: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

## COHERENCE
**Definition**: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
*Scale*:
- Score 5: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding.
- Score 4: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text.
- Score 3: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience.
- Score 2: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort.
- Score 1: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

**IMPORTANT**: The text provided is a section extracted from a longer document. It may be preceded or followed by other parts not shown.
Evaluate the section on its own terms, while allowing for the possibility that some context may lie outside the visible excerpt.
A short summary of the preceding section is included to give minimal context.

Previous Section Summary: {AddOn}

Current Section to Evaluate: {Content}

You must output only and exactly the following format:
Evaluation Form:
1) Fluency Issues: - [LABEL] [ISSUE] (only name short bullet points)
2) Coherence Issues: - [LABEL] [ISSUE] (only name short bullet points)
3) FINAL Coherence Score: [SCORE] (between 1 and 5; half-points are allowed)
4) FINAL Fluency Score: [SCORE] (between 1 and 5; half-points are allowed)

Your assessment of the text:
```

Prompt B.8: Section evaluation prompt based on prompt version `v6-3` (cf. Prompt B.3) used for section evaluations in hard scanning mode with section summaries as a fundamental part of our JUDGEMEMO pipeline.

Prompt Section Evaluation (stride; no summary)

```
You will be given a human-written text. Your task is to rate the text according to the evaluation criterion on a Likert scale from 1 to 5.
You are allowed to use half-points: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. Make sure you read and understand these instructions carefully.

# Task Description:
1. Before producing your output, follow these internal steps:
        a) For each metric, detect problems in the text that lead to a point deduction.
        b) For each metric, summarize the detected problems.
2. For each metric, give **the most serious problems** in the text that justify point deductions. Use bullet points only.
3. Label each issue with a category that describes the issue best, e.g.
- for *fluency*: [GRAMMAR], [SPELLING], [SYNTAX], [LEXICON], or
- for *coherence*: [LOGIC], [STRUCTURE], [CLARITY], [TRANSITION].
These are examples - you may create your own label if it better fits the issue. Each label should reflect the type of problem that best describes the issue.
4. For each metric, give a score between 1 and 5. You are allowed to use half-points. You should refer to the requested metrics criteria and corresponding accuracy scales.
5. It is forbidden to generate any other opening, closing, and explanations.
6. It is forbidden to give corrections for detected issues.

# Evaluation Criterion and Metric Accuracy Scale:
## FLUENCY
**Definition**: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
*Scale*:
- Score 5: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding.
- Score 4: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability.
- Score 3: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways.
- Score 2: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort.
- Score 1: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

## COHERENCE
**Definition**: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
*Scale*:
- Score 5: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding.
- Score 4: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text.
- Score 3: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience.
- Score 2: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort.
- Score 1: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

**IMPORTANT**: The text provided is a section extracted from a longer document. It may be preceded or followed by other parts not shown.
Evaluate the section on its own terms, while allowing for the possibility that some context may lie outside the visible excerpt.
You will be provided with two text segments:
- A *context segment*: this is a portion of the text that comes immediately before the section you are asked to evaluate. It has already been evaluated and is included only for context.
- A *current segment*: this is the section to evaluate. Your judgments must be based only on this segment, even if it references or continues ideas from the context.
Do not score or analyze the context segment.

Previous Section (ignore for evaluation): {AddOn}

Current Section to Evaluate: {Content}

You must output only and exactly the following format:
Evaluation Form:
1) Fluency Issues: - [LABEL] [ISSUE] (only name short bullet points)
2) Coherence Issues: - [LABEL] [ISSUE] (only name short bullet points)
3) FINAL Coherence Score: [SCORE] (between 1 and 5; half-points are allowed)
4) FINAL Fluency Score: [SCORE] (between 1 and 5; half-points are allowed)

Your assessment of the text:
```

Prompt B.9: Section evaluation prompt based on prompt version `v6-3` (cf. Prompt B.3) used for section evaluations in stride scanning mode without section summaries as a fundamental part of our JUDGEMEMO pipeline.

Prompt Section Evaluation (stride; with summary)

```
You will be given a human-written text. Your task is to rate the text according to the evaluation criterion on a Likert scale from 1 to 5.
You are allowed to use half-points: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. Make sure you read and understand these instructions carefully.

# Task Description:
1. Before producing your output, follow these internal steps:
        a) For each metric, detect problems in the text that lead to a point deduction.
        b) For each metric, summarize the detected problems.
2. For each metric, give **the most serious problems** in the text that justify point deductions. Use bullet points only.
3. Label each issue with a category that describes the issue best, e.g.
- for *fluency*: [GRAMMAR], [SPELLING], [SYNTAX], [LEXICON], or
- for *coherence*: [LOGIC], [STRUCTURE], [CLARITY], [TRANSITION].
These are examples - you may create your own label if it better fits the issue. Each label should reflect the type of problem that best describes the issue.
4. For each metric, give a score between 1 and 5. You are allowed to use half-points. You should refer to the requested metrics criteria and corresponding accuracy scales.
5. It is forbidden to generate any other opening, closing, and explanations.
6. It is forbidden to give corrections for detected issues.

# Evaluation Criterion and Metric Accuracy Scale:
## FLUENCY
**Definition**: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
*Scale*:
- Score 5: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding.
- Score 4: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability.
- Score 3: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways.
- Score 2: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort.
- Score 1: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

## COHERENCE
**Definition**: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
*Scale*:
- Score 5: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding.
- Score 4: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text.
- Score 3: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience.
- Score 2: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort.
- Score 1: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

**IMPORTANT**: The text provided is a section extracted from a longer document. It may be preceded or followed by other parts not shown.
Evaluate the section on its own terms, while allowing for the possibility that some context may lie outside the visible excerpt.
You will be provided with three text segments:
- A *short summary* of the preceding section to give minimal context.
- A *context segment*: this is a portion of the text that comes immediately before the section you are asked to evaluate. It has already been evaluated and is included only for context.
- A *current segment*: this is the section to evaluate. Your judgments must be based only on this segment, even if it references or continues ideas from the context.
Do not score or analyze the context segment and the summary.

Previous Section Summary (ignore for evaluation): {Text}

Previous Section (ignore for evaluation): {AddOn}

Current Section to Evaluate: {Content}

You must output only and exactly the following format:
Evaluation Form:
1) Fluency Issues: - [LABEL] [ISSUE] (only name short bullet points)
2) Coherence Issues: - [LABEL] [ISSUE] (only name short bullet points)
3) FINAL Coherence Score: [SCORE] (between 1 and 5; half-points are allowed)
4) FINAL Fluency Score: [SCORE] (between 1 and 5; half-points are allowed)

Your assessment of the text:
```

Prompt B.10: Section evaluation prompt based on prompt version `v6-3` (cf. Prompt B.3) used for section evaluations in stride scanning mode with section summaries as a fundamental part of our JUDGEMEMO pipeline.

Prompt Final Assessment (report only)

You will be given a section-wise evaluation report of a human-written text. Your task is to rate the entire text according to the evaluation criterion on a Likert scale from 1 to 5.
You are allowed to use half-points: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. Make sure you read and understand these instructions carefully.

# Task Description:
1. Before producing your output, follow these internal steps:
    a) For each metric, analyze the section-wise report and detect recurring or serious problems that lead to point deduction. Each section includes scores and labeled issues for fluency and coherence.
    b) For each metric, summarize the most impactful problems across sections.
2. For each metric, give a score between 1 and 5. You are allowed to use half-points. You should refer to the requested metrics criteria and corresponding accuracy scales.
3. It is forbidden to generate any other opening, closing, and explanations.
4. It is forbidden to give corrections for detected issues.

# Evaluation Criterion and Metric Accuracy Scale:
## FLUENCY
Definition: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
Scale:
- Score 5: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding.
- Score 4: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability.
- Score 3: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways.
- Score 2: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort.
- Score 1: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

## COHERENCE
Definition: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
Scale:
- Score 5: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding.
- Score 4: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text.
- Score 3: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience.
- Score 2: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort.
- Score 1: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

The report is given below:
Report: {Story}

You must output only and exactly the following format:
Evaluation Form:
1) FINAL Coherence Score: [SCORE] (between 1 and 5; half-points are allowed)
2) FINAL Fluency Score: [SCORE] (between 1 and 5; half-points are allowed)

Your assessment of the text:

Prompt B.11: Final assessment prompt based on just the report obtained from section evaluations. This prompt is used for the final evaluation of the full document as a fundamental part of our JUDGEMEMO pipeline.

Prompt Final Assessment (report + original)

```
You will be given a section-wise evaluation report of a human-written text. Your task is to rate the entire text according to the evaluation criterion on a Likert scale from 1 to 5.
You are allowed to use half-points: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. Make sure you read and understand these instructions carefully.

# Task Description:
1. Before producing your output, follow these internal steps:
    a) Read and understand the text the report was created for.
    b) For each metric, analyze the section-wise report and detect recurring or serious problems that lead to point deduction. Each section includes scores and labeled issues for fluency and coherence.
    c) For each metric, summarize the most impactful problems across sections.
2. For each metric, give a score between 1 and 5. You are allowed to use half-points. You should refer to the requested metrics criteria and corresponding accuracy scales.
3. It is forbidden to generate any other opening, closing, and explanations.
4. It is forbidden to give corrections for detected issues.

# Evaluation Criterion and Metric Accuracy Scale:
## FLUENCY
**Definition**: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
*Scale*:
- Score 5: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding.
- Score 4: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability.
- Score 3: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways.
- Score 2: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort.
- Score 1: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

## COHERENCE
**Definition**: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
*Scale*:
- Score 5: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding.
- Score 4: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text.
- Score 3: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience.
- Score 2: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort.
- Score 1: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

The text to be evaluated is given below:
Text: {AddOn}

The report for the text is given below:
Report: {Content}

You must output only and exactly the following format:
Evaluation Form:
1) FINAL Coherence Score: [SCORE] (between 1 and 5; half-points are allowed)
2) FINAL Fluency Score: [SCORE] (between 1 and 5; half-points are allowed)

Your assessment of the text:
```

Prompt B.12: Final assessment prompt based on the report obtained from section evaluations and the original full document. This prompt is used for the final evaluation of the full document as a fundamental part of our JudgeMemo pipeline.

Prompt Final Assessment (report + summary)

```
You will be given a section-wise evaluation report of a human-written text. Your task is to rate the entire text according to the evaluation criterion on a Likert scale from 1 to 5.
You are allowed to use half-points: [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]. Make sure you read and understand these instructions carefully.

# Task Description:
1. Before producing your output, follow these internal steps:
    a) Read and understand the summary of the text the report was created for.
    b) For each metric, analyze the section-wise report and detect recurring or serious problems that lead to point deduction. Each section includes scores and labeled issues for fluency and coherence.
    c) For each metric, summarize the most impactful problems across sections.
2. For each metric, give a score between 1 and 5. You are allowed to use half-points. You should refer to the requested metrics criteria and corresponding accuracy scales.
3. It is forbidden to generate any other opening, closing, and explanations.
4. It is forbidden to give corrections for detected issues.

# Evaluation Criterion and Metric Accuracy Scale:
## FLUENCY
**Definition**: Fluency assesses how the text reads from start to finish. It mainly focuses on syntax, grammar, spelling, word choice, phrasing, and punctuation of individual sentences. It ensures that the language flows smoothly without awkward phrasing or errors.
*Scale*:
- Score 5: Highly fluent, with clear, natural phrasing and minimal to no grammatical issues. Any errors, if present, are rare and do not distract from reading or understanding.
- Score 4: Mostly fluent and well-structured, though may contain minor issues in grammar or phrasing. These issues may be noticeable but do not disrupt the overall readability.
- Score 3: Generally readable and mostly fluent, but contains multiple grammatical or structural issues that interrupt the flow or clarity of the text in noticeable ways.
- Score 2: Text contains frequent grammatical errors, awkward phrasing, or confusing structure. Some segments are clear, but comprehension is often difficult without effort.
- Score 1: Largely unintelligible or fragmented. The text lacks coherent structure or meaning, making it very difficult or impossible to understand.

## COHERENCE
**Definition**: Coherence assesses how the story unfolds as a whole. Important criteria are logically sequenced, non-repetitive and smoothly connected ideas, a clear progression from one section to another, avoidance of ambiguities and abrupt jumps, and how consistent and clear structured the narrative is.
*Scale*:
- Score 5: Highly coherent, with a clear and logical progression throughout. Sentences and ideas connect smoothly to form a unified and well-organized whole. Minor lapses, if any, are barely noticeable and do not hinder understanding.
- Score 4: Mostly coherent, with a few weak or slightly disconnected parts. These do not significantly disrupt the flow or understanding of the text.
- Score 3: Generally coherent, but contains noticeable abrupt shifts, unclear transitions, or confusing segments that are not resolved and affect the reading experience.
- Score 2: The text has frequent inconsistencies, disconnected ideas, or illogical sequencing. Some parts may be understandable, but the overall coherence is difficult to follow without extra effort.
- Score 1: Largely incoherent. The text lacks logical structure or progression, with many disjointed, contradictory, or confusing segments that make understanding nearly impossible.

The summary of the text to be evaluated is given below:
Summary: {AddOn}

The report for the text is given below:
Report: {Content}

You must output only and exactly the following format:
Evaluation Form:
1) FINAL Coherence Score: [SCORE] (between 1 and 5; half-points are allowed)
2) FINAL Fluency Score: [SCORE] (between 1 and 5; half-points are allowed)

Your assessment of the text:
```

Prompt B.13: Final assessment prompt based on the report obtained from section evaluations and the summary of the full document. This prompt is used for the final evaluation of the full document as a fundamental part of our JUDGEMEMO pipeline.

C Submitted Software and Data Files

All scripts, datasets, and experiment results related to this master thesis are publicly available on GitHub[25]. Additionally, an online version of this thesis can be accessed through the same repository.

Overview of Submitted Files An overview of the files submitted alongside this thesis is provided below for easy reference.

- `Dataset_Analysis/` - code for dataset analysis during data collection processes
- `Dataset_Creation/` - code for Project Gutenberg data preprocessing and dataset creation (gold and manipulated)
  - `Dataset_Creation/DatasetCreator/` - preprocessing package
  - `Dataset_Creation/templates/` - `.json`-templates for dataset creation
  - `Dataset_Creation/TextManipulation/` - data manipulation package
  - `Dataset_Creation/apply_manipulations_to_data.py` - script to manipulate gold documents
- `data/` - datasets and processed data files
- `baseline_model_inference/` - vanilla evaluation pipeline and prompt engineering scripts
  - `baseline_model_inference/prompts/` - all prompts needed to run our vanilla pipeline
- `experiments/` - experimental results for vanilla evaluation set up and prompt engineering
- `JudgeMemo_Method/` - code for our framework JUDGEMEMO
  - `JudgeMemo_Method/JudgeMemo/` - pipeline package
  - `JudgeMemo_Method/prompts/` - all prompts needed to run our pipeline (with context-awareness)
  - `JudgeMemo_Method/prompts_no-context/` - all prompts needed to run our pipeline (without context-awareness)
  - `JudgeMemo_Method/pipeline_JudgeMemo.py` - script to run our framework
- `experiments_JM/` - experimental results (outputs) for our method (JUDGEMEMO) and ablation studies
- `requirements.txt` - Lists Python dependencies for installation via pip
- `environment.yml` - Defines the conda environment and its dependencies
- `README.md` - File with installation instructions and overview of repository.

[25] https://github.com/hkleiner/JudgeMemo