DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Studying the Effectiveness of Longer Context Windows in LLMs for Text Summarization and Question Answering Tasks

Clemens Magg

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Studying the Effectiveness of Longer Context Windows in LLMs for Text Summarization and Question Answering Tasks

Untersuchung der Effektivität längerer Kontextfenster in LLMs für Textzusammenfassungs -und Fragenbeantwortungsaufgaben

Author:	Clemens Magg
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Anum Afzal, M.Sc.
Submission Date:	24.02.2025

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 24.02.2025 Clemens Magg

AI Assistant Usage Disclosure

Introduction

Performing work or conducting research at the Chair of Software Engineering for Business Information Systems (sebis) at TUM often entails dynamic and multi-faceted tasks. At sebis, we promote the responsible use of *AI Assistants* in the effective and efficient completion of such work. However, in the spirit of ethical and transparent research, we require all student researchers working with sebis to disclose their usage of such assistants.

For examples of correct and incorrect AI Assistant usage, please refer to the original, unabridged version of this form, located at this link.

Use of *AI Assistants* for Research Purposes

I have used AI Assistant(s) for the purposes of my research as part of this thesis.

X **Yes** ☐ **No**

Explanation: I have used Grammarly and ChatGPT for spelling corrections and grammar improvements. For debugging, I have used ChatGPT and Copilot.

I confirm in signing below, that I have reported all usage of AI Assistants for my research, and that the report is truthful and complete.

Munich, 24.02.2025
Location, Date Author

Acknowledgments

I sincerely thank Prof. Dr. Florian Matthes and the entire Chair of Software Engineering for Business Information Systems for the opportunity to complete my thesis under their supervision. I am particularly indebted to my advisor, Anum Afzal, whose insightful guidance and steadfast support were instrumental in the success of this thesis. I am immeasurably thankful to my parents, who always came to support me throughout this and all other chapters of my life.

Abstract

Previous language models (LMs) have been inadequate for long-text summarization due to their limited context windows, which restrict their ability to effectively process and comprehend extensive textual data. This limitation has made it challenging for traditional LMs to capture critical long-range dependencies and contextual nuances in longer documents. Models with significantly longer context windows have emerged through various innovative improvements to model architecture and training practices. Large language models (LLMs) are tremendously effective in processing large amounts of textual data and show outstanding results for various NLP tasks. Despite their proficiency with extended context windows, quantifying the performance of LLMs on text summarization and question-answering tasks remains largely unexplored on context window sizes of up to 128k tokens, representing a significant research gap in the field of NLP.

To address this gap, our research implements new traceability metrics to evaluate how well LLMs utilize information within their extended context windows for long-text summarization and question-answering tasks. We propose an evaluation framework integrating automated traceability metrics to provide a nuanced understanding of model performance. Our approach focuses on determining which information models use and which they neglect depending on their context window size.

For our evaluation, we employ three datasets, InfiniteBench, BookSum, and XSum, that exhibit characteristics essential for extended context windows, such as uniform distribution of salient information. We compare the performance across four models: LLaMA 3.1, Qwen 2.5, Phi 3, and Command R7B.

Ultimately, our research reveals that the models exhibit inherent positional biases, leading to varying performance based on the location of information within their context window and the overall length of that context window. These findings highlight the importance of understanding how context dynamics influence the efficacy of text summarization and question-answering tasks.

Keywords: Large Language Models, Benchmarking, Text Summarization, Question-Answering

Contents

1 Introduction

1.1 Motivation

When used for text summarization and question-answering, LLMs hold immense potential for big data applications and individual use cases, such as consuming news or gaining new knowledge. Condensing information spread over a long document can bring tremendous value and time savings to various fields of research, business, and other key areas that engage with large amounts of data. The advent of vast quantities of easily accessible digital information, exemplified by the exponential growth of scientific publications [1] and the commodification of digital data [2], necessitates practices and tools that help in effectively managing, analyzing, and synthesizing this information. Natural language processing (NLP) research strives to develop systems that help navigate complex and information-rich fields. LLM summarizers and question-answering systems have been demonstrated to be effective productivity drivers across diverse domains, including law, medicine, and finance, by providing concise and relevant summaries that enable users to quickly grasp essential information and make informed decisions [3].

LLMs with context window sizes of more than a hundred thousand tokens, like Open Ai's 'GPT-4o' or the Meta ' Llama3.1' models, enable inference over exceptionally long documents. However, summarizing such extensive documents presents significant challenges for LLMs. Long texts often contain causal information chained over wide-ranging distances, making it necessary for a model during inference to distribute its attention over all parts of the text and retain early information for later reference. Directing the model's attention to the parts of the text that hold the most important information is key to generating summaries that align well with human expectations. Additional challenges emerge with more abstract and nonlinear text forms, such as novels, which necessitate that LLMs grasp contextual nuances and reduce the likelihood of generating hallucinations. For LLM-generated summaries to be valuable rather than problematic, they must be trusted sources of accurate, relevant, and complete information. Therefore, it is crucial to rigorously assess how well a model understands a text and how effectively it represents the text in its summary. NLP research has developed numerous performance benchmarking tools and metrics to safeguard the reliability and credibility of LLMs when dealing with extended context windows. However, we found that many of these tests assess only limited context window sizes and thus do not capture the latest advances and innovations in extended context window sizes. Conventional benchmarks that test only up to a limited context window size are inadequate for assessing models capable of processing entire books simultaneously, as they do not sufficiently account for long-range dependencies and comprehensive contextual understanding. Our approach to evaluating

the performance of LLMs seeks to adapt to increasingly large context window sizes, aiming to determine whether more information equals higher quality model output or if models become overwhelmed and display weaknesses, like catastrophic forgetting or hallucinations.

1.2 Problem Statement

This thesis will examine how large language models (LLMs) deal with extended context window sizes. We propose a methodology for evaluating the long-document comprehension capabilities of modern LLMs by implementing a benchmarking framework. This evaluation pipeline aims to systematically assess and quantify the performance of various open-source LLMs that employ distinct context window extension techniques aimed at effectively processing increasingly large amounts of data. The methodology encompasses text summarization and question-answering tasks to test how candidate LLMs use provided information and whether they can accurately capture and reproduce the most salient features of a text.

1.3 Research Questions

In this thesis, we investigate various approaches to understanding how large language models process long context windows while also developing our evaluation pipeline. We focus on three key research questions that will guide and set the objectives of this thesis.

- **RQ1:** *What are the most effective techniques for extending the context window of LLMs?*

The first research question puts together an overview of recent developments regarding long context window understanding. It will cover methodologies and technologies that help modern LLMs extend their pre-trained context window size. This presentation of multiple approaches offers room for comparison, which will be utilized later in this thesis.

- **RQ2:** *How can we adequately test the quality of text summarization and question answering produced by LLMs? Does the quality of the generated summary or answer improve when more content from the article is provided to the model?*

For the second research question, we will discuss how to best measure the long document performance of LLMs. For this purpose, we will develop an evaluation pipeline comprising three text-processing tasks. We want to understand how LLMs utilize their input data and whether more information from extended context windows changes the performance of LLMs.

- **RQ3:** *How do LLMs utilize the information within their context window during text summarization and question-answering tasks? Can LLMs distribute their attention uniformly across the entire document, or is their attention clustered towards specific sections?*

The final research question focuses on analyzing the evaluation results of a collection of models that implement selected context window extension techniques discussed in RQ1. We will generate performance indicators and graphs that present the differences between the models and the different datasets, measuring factors such as retention of relevant information, completeness, and accuracy in summarization tasks.

1.4 Thesis Outline

The following chapter discusses relevant background for NLP principles and methods that enable long context window understanding of LLMs, covering model benchmarking approaches and concepts relevant to the thesis, including automated text processing, transformers, similarity metrics, and model prompting. These topics provide the theoretical foundation necessary for understanding the methodologies applied in later sections. Chapter 3 reviews previous research in areas such as question answering, benchmarking language models, and scoring generative text summarization, contextualizing our thesis within existing literature. In Chapter 4, we discuss the selected datasets used for our evaluation pipeline. Chapter 5 describes the methodology taken in this research, including efficient transformer techniques, context window extension methods, dataset selection, and the evaluation pipeline. Chapter 6 presents the results of our evaluation pipeline. Chapters 7 and 8 conclude this thesis by summarizing the key contributions of the thesis, acknowledging limitations, and outlining directions for future work.

2 Background

To provide a comprehensive understanding of how modern LLMs process and understand text for summarization and question-answer purposes, this chapter will highlight the success of LLMs and the transformer architecture and discuss NLP applications of LLMs. It will explore text evaluation metrics, such as similarity metrics, and subsequently discuss techniques for contextual data representation and information about model prompting.

2.1 Automated Text Processing

2.1.1 Text Summarization

In today's digital age, we face an overwhelming flood of textual information that exceeds the human capacity to manually read and comprehend. This reality has created an urgent need for automated text summarization systems. These systems serve a crucial role by automatically identifying the most salient features of a text and condensing them into concise summaries that maintain the essential meaning and overarching narrative of the original content. Generative text summarization enhances human understanding of various domains and enables effective processing of large amounts of information with reduced reading time and cost restrictions. The literature commonly divides into extractive and abstract text summarization [4]. Extractive summarization techniques identify the sentences that hold the most relevant information of the text according to a specified sentence scoring technique, like TF/IDF [5], or TextRank [6]. Abstract summarization methods diverge syntactically from the source material while aligning semantical key features of the text by generating novel sentences that rephrase and condense key concepts and ideas, eliminating redundancies. As noted by Shakil et al. [7], abstract summarization techniques produce higher-quality summaries by creating more cohesive representations of the original text than extractive methods, which often string together unrelated sentences. Shakil et al. defines a high-quality summary as a concise and coherent representation of the source material. It should incorporate relevant and accurate information from the text, omit redundancies, and maintain the original style, tone, and format.

2.1.2 Question Answering

Answering questions about long documents and retrieving factual data or key concepts requires inferring knowledge over the entire length of a document. Similarly to text summarization, question answering (QA) offers opportunities for automation using NLP techniques.

Traditional, rule-based QA approaches rely on decision trees, lexical matching, and semantical heuristics for representing the syntactic and grammatical text structures that encode the contextual meaning of the documents [8]. Such techniques have shown to be proficient with already encountered data; however, they fail to adapt to unseen data that require learning new rules [8]. Statistical approaches like support vector machines (SVM) or Bayesian classifiers enable learning inherent textual data structures, paving the way for machine -and deep-learning solutions for QA tasks [8]. Approaches likePDFTriage [9] use the inherent structure of a document to answer questions based on the retrieved context. It simulates the reader's perception of the text structure by making use of structural metadata about the text. However, this approach necessitates the input text to be highly structured, which is unsuitable for many use cases that comprise inferring abstract texts that span long distances. Learning-capable models, most prominently LLMs, are successfully deployed to tackle more abstract questions, but they are limited by their context window size when facing longer documents. Particularly, answering questions about long novels containing long-range dependencies requires a focus on extending the context window and gaining an understanding of the entire document. One common approach to circumvent this limitation is retrieval augmented generation (RAG), which first retrieves relevant information from a large corpus (e.g., Wikipedia) and conditions its generation on retrieved information [10].

2.2 Large Language Models

2.2.1 Statistical and Rule-based Models

The mathematical foundations from what later emerged LLMs were established in the early 20th century with Markov Chains, introducing probabilistic modeling for predicting sequential data, and N-gram prediction mechanisms, which popularized statistical language modeling [11]. Rule-based approaches focused on identifying language patterns and structures but proved insufficient for adequately learning and understanding language structures and nuances.

2.2.2 Neural Networks

Neural networks progressed from static modeling systems to a learning-capable approach that enabled systems to dynamically generate text [12]. Recurrent neural networks (RNNs), the first iteration of a successful neural network, enable context-aware sequential processing of longer texts, allowing for more flexible and generalizable language modeling [13]. Another example of a learning-capable approach is Long Short-Term Memory (LSTM), which facilitates long document understanding, allowing the retention of information with memory cells and gating mechanisms [14] However, these approaches face difficulties when processing large amounts of data, encountering high memory and computing costs and insufficiencies in retaining long stretching causal relationships [11].

2.2.3 The Transformer

Ashish Vaswani [15] introduced the now-famous transformer architecture for neural networks. The transformer tackles the challenges of high processing costs and lacking long-range text understanding faced by RNNs and LSTMs. It excels at understanding long-form textual data, enabling a holistic language understanding. The most integral part of the transformer is its attention mechanism, which allows it to encode contextual information into word embeddings. The meaning of words in a text largely depends on their relationships with their surroundings and words from different parts of the text; the attention mechanism quantifies these inter-word relationships. The transformer considers information inherent to a word itself combined with information about where the word exists in the context of other words. The attention mechanism assigns higher weights to words that are semantically relevant to each other. This weighting process occurs when the attention score, calculated from the interaction between query and key vectors, determines how strongly one word's value vector influences the representation of another word. The weighted sum of the value vectors then forms the updated, context-rich representation of the word. These computations allow the transformer to understand contextual nuances in a learning-oriented manner. Unlike the inherently sequential architecture of RNNs and LSTMS, the transformer simultaneously computes its contextual representations for each word. This design allows for massive parallelization on compute-efficient hardware, explaining the wide adaption and success of the transformer [15].

2.2.4 Pre-trained and fine-tuned Models

Through pre-training, transformer-based models can develop a broad understanding of language by learning patterns, structures, and relationships from large-scale language corpora [16]. Fine-tuning can help adapt this generalized language understanding to specific tasks, transferring the knowledge acquired from the model's training to infer new knowledge in task-specific scenarios [17]. Instruction tuning directs the model output to desired characteristics by training the model on instruction-output pairs, enabling more effective performance on targeted applications such as text summarization and making LLMs highly practical for real-world scenarios [17].

2.3 Efficient Transformers

The advantage of transformer-based models is their ability to process information in parallel. However, the transformer faces runtime issues and computational challenges in scenarios with increasingly longer input data. In particular, its self-attention mechanism has a time complexity of $O(n^2 d)$, where n is the sequence length and d is the embedding dimension, due to the computation of pairwise attention and the subsequent weighted sum of values [18]. Therefore, the basis for processing long textual data is improved transformer iterations that handle data more efficiently, reducing memory and computational requirements of the attention mechanism. Tay et al. [19] categorizes the modification of the attention layer into the following types: **Fixed Patterns (FP)** statically restrict the scope of the self-attention

mechanism with pre-determined patterns, **Combination of Patterns (CP)** synergize multiple fixed patterns, **Learnable Patterns (LP)** take a data-driven approach to recognize existing patterns of the token distribution of an input sequence, **Memory (M)** aims to optimize memory usage, **Low Rank (LR)** compress the attention mechanism to a degree in order to retain accuracy and save computational costs, **Kernel (KR)**, as well as LR, is a blockwise approximation of the attention mechanism, specifically the attention matrix. The following is an incomplete list of examples of the highlighted approaches:

- The **Image Transformer** [19] simplifies self-attention by reducing the transformer's attention span to local input sequence subsets. **(FP)**
- **Sparse Transformer** calculates self-attention on a reduced number of key-value pairs. **(FP+CP)**
- The **Longformer** [20] utilizes sliding window attention, which reduces the attention range for each token to a local subset, and dilated sliding window, which excludes parts from the receptive field, spreading the attention span farther over the sequence. $\mathcal{O}(n(k+m)$ $O(n(k+m)$ **(FP+M)**
- The **Linformer** approximates the self-attention matrix with a low-rank matrix reducing complexity to $\mathcal{O}(n)$ [21]. The method reduces the parameters inside the attention matrix, exploiting sparse patterns [22].
- **Reformer** groups similar word vectors with *locality-sensitive hashing (LSH)* allowing to compute attention for a subset of all key-value pairs, reducing complexity to with a complexity of $\mathcal{O}(n * log(n))$ [23]. **(LR)**
- **Mixture-of-Experts (MoE)** [24] divides a model into experts, sub-networks specialized in a subset of the training data. MoE models reduce computational overhead by activating only a limited number of experts for each input. Allows scaling up the model size while retaining computational cost [25]**(LR)**

2.3.1 Modified Self-Attention

An example of an adaptation of the attention mechanism is Grouped-Query Attention (GQA) [26], which is frequently utilized in open-source models such as LLaMA and Qwen [27] [28]. One of the main advantages of GQA is its ability to significantly speed up inference while preserving performance. By reducing the number of unique key-value pairs that are loaded during inference, GQA lowers memory usage, leading to substantial efficiency gains.

2.3.2 Hardware-Aware Transformers

Beyond architectural modifications to the transformer, the idea to improve the implementation side contributed significantly to more efficient alternations of the baseline transformer. *FlashAttention* [29] represents an approach to improve the efficiency of the transformer through

optimizing memory management. Traditional attention computations are limited by memory bandwidth and high activation memory usage, leading to inefficiencies on modern hardware like GPUs. FlashAttention addresses these issues using a tiling approach that reads and writes more frequently used data to high-bandwidth SRAM (shared memory) instead of repeatedly accessing slower DRAM. This IO-aware approach significantly reduces memory traffic and improves performance, enabling exact attention computation with lower memory usage. The authors demonstrate that FlashAttention achieves up to 3× speedup and reduces memory usage by up to 10× compared to standard attention mechanisms, making it particularly useful for training and inference of large language models.

2.3.3 Quantization

Quantization is a model compression technique that reduces LLMs' computational complexity and memory requirements by representing weights and activations with lower precision. Standard LLMs typically use 32-bit floating point representations of model weights and activations, but quantization reduces this to lower bit-widths down to 8 and 4-bit representations. Quantization can be applied at different stages of model development, primarily through Quantization-Aware Training and Post-Training Quantization [22]. Additionally, quantization is often combined with pruning, which removes redundant parameters to further optimize model efficiency [30].

2.4 Context Window Extension Techniques

Pre-trained language models typically have a context window length of around 4K–8K tokens, which can be limiting for tasks requiring long-context understanding [31]. One of three research questions of this thesis aims to identify the most effective techniques for extending the context window. Various modifications help mitigate the quadratic computational and memory requirements of self-attention, making extended context window inference more feasible [22].

2.4.1 Positional Techniques

Transformers require positional information to be injected into their architecture since the attention mechanism is inherently position-agnostic, meaning tokens are processed without considering their position within the text [32]. This addition is crucial for maintaining the sequential nature of natural language. Transformers can learn word positions within their limited training length in numerous ways during training. There are two main approaches to handling positional information: absolute and relative positional embeddings. The original transformer architecture implements absolute positional embeddings with sinusoidal functions [15]. These functions generate unique position vectors for each token added directly to the word embeddings at the bottom of the transformer. However, this approach has significant limitations when processing positions beyond the trained context length of the model. In this case, the sinusoidal values can lead to degraded output quality, lowering the

extrapolation and generalization capacity of the model to longer sequences. Furthermore, absolute positional embeddings treat positions independently from each other, failing to capture the intuitive notion that nearby tokens should have more contextual relevance than more distant tokens. In understanding language, the relative positioning of words is crucial. Within a transformer's architecture, relative positions can be computed through the differences between queries and keys inside the attention mechanism. The subsequent sections will discuss the prominent approaches for enabling long-context understanding by modifying the transformer's positional embedding stage.

The efficient baseline models can undergo architectural augmentation to accommodate extended context processing in NLP tasks. Substantial research has been conducted to take advantage of the intrinsic model characteristics to enhance contextual comprehension of transformers, resulting in a diverse spectrum of methodological approaches. Following the taxonomic framework proposed by Pawar et al. [33], contemporary context extension techniques can be classified into extrapolative and interpolative paradigms. Extrapolative techniques employ predictive algorithms to project beyond the initial pre-trained context boundaries, using various mathematical models to estimate token relationships in extended sequences [34]. On the other hand, interpolative approaches focus on enriching the existing context space by synthesizing novel data points within the unexplored regions of the pre-trained context manifold, effectively creating a more densely populated attention space [34].

ALiBi

Zero-shot context length extrapolation [33] refers to an LLM's natural ability to handle sequences longer than encountered during training without additional training or fine-tuning. Attention with Linear Biases (ALiBi) [35] is a positional extrapolation (PE) technique that harnesses this ability. It has proven to be proficient in generalizing sequences that are longer than those encountered during training. As compared to other PE schemes, ALiBi does not use positional encodings; it encodes position information by biasing the query-key attention scores proportionally to the distance between each pair of tokens [22]. The bias matrix applied to the attention is static and not learned [25].

RoPE

Rotary Position Embeddings (RoPE) [36] are an approach for injecting relative positional information into the transformer. It significantly improves sequence length flexibility and generalizes to longer context windows over previous iterations. RoPE applies a rotational transformation to the query, key, and value vectors based on their positions in the sequence. Greater rotations indicate later positions inside the sequence, and relative positions between words are preserved through rotation differences. Closer tokens with more significant contextual similarity applied to the RoPE rotation have a smaller dot product, indicating closer contextual alignment. The rotation angle observes a natural attention decay pattern where distant words yield smaller dot products [36]. [37] RoPE has been the de-facto standard

technique for various LLMs, i.e., the suite of Llama models [27], or models from Mistral AI [38].

xPOS

Positional extrapolation with solely RoPE-based models, however, is limited [37]. Extrapolatable Position Embedding (xPOS) [39] aims to generalize to sequences longer than those encountered during training. Adding an exponential decay component to the RoPE component, where the decay becomes smaller with larger basis frequencies, improves stability during training, mainly when modeling relationships between tokens that are far apart in the sequence [25]. They additionally incorporate a block-wise form of causal attention to enhance the model's capacity during the inference phase. Testing on language modeling tasks revealed strong results: the model achieved better perplexity scores on standard validation data and maintained its effectiveness when processing sequences that exceeded the length of its training examples.

Positional Interpolation (PI)

Instead of positional extrapolation, Positional interpolation deals with longer context windows by scaling new positional indices to the maximum range of the context window limit established during the pre-training phase [22]. (show this wavelength graph?) The idea is to directly downscale position indices that exceed the maximum context length, effectively populating the existing RoPE curve with data points of unseen positions [37]. Their analysis demonstrates competitive performance evaluated on text summarization tasks up to a context window length of 32k tokens with minimal additional training.

NTK-aware and NTK-by-parts Interpolation

RoPE transforms the one-dimensional positions of tokens into n-dimensional complex vectors, which are rotated inside a vector space [36]. In order to capture relative positional information across the entire context length, RoPE embeddings assign different frequencies to each dimension of the positional RoPE vector, which then maps to a sinusoidal function. Low-frequency components with long wavelengths capture long-range positional information, while high-frequency components allow fine-grained, precise positional distinction. The wavelength of the function can be understood as the length of tokens needed in order for the RoPE embedding at dimension d to perform a full rotation (2π) and produce duplicated values (copy) [40], short wavelength, i.e. high-frequency, its rotation cycles very fast, meaning token positions quickly start to look similar. Interpolating RoPE embeddings for longer sequences, as described in the corresponding section of (PI) involves linearly stretching all embedding dimensions based on the scaling factor, irrespective of their wavelength and frequency, to accommodate unseen values. For a target extension ratio, the rotation angles of all positions are linearly reduced across all RoPE dimensions. However, this makes the positional information very crowded, which leads to the loss of high-frequency information

since close positions are less distinctive due to the associated shortened wavelengths [32]. NTK-aware interpolation addresses this complication by scaling low-frequency components that represent long-range dependencies to a higher degree than high-frequency components, which retrain fine-grained positional information [40].

YaRN

Yet another RoPE extension method (YaRN) [40] builds upon the selective interpolation of NTK-aware interpolation of low and high-frequency dimensions [22]. YaRN groups the RoPE dimensions based on their frequency. It applies different extension strategies for each group of dimensions [41]. High-frequency dimensions undergo extrapolation, preserving granular information, and low-frequency dimensions use linear interpolation (PI). The non-linear interpolation function for positional embeddings maps the original position space to an extended space, helping stabilize attention scores across varying sequence lengths while preserving critical positional information and mitigating performance degradation at longer contexts [40]. YaRN models show improved performance on context windows of up to 32k.

LongRoPE

The proposition of LongRoPE [41] states that positional encoding should be handled non-uniformly. Previously discussed techniques like YaRN or PI scale RoPE embeddings based on an arbitrary human-designed set of rules. PI scales each dimension equally with a constant factor s; YaRN determines dimension scaling based on its frequency [40]. LongRoPE exploits prevalent non-uniformities and sets individual rescale factors according to each dimension. LongRoPE applies evolutionary search to find optimal scaling factors for each RoPE dimension, making the positional encoding operation non-linear across all dimensions [41]. Earlier token positions require less interpolation; finding the optimal configuration depends on the desired sequence length extension. Interpolation for high-frequency components is bad -> adjusting resale factors (mitigating performance on the original context window). By leveraging these non-uniformities, LongRoPE minimizes the information loss typically caused by positional interpolation. This approach minimizes information loss during interpolation, providing a more decisive starting point for fine-tuning. It enables models to handle sequences eight times longer than their original training length without additional training. LongRoPE leverages an efficient, progressive extension strategy to achieve up to 2 million token context windows without requiring direct fine-tuning on texts with highly long lengths, which are scarce and hardly available. Extensive experiments validate the effectiveness of LongRoPE. The method demonstrates superior performance in perplexity and retrieval tasks across various models, including LLaMA2 and Mistral. It maintains low perplexity across extended contexts and achieves over 90% accuracy in retrieval tasks, outperforming state-of-the-art alternatives like linear interpolation, NTK-aware scaling, and YaRN [41].

2.5 Similarity Metrics

Although evaluating text quality is inherently subjective, measuring it against widely accepted characteristics offers a clear and transparent framework for assessment and comparison. Human judgment is the gold standard for determining whether a text meets specific requirements, but it is infeasible for large-scale evaluations regarding NLP tasks. Supervised machine learning practices assume a reference summary that combines desired qualities on which a candidate generation can be measured. For text summarization, we have ground truth summaries that can be human-written, human-annotated, or generated by other LLMs. Automatically calculating the similarity of a candidate and a reference becomes vital for large sample sizes and generative models. The goal is to capture syntactic and semantic equivalences, ensuring the candidate represents the ground truth well.

2.5.1 Lexical Similarity

The similarity of two texts can be assessed at different levels of granularity, ranging from the token level to the sentence level. Perplexity is the earliest adoption of automatically evaluating model generations. It measures the quality of a text on a token-level alignment, making it compute efficient but limited in measuring semantic similarity as it considers a very restrictive view of the data, leaving out semantic nuances and overarching themes of a text. [42] *N*-gram-based metrics statically evaluate texts on a word level by measuring how many subsequences of *n*-words match between the candidate and the reference. Increasing the n reduces the flexibility of a candidate generation to express equivalent meaning using novel expressions, constraining it to the exact match to the reference. Measuring this lexical alignment along two dimensions, recall and precision [43] is standard practice. Recall measures how well a candidate captures all relevant features of the ground truth and thus indicates the diversity and completeness of the model output. Precision evaluates the accuracy of the candidate's proposed solutions, making it a measure of quality. A candidate text can have high precision, indicating that what it generates is correct but captures only a fraction of all relevant information, suggesting a low recall score. Therefore, we can combine the two measurements into one metric, the F1-score. It is calculated as the harmonic mean of precision and recall, balancing the two metrics. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a widely used framework for automatically evaluating the quality of text candidates [44]. It encompasses several measures, including ROUGE-N for n-gram overlap, ROUGE-L for longest standard subsequence matching, and ROUGE-S for skip-bigram co-occurrence, which refers to the number of pairs of words that appear in the same order within a certain distance in the text. One of ROUGE's main strengths is its capability to evaluate precision and recall in summary generation, effectively measuring linguistic coherence and content accuracy.

TF/IDF

Term Frequency/Inverse Document Frequency (TF/IDF) [5] measures the importance of terms within a text by assigning a weight to each term. TF measures how often a term t appears in a document d; a higher frequency indicates that the term is more significant within that specific document. IDF assesses the importance of a term across a corpus of N documents. Terms that appear in many documents are considered less informative and discriminative, while those that appear in fewer documents are deemed more significant. IDF computes the logarithm of the inverse of the fraction of documents containing the term (n_i):

$$TF = \frac{\text{occurence of } t \text{ in } d}{\text{number of terms in } d}$$

$$IDF = log\frac{N}{n_i}$$

$$TF/IDF = TF * IDF$$

TF/IDF can be used to measure the similarity of a candidate text to a reference text on a sentence or paragraph level. TF measures the importance of terms within a sentence or paragraph, and IDF measures how significant the term is considering the entire text. The weights TF/IDF assigns to each term form a vector representation of the sentences, which then can be compared with a similarity metric like the cosine similarity.

2.5.2 Contextual Similarity

Cosine Similarity

The cosine similarity [45] measures the similarity of two n-dimensional vectors (A and B) by calculating the cosine of the angle between the two vectors:

$$cos(\theta) = \frac{A * B}{\|A\|\|B\|}$$

Where $A * B$ is the dot product of the vectors and $\|x\|$ is the magnitude of the vectors. A cosine similarity score of 1 indicates that the vectors are perfectly aligned, signifying complete similarity. A score of -1 denotes that the vectors are oriented in opposite directions, representing total dissimilarity. A score of 0 implies that the vectors are orthogonal, indicating no similarity between them.

BERTscore

BERTScore [46] contributes a semantic sensitive similarity metric. It uses contextual token embeddings of the pre-trained BERT model, a bidirectional transformer-based encoder model for contextual text understanding [47]. It uses the sum of the cosine similarities between the token embeddings to determine the similarity of the candidate and the reference text.

Compared to lexical *n*-gram-based metrics, BERTscore can detect matching paraphrases and better capture distant word dependencies. Changing the order of words in a sentence can alter its meaning, especially cause-effect relationships that change meaning depending on their word order. While traditional n-gram models fail to catch these important differences, BERTScore is proven to successfully identify when word order changes affect meaning. BERTScore correlates highly with human judgment, allowing for a soft similarity measure instead of exact-string or heuristic matching.

Sentence Transformers

Sentence Transformers (SBERT) [48] employs a siamese and triplet network structure for computing text representations with semantic sentence embeddings. It adapts the pre-trained BERT encoder model and enables contextual similarity comparison of sentences with cosine similarity. This thesis will use the sentence transformer 'all-MiniLM-L6-v2' [49].

2.6 Model Prompting

Prompt engineering can significantly boost the inference quality of LLMs, and it can help align LLM-generated output to task and domain-specific characteristics [25]. It involves designing and optimizing instructions to produce accurate, relevant, and coherent model outputs [50]. The underlying principles for well-structured prompts lie in inter-human communication. Unambiguous and precise language helps the model reduce the risk of generating generic output, directing it to more specific responses. Giving examples or setting a scene with a role allocation for the model further improves the alignment to desired outcomes by providing context, expectations, and constraints that guide its responses more effectively [51].

Zero-shot prompting refers to the practice of providing a model without prior context for solving a specific task. This challenges the model to infer resolutions from its more or less generalizable knowledge learned during training. *One-shot* and *few-shot* prompting provides the models with one or more examples of how a specific task should be completed successfully at inference time. More sophisticated prompting techniques include *Chain-of-thought* (CoT) prompting that dissects logical tasks into intermediate problems, enhancing the transparency of how the model reasons over a given task.

3 Related Work

This chapter surveys different LLM benchmarks, exploring how these evaluation frameworks measure the performance of LLMs across different NLP tasks. This survey will guide the design of our evaluation benchmark later on. The chapter will subsequently present metrics dedicated explicitly to text summarization and question-answering tasks.

3.1 LLM Benchmarking

Various benchmarks have been developed to facilitate the comparison of different LLMs and determine the quality of the model. Since LLMs possess a wide range of capabilities, these benchmarks must assess multiple performance dimensions - from basic language understanding to complex reasoning, creative generation to factual accuracy, and task-specific proficiency to general knowledge application. Conventionally, many benchmarks evaluate models using a relatively small context window. However, as modern LLMs are increasingly required to process and comprehend longer texts, it is essential to develop benchmarks assessing their ability to effectively handle extended context windows. The evaluation of LLMs on long documents remains an especially active research area, offering valuable insights into how these models distribute their attention across extended contexts and retrieve and understand long-range dependencies. In particular, evaluating generative text summarization is crucial, as models must effectively process entire documents to generate comprehensive summaries. Various methodologies are used to test this capability. As LLMs continue to advance, researchers have developed numerous benchmark datasets to assess their performance across diverse tasks. The following sub-chapters will introduce notable benchmarks for evaluating long-document understanding, summarization, and question-answering tasks.

3.1.1 SCROLLS

Standardized Comparison Over Long Language Sequences (SCROLLS) [52] is a benchmark designed to evaluate LLMs on tasks involving long texts. The creators of SCROLLS curated a suite of datasets where inputs are naturally lengthy and require synthesizing information across the text. The benchmark tests LLMs on summarization and question-answering tasks across various domains, including literature, science, business, and entertainment. SCROLLS establishes consistent evaluation methodologies, using standard metrics such as ROUGE for summarization and F1-score for question-answering tasks.

3.1.2 LongBench

LongBench [53] is another example of a benchmark for long document inference. It comprises 21 unique datasets spanning six categories and includes texts in both English and Chinese. Along with SCROLLs, LongBench offers model evaluation with text summarization and question-answering tasks. Testing of eight popular LLMs, including GPT-4, Llama-2, and Mistral revealed that while commercial models performed better than open-source ones, all models struggled with very long contexts. They also experiment with techniques such as scaled positional embeddings and long-sequence fine-tuning to improve the understanding of long-context windows.

3.1.3 LEval

L-Eval [54] tests long-context understanding of models through summarization and question-answering tasks, covering open-ended and definitive questions across various domains, including law, financial records, and poetry. This benchmark stands out by using documents up to 200k tokens in length, significantly longer than other benchmarks. L-Eval points out that traditional n-gram metrics, including ROUGE, often fall short when evaluating long-form text and frequently disagree with human assessment. To address this, L-Eval implements AI-powered evaluation using advanced LLMs instead of conventional metrics. The authors of the paper test 16 different models with L-Eval, and concluded that commercial and open-source capabilities see a substantial difference, particularly in tasks requiring deep contextual understanding.

3.1.4 InfiniteBench

InfiniteBench [55] contributes a benchmark for extremely long context windows, exceeding 100k tokens for English and Chinese data. Using synthetic and real-world examples, it measures how well models maintain coherence and extract information from lengthy contexts. Initial results reveal that even advanced models struggle with such long sequences.

3.1.5 Lost in the Middle

Lui et al. [56] investigate how well LLMs use long context windows more closely. It utilizes multi-document question answering and key-value retrieval to assess whether models can attend to all parts of the input equally. They feed the models multiple documents, of which only one contains the necessary information to tackle the question or to complete the key-value pair. They control for the number of distractor documents and the position of the single relevant document to determine if the model performance depends on the context window size and the position within it. The evaluation reveals a U-shaped performance trend across the context window length for all tested models, including GPT-3.5, Claude-1.3, and Llama-13B. The models struggle to accurately utilize information positioned in the middle of the context window, performing better when the information appears at the beginning (primacy bias) or the end (recency bias). Expanding the context window and supplying the

model with more information presents a trade-off. While a larger input context can enhance performance by providing more relevant details, it also increases the amount of irrelevant content the model must process and stretches the limited attention span of the model, which may reduce accuracy.

3.2 Evaluation Metrics for Text Summarization and Question-Answering

Summarization Metrics Koh et al. [31] establish a framework for evaluating the quality of generative text summarization. Their paper assesses a summary along multiple dimensions, focusing on how well it captures the source material's core concepts while maintaining coherence and traceability. Traditional metrics like ROUGE and BERTScore measure lexical and semantic similarity between the generated summary and reference texts, assessing relevance and recall. Beyond these, extractive coverage and density metrics help trace back summary content to the original document, indicating how much of the summary is directly derived from the source. The study also evaluates the Informativeness of the summary by analyzing how well different sections of the original text are represented in the summary, using ROUGE-L scores to map summary sentences to corresponding sections.

Metrics for Question-Answering KPQA [57] evaluates question-answering. It focuses on key-phrase prediction instead of relying solely on traditional lexical overlap metrics like ROUGE. KPQA assigns different weights to each token based on its importance, ensuring that key meaning is preserved in the generated answer. The study found that KPQA correlates more strongly with human judgments compared to existing evaluation methods.

4 Datasets

This chapter discusses the characteristics of the selected datasets used for our evaluation pipeline.

4.1 Dataset Characteristics

For our evaluation tool, we utilized three datasets that adhere to characteristics favorable to extended document inference. We require datasets with long-range texts, little layout biases, and abstractive summaries. Each dataset is filtered to include only entries with at least 128,000 tokens and is evaluated based on the following four metrics introduced by Koh et al. [31]:

- The ***compression ratio*** is the document length divided by the summary length. It indicates how much of the original document is condensed to arrive at the dimensions of its summary. The ratio increases exponentially with the document length since longer texts contain increasingly more redundant information.
- ***Extractive Coverage*** and ***Density*** measure how much the summary directly extracts from the source document, revealing whether a model generates a summary by directly copying text chunks or by reorganizing and rephrasing content. A lower value of the two metrics suggests that the summary is more abstract than extractive and vice versa.
- ***Redundancy*** averages the pair-wise similarity score for each summary word. A higher redundancy score indicates more overlap in content between sentence pairs. The metric helps evaluate whether a summary is unnecessarily repetitive or efficiently conveys unique information in each sentence.
- ***Uniformity*** examines the distribution of important information throughout the source document. It calculates the normalized entropy of the positions within the document of the 20 most important summary words. A low uniformity indicates a structural information bias of the source document, whereas a high value suggests that the information is evenly distributed over the whole length of the document.

4.2 BookSum

BookSum [58] contains 405 plays, short stories, and novels with expired copyrights collected from the Gutenberg project. Each entry is divided into book, chapter, and paragraph level

splits, with an average length of 145.000, 6.600, and 220 tokens, respectively. Each split level is paired with a highly abstractive human-written reference summary containing an average of 1100 tokens.

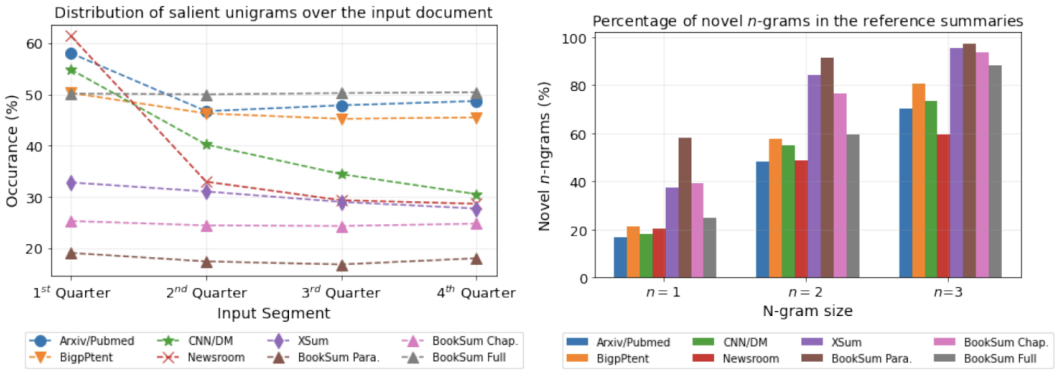


Figure 4.1: (Left) The illustration, taken from [58] of the salient unigram distribution of different datasets, measures how well relevant information is distributed across the document. BookSum displays an even distribution of information. (Right) Percentage of novel *n*-grams, highlighting the degree to which the reference summary abstracts from the input text.

4.3 IniniteBench

Similarly, the text summarization subset of the InfiniteBench [55] benchmark includes novels and play writes from five distinct domains, with an average length of 103,500 tokens. They employ *key entity replacement* to obscure the potential familiarity resulting from the training of language models to the novel which are publicly available online.

	id int64	context string	input string	answer sequence
0		"\n \"Since I can do no good because a woman, Reach constantly at something that is near it. --The M(...TRUNCATED)	The summary of the book is:	["Jennifer is an earnest intelligent woman who makes a serious error in judgment when she chooses to(...TRUNCATED)
1		"'Yes, of course, if it's fine to-morrow,' said Mrs Bronwyn. 'But you'll have to be up wit(...TRUNCATED)	The summary of the book is:	["The novel is fragmented into stream-of-consciousness contributions from various narrators.\n\n"T(...TRUNCATED)
2		"\nChapter 1\ny father's family name being Pirrip, and my Christian name Philip, my infant tongue (...TRUNCATED)	The summary of the book is:	["Kiara, a young orphan living with his sister and her husband in the marshes of Kent, sits in a cem(...TRUNCATED)
3		"I am by birth a Genevese, and my family is one of the most distinguished of that republic. My ances(...TRUNCATED)	The summary of the book is:	["In a series of letters, Shreya Lillie, the captain of a ship bound for the North Pole, recounts to(...TRUNCATED)
4		"PART ONE\nWhen he was nearly thirteen, my brother Carsen got his arm badly brokenat the elbow. When(...TRUNCATED)	The summary of the book is:	["Vance Chase lives with her brother, Carsen, and their widowed father, Ariel, in the sleepy Alabama(...TRUNCATED)

Figure 4.2: First five rows of the InfiniteBench dataset [59].

4.4 XSum

The third dataset XSum [60] compared to the previous two dataset contains much shorter news articles, on average 431 words. The articles are aligned with single-sentence, human-written, and non-extractive summaries. The dataset consists of 204.045 articles from various domains crawled from the BBC archive between 2010 and 2017. In order to comply with the long context setting of our evaluation tool, we concatenate a subset of articles until they reach a length of 128,000 tokens. The same process is applied to the summary sentences.

4.5 Dataset Statistics

The metrics shown in Table 4.1 that confirm that the datasets adhere to the characteristics listed above. With an average uniformity score of 78.3, the datasets have a even distribution of information across the entire document length. Additionally, the low overall values for CRT and CRS suggest a highly abstract nature of the documents and reference summaries.

Dataset	**BookSum**	**InfiniteBench**	**XSum**
CRT/CRS	19,0/18,1	237,73/274,43	17,78/17,76
EC/ED	0,03/0,19	0.012/0.06	0,06/0.35
Redundancy	0,10	0,11	0,10
Uniformity	0,70	0,75	0,90
Number of Rows	405	103	204045
Words per Entry	84.660	128.000	431,07
Words per Summary	875	825	23,26

Table 4.1: Dataset metrics: CRT=Compression Ratio Token-Level, CRS=Compression Ratio Sentence-Level, EC=Extractive Coverage, ED=Extractive Density, Redundancy, and Uniformity; after these metrics, we have statistics that depict the dimensions of the datasets.

5 Methodology

This chapter will present the methodology used to answer our research questions and present the implementation of our evaluation pipeline. We outline the specifications of the evaluation tool that aims to provide a comparative performance analysis of various models that utilize the discussed techniques for extending their context window.

5.1 Evaluation Pipeline

The primary focus of this thesis is to examine how LLMs leverage their context window for text summarization and question-answering tasks based on their implementation details and the volume of input data provided. The preceding literature review of contemporary context window extension techniques in Chapter 2.4 provided an orientation within the dynamic landscape of many emerging architectural approaches for enhancing long context window understanding. In subsequent chapters, we highlighted the significance of testing the performance of LLMs. We conclude that the benchmarks presented in Chapter 3.1 are insufficient to test LLMs in particularly long context windows of up to 128.000 tokens. In order to meet the importance of these points, we will select a suite of four models, representing one context window extension each, and compare these models with our evaluation tool for long document understanding on three tasks, which include text summarization and question-answering. The following three points introduce our proposed evaluation metrics:

- **Positional Information Analysis:** The first metric is a granular examination of the model's text summarization ability. It examines the extent to which the model can allocate its attention across the full length of its context window, determining what information it prioritizes for summarization depending on the context window size.
- **Atomic Fact Extraction and Validation:** Secondly, we implement a strategy for evaluating the factuality of the model summary. We implement a fact decomposition algorithm, which segments the summaries into factual units. The metric analyzes the change of information depending on the size of the context window. We want to discern where the model derives new information and loses information when we prompt it with a longer context.
- **Question-Answering:** Systematically evaluates the question-answering performance across variable context window lengths, measuring the model's comprehension accuracy and response consistency.

5.2 Pipeline Functionalities

In this chapter, we introduce the core functionalities of our evaluation tool, designed to provide a comprehensive and flexible framework for analyzing model performance.

- **Incremental Data Loading:** The pipeline should facilitate the analysis of performance trends of different models along the length of a variable input sequence. We implement an incremental data loader for prompting the models with input sequences from 10.000 to 128.000 tokens.
- **Experiment Modularity:** The tool should allow users to fine-tune experiment configurations with granular control. This flexibility enables in-depth exploration of model capabilities, allowing users to identify strengths and weaknesses effectively.
- **Model Modularity:** The user should be able to upload custom models and adjust those settings according to individual use cases, allowing for a detailed analysis of the strengths and weaknesses of their different models.
- **Evaluation Metrics:** The pipeline should provide in-depth insights into how models utilize their context window size using distinct evaluation metrics.
- **Data Visualization:** Users should benefit from thoroughly visualizing results across various configurations, enabling a visual comparative analysis.

5.3 Pipeline Design

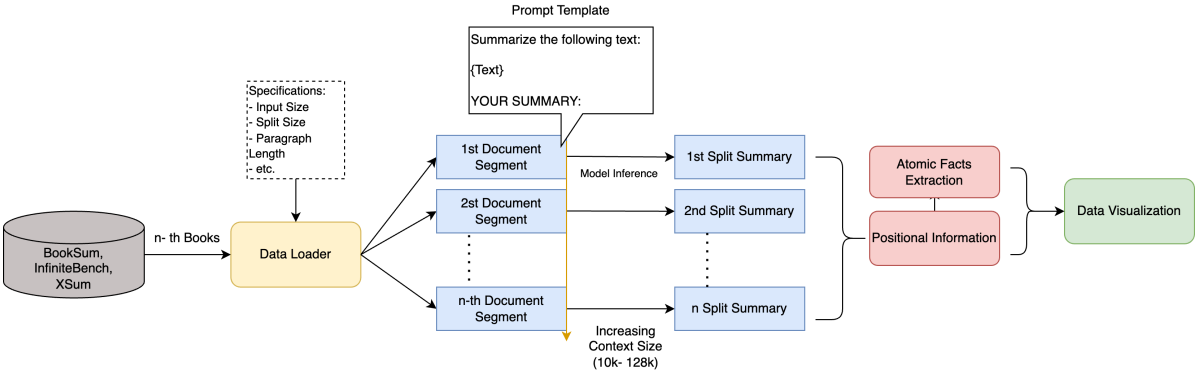


Figure 5.1: This flowchart visualizes the underlying structure of the summarization tasks. It feeds the entries of the datasets through the data loader, which splits them into chunks that are incrementally longer by 10.000 tokens. We then run a summarization inference with the model for each book split. We subsequently retrieve positional information from each model summary and use it for the positional and factual analysis.

The design of our pipeline aims to assess LLMs according to two dimensions:

- The impact of context length on performance - does providing more context enhance or degrade the model's accuracy?
- The influence of information placement within the context - does the model exhibit a bias toward certain sections of the input text?

To explore these factors, the data loader component divides input documents into thirteen segments, increasing in length by 10,000 tokens, ranging from 10,000 to 128,000 tokens. Next, we prompt the candidate model to summarize each document segment. We then extract positional information from the summary derived from each document segment, creating a mapping that links summary sentences to their corresponding positions in the book. For the first evaluation step, we analyze the distribution of the model's attention throughout its context window using this positional information. This analysis reveals whether the model observes positional biases toward a specific part of the input sequence or whether it can evenly distribute its attention over the context window. The subsequent second metric extracts the information in the summary and analyses how it changes with increasing context lengths by comparing two summary pairs from different document segments. We visualize the structure of the pipeline architecture for the first two metrics in figure 5.1. For the third metric, the question-answering task, we split each document segment into three parts; for each part, we generate question-answer pairs of four question types using GPT-4o-mini. The models then answer the questions, and we measure their similarity to the generated reference answers. Figure 5.2 illustrates this part of the pipeline. The pipeline runs these experiments on each selected dataset and visualizes the results through graphs.

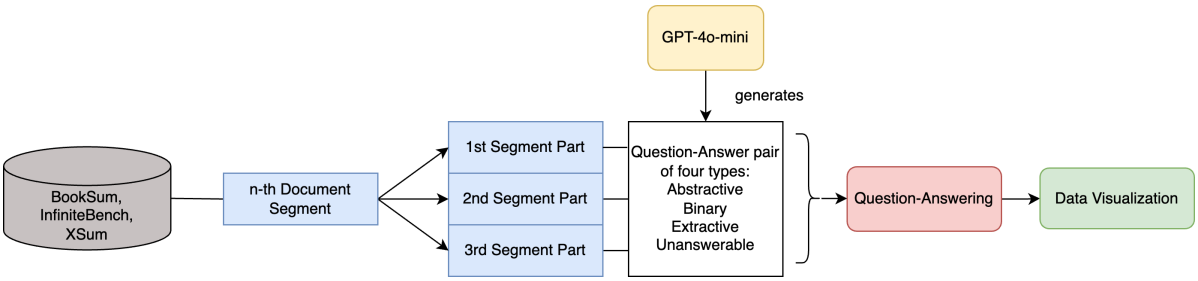


Figure 5.2: This figure illustrates the question-answering component of the pipeline. We split each book into three parts; for each part, we generate question-answer pairs using GPT-4o-mini of four question types. The candidate model is then prompted to respond to these questions, and the similarity between its answers and the generated references is measured. The results are then visualized.

Full modularity of the pipeline We enable users to configure their custom models and experiment setups. The pipeline employs a modular approach for benchmark initialization. The model configuration includes adjusting parameters that influence the tone of the output, such as setting the temperature and top_p, as well as specifying the maximum allowed tokens for each response. This flexibility allows users to tailor the model's behavior to suit their

specific needs and preferences, ultimately enhancing the overall effectiveness of the pipeline. The experimental setup can be adjusted by controlling for the maximum context window size, the number of books, or the similarity metric, choosing between tf/idf and sentence transformers used to evaluate candidate generations.

5.4 Data Loader

Our evaluation framework uses a data loader that processes books by dividing them into segments of incrementally larger sizes. These segments progressively fill the models' context windows, reaching up to 128,000 tokens. We test the model performance depending on the size of the segments. This approach allows us to:

- Test how effectively models utilize their maximum context capacity and test for inherent context window biases.
- Evaluate information processing quality as the input size grows and identify potential performance degradation points

The data loader allows us to assess whether more information enhances or diminishes the quality of generated summaries. This method effectively reveals the practical limits of each model's context understanding and processing capabilities.

5.5 Model Prompting

The pipeline describes constraints for the different models to ensure uniform output. We instruct the models to generate suitable summaries that align with the length, scope, and tone of the reference summaries. Due to the immense informational scope of our documents, the models tend to generate extensive summaries, often losing focus on the main points and discussing minor unimportant details of the document. Including a statement about the desired length—specifically, the length of the reference summary—of the model summary in the prompt alleviates this problem to a great extent, leading to more concise and holistic summaries. Additionally, we instruct the model to generate continuous summaries, restraining them from using bullet points of salient facts of the book. These principles of conciseness also apply to the question-answering task. This enhanced control over the model outputs facilitates a meaningful comparison of consistent summaries between different models. In practice, however, we cannot control for all deviations. A common challenge across all models is adhering to the maximum output length specifications, which can sometimes lead to incomplete summaries. Even with the instructions in the prompt, some models may summarize content chapter by chapter, resulting in redundancy and unnecessarily lengthy outputs. We apply zero-shot prompting for the text summarization and few-shot prompting for the question-answering, giving examples for the ideal model performance.

5.6 Positional Information Analysis

The first of the three evaluation metrics in our pipeline is the positional analysis of the model summaries. We perform a thorough positional analysis to identify the source of the information in each model summary. The summaries are divided into individual sentences, while the books are segmented into consistently sized paragraphs. Next, we establish connections between summary sentences and their source locations by calculating the similarity of every pair of summary sentences and document paragraphs. This mapping uses three complementary similarity metrics:

- **TF-IDF (2.5.1)** measures text similarity based on word frequency-based.
- **Sentence Transformer(2.5.2)**: We use 'all-MiniLM-L6-v2' [49] to compute semantic similarity through the cosine distance between contextual text embeddings.

We also evaluate the reference summary against the complete text to establish a baseline. The result is a distribution map showing where each summary sentence likely drew its information from in the source text, revealing the focus of the model's attention over its context window.

5.7 Atomic Facts Evaluation

The second metric evaluates the correctness of a summary by extracting and scoring information from it. This approach uses FActScore, which is a fact-extraction method for autonomously evaluating.

FActScore FActScore [61] approaches the evaluation of model summaries from the perspective of factual accuracy. Determining the factuality of a text often relies on human evaluation, which is not scalable for large NLP tasks. To address this issue, FActScore introduces an evaluation methodology that aims to automate and approximate the accuracy of human evaluation. The approach leverages powerful LLMs to extract *atomic facts* from the model summary, which it then validates against a knowledge source, such as the Wikipedia corpus. An **atomic facts** is a concise sentence that conveys a single piece of information [61]. The authors claim that FActScore classifies only less than 2% of all facts as either correct or incorrect falsely. For the approach to work, the authors require the data to be *objective*, containing *specific* information; thus, they use biographies that adhere to those requirements. The study concludes that the majority of LLMs produce summaries that are significantly less factual than human-written summaries.

5.7.1 Adapted Fact-scoring

We utilize the fact extraction methodology of the preceding approach for our atomic facts evaluation. Our metric evaluates where models obtain the information from the source

He made his acting debut in the film The Moon is the Sun's Dream (1992), and continued to appear in small and supporting roles throughout the 1990s.
- He made his acting debut in the film.
- He made his acting debut in The Moon is the Sun's Dream.
- The Moon is the Sun's Dream is a film.
- The Moon is the Sun's Dream was released in 1992.
- After his acting debut, he appeared in small and supporting roles.
- After his acting debut, he appeared in small and supporting roles throughout the 1990s.

Handsome and intelligent, but neurotic and mercurial, Raskolnikov spends his days sleeping in run-down quarters and dodging his landlady, to whom he's deeply in debt.

- Raskolnikov is handsome and intelligent.
- He is neurotic and mercurial.
- He spends his days sleeping.
- He lives in run-down quarters.
- He avoids his landlady.
- He is deeply in debt to his landlady.

Figure 5.3: Illustrates the two different approaches for prompting the extractor LLMs GPT-4o-mini. On the right, we have an example of the original prompt template from FActScore featuring a sentence from a biography containing factual information. On the left, we have an example from our prompt template, a more abstract sentence requiring more nuanced information extraction.

document depending on the context window size and whether generated summaries present verifiable claims that align with the original text's context. FActScore employs few-shot prompting to extract atomic facts sentence-wise. The extractor LLM is prompted with multiple sentences and their associated facts as examples for a successful fact extraction. We adapt this prompt template to better suit our use case of more abstract text forms. As a result of this modification, we notice a slight increase in fact accuracy, as determined by our fact-scoring approach, which will be discussed later in this section. Figure 5.3 shows two examples for the original prompt sentence and the modified, more abstract iteration. We assume that this modification and the fact that we do not use the fact-scoring technique of FActScore mitigate the effects of our data's abstract nature on the process's factuality. Since FActScore uses the dated InstructGPT as their fact extractor, we changed to the more recent GPT-4o-mini for extracting factual information from the model summaries.

Instead of using the FActScores scoring technique, we employ the following two distinct custom scoring processes:

- First, we use a matching process to trace each extracted summary fact back to the paragraph from which it is derived in the source text. We use the positional mapping from our first evaluation metrics from Chapter 5.6 for this purpose. The highest

similarity to the facts extracted from the located document paragraph then determines the accuracy of the summary fact.

- Secondly, we score the model summary according to the extracted atomic facts from the reference summary to match each to the best-fitting summary fact, which we then use as the score.

These two approaches quantitatively measure how well the source text supports each atomic fact from the candidate summary.

5.7.2 Gradual pair-wise comparison

In addition to the fact-scoring, we want to analyze how each incremental expansion of the context window changes the information contained in the summary. We perform a pairwise comparison of two model summaries as the context window expands, with the first summary generated from a prompt containing 10,000 fewer tokens than the second. We leverage the positional mapping established in Chapter 5.6 to locate each summary fact's origin. With this comparison, we can identify how the information contained in the summaries changes depending on the context window size. We classify all facts from the two summaries into three categories based on the pairwise *cosine similarity*. We first compared each fact from the first summary to each of the second, and then vice versa:

- A fact is considered a **matching** fact if its similarity score with any fact in the other summary exceeds a similarity score of 0.7.
- If no fact crosses this similarity threshold, the fact is designated a **lost fact**.
- Similarly, when comparing facts from the second summary with those in the first, a fact is deemed a **new fact** if no fact in the first summary has a similarity score greater than 0.7.

This comparison explains where the model maintains, acquires, or loses information as their context windows expand, offering insights into its information processing capabilities across varying input scales.

5.8 Question Answering

Our third task evaluates the model's proficiency in responding to questions concerning a given document. We draw inspiration from the method proposed by Hilgert et al. [62], which involves question-answering across four distinct types of questions about scientific papers. The study delineates four distinct categories of questions designed to assess various dimensions of the model's capabilities: abstract (A), binary (B), extractive (E), and unanswerable (U).

- **Abstract questions** require the model to freely infer information and relationships from documents and to generate novel expressions in its responses. This introduces difficulties in maintaining factual accuracy, ensuring coherence, and preventing hallucinations.

- **Extractive questions** require LLMs to locate and directly retrieve relevant information from the given context without altering its wording. This presents challenges such as accurately identifying the correct span of text, distinguishing between closely related but distinct details, and handling cases where the exact answer is scattered across multiple sentences.
- **Binary questions** questions require LLMs to accurately interpret the given context, determine factual correctness, and provide a clear affirmative or negative response. We additionally prompt the model to give a reason for their decision to answer yes or no.
- **Unanswerable questions** require LLMs to recognize when the necessary information is absent from the given context and confidently refrain from generating an incorrect or speculative answer. We can test whether the model is prone to hallucinations when no relevant information is available.

We generated these questions with GPT-4o-mini, which prompted us to craft a question based on an excerpt from the book and the corresponding answer, which serves as the ground truth reference. For additional context, we provide GPT-4o-mini with the reference summary of the book, which helps it to phrase better-informed questions that relate to the key features of the text.

The positional analysis for this metric hinges on the location within the book from which the information required to answer each question is derived. We divide each document into three sections: the beginning, middle, and end. We devise questions of the four types for each document split. This approach allows us to examine for which position question position the model performs best.

5.9 User Interface

To ensure a seamless interaction with our evaluation pipeline, we develop a user interface that connects to a remote machine—specifically, the LRZ cluster—and enables the metrics to be run on custom models and different datasets downloaded from Hugging Face. The modular design of the UI allows users to specify individual experimental setups. Additionally, the user can compare their models' results to those from experiments we conducted on four different LLMs.

5.9.1 Data Visualization

We provide graphs for visually comparing model performances for each of our evaluation metrics.

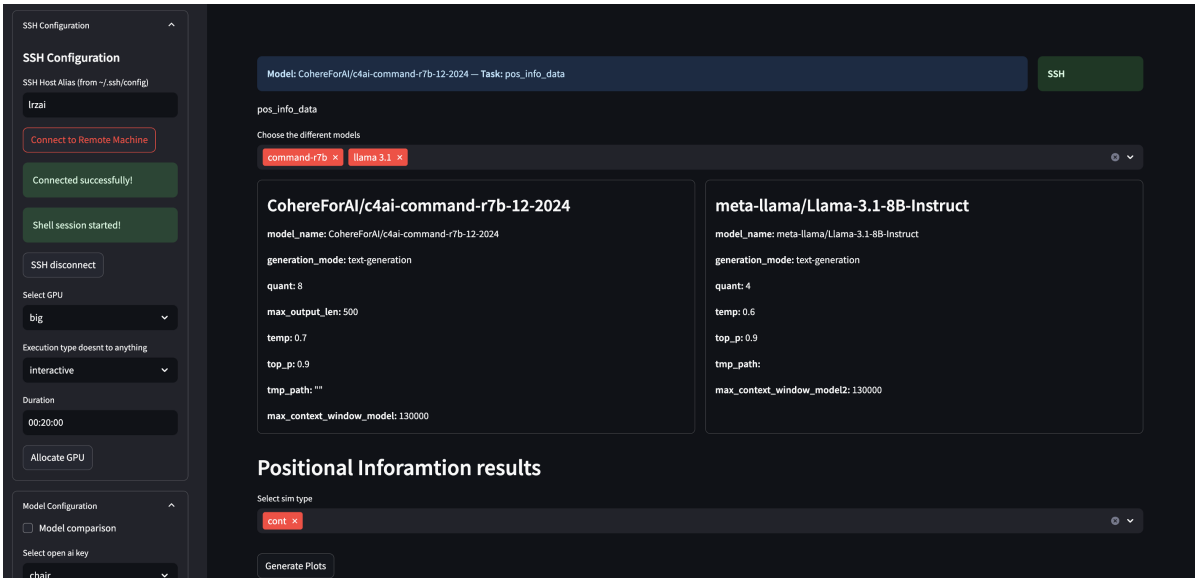


Figure 5.4: Displays the Configuration fields of our demo app for our evaluation tool, where we can compare the model parameters and the performance.

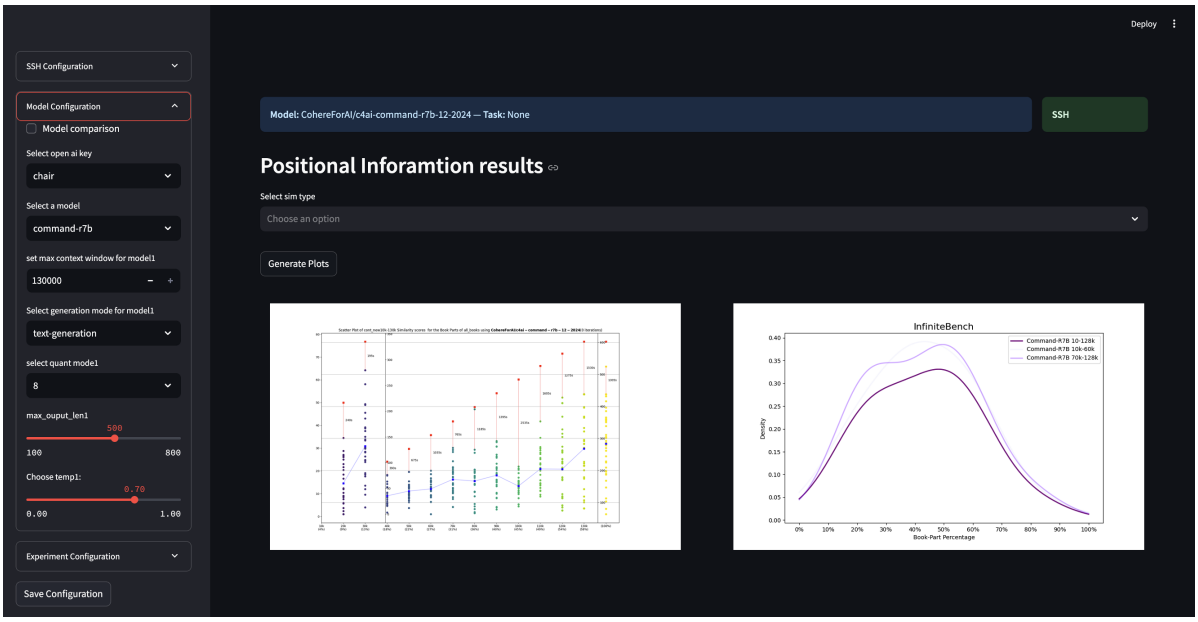


Figure 5.5: The visualization tool provides an easy way to compare different models, experiments, and datasets. This figure displays how our UI allows users to configure custom models and visualize and compare their results.

5.10 Model Selection

For a comprehensive analysis, we test three open-source models and one proprietary model as reference performance. In order to examine the impact of different context window extension techniques on the model's performance, we select our models so that each represents a different extension technique. In order to have comparable results, all models are of the same size, ranging between 7 and 8B parameters, and have the maximum context window size of 128000 tokens. In the following sections, we offer a detailed description of the chosen models.

- **Cohere Command-R7B** [63], the only proprietary model in our comparison is a very new language model, featuring a hybrid attention mechanism combining sliding window attention and a global attention layer without positional embeddings, allowing efficient local and long-range context modeling. The model uses RoPE for relative positional encoding in sliding window layers. It undergoes supervised fine-tuning and preference training to align with human preferences.
- **LLaMA-3.1-8B** [27] is a widely used open-source model that employs Grouped-Query Attention (GQA). For handling token positions, the model implements an iteration of RoPE that increases the base RoPE parameter from 10,000 to 50,000, which performs better for extended context window sizes. The training process of LLaMA-3.1 8B utilized diverse, high-quality datasets for pre-training and instruction-following tasks.
- **Qwen2.5-7B** [28] features advanced attention mechanism, positional encoding, and training procedures. Qwen2.5 implements Grouped-Query Attention (GQA), which optimizes key-value (KV) cache usage during inference. The model uses YaRN to extrapolate from its pre-trained context window.
- **Phi-3-small** [64] implements LongRoPE for positional encoding, which enables the model to extrapolate its pre-trained context window.

6 Results

This chapter will present the results from our evaluation pipeline, which includes four models of equal size, as chosen in section 5.10, and three datasets. We examined the model performances using prompt sizes of up to 128,000 tokens.

6.1 Positional Information Analysis

As outlined in chapter 5.6, we investigate where each model sources the information it incorporates into its summary of the book. For this purpose, we traced the informational origin of each summary sentence back to the book's most similar paragraph. For the evaluation, we split the input book that is to be summarized by the model into three parts. We count the number of sentences placed within each prompt split and calculate its percentage relative to all summary sentences. This way, we can determine which parts of the context window the models attend to most. We want to see whether the models can distribute their attention evenly over the whole context window or if they observe positional biases, focusing on the beginning, the middle, or the end and leaving out information from other parts of the prompt. As described in chapter 4, we assume that salient information is evenly distributed throughout the entire length of each document in our dataset collection; we, therefore, define the performance of an ideal summary as one that incorporates information from all sections of the document. To see whether the positional distribution changes with the length of the book, we gradually increase the number of input tokens up to 128k tokens, resulting in a total of 13 summaries for each dataset entry. We evaluate results for all three datasets, InfiniteBench, BookSum, and XSum.

The following subsections will provide a detailed analysis of the results for each model. To highlight positional dependencies, we evaluate performance across the beginning, middle, and ending sections of each document separately. We look at the relative contribution of each part to the total number of sentences in the summary of the model. Additionally, we compare these proportions along the increasing context window. This involves evaluating the distributions of summaries generated in the initial six prompt steps, which cover 10k to 60k tokens against those from later steps, spanning 70k to 128k tokens. This approach enables us to identify potential shifts in positional distributions as context length increases.

6.1.1 Command-R7B

Command-R7B has the worst performance of all models in the last third of the context window over all datasets. On average, only 6.8% of all sentences of the model summary

retrieve information from the ending section of the book. For InfiniteBench and BookSum, Command-R7B evenly distributes its attention over the first two prompt thirds, with, on average, 46% of all sentences tracing back to these parts. The attention distribution of XSum is much more skewed toward the starting section, with 77.1% of all sentences than the other two datasets. This extreme value constitutes an outlier with nearly 10 percentage points more than the subsequent most populated third over all datasets and models. Increasing the context window size significantly changes the positional distribution over the length of the prompt. The first book part shows an increase of 38.6%, while the second and last parts decrease by 39.3% and 36.5%, respectively. We conclude that Command-R7B faces challenges retrieving information from later parts of the document if the context window increases to 128k.

Model	**InfiniteBench**			**BookSum**			**XSum**		
	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
CommandR7B	**44.3**	**46.4**	9.3	52.2	42.2	5.6	82.0	10.0	8.00
	45.35	-31.27	-6.46	50.64	-36.97	-30.86	19.79	-49.67	-72.23
LLaMA-3.1	20.4	64.5	15.1	21.9	70.7	7.4	**39.6**	**47.8**	12.6
	-39.09	29.42	-32.8	10.32	13.08	-80.24	-55.69	110.43	-7.39
Phi-3	27.5	61.5	11.0	26.9	68.2	4.9	56.4	31.7	11.9
	-22.3	11.44	0.79	-27.29	11.41	43.37	-24.01	141.47	-51.85
Qwen2.5	17.5	58.9	23.6	21.6	57.5	20.9	26.2	37.1	36.7
	3.02	-15.41	49.05	-40.16	-1.8	74.61	-91.59	14.6	271.31

Table 6.1: The first three values for each model represent the percentage of located positions of all positions in each third of the context window size. The second row of numbers shows the percentage differences between the first half of the 13 summaries, which are derived from the context windows from 10k to 60k tokens, and the second half (70k-130k). We calculate this difference for all three parts of the context window. Since we desire an even distribution, extreme values represent negative performance indicators.

6.1.2 LLaMA 3.1

The positional distribution of LLaMA 3.1 is centered around the second third of the prompt for each data set. An average of 62.9% of all summary sentences originate from the middle section of the book, which is the highest value of all models in this section. Consequently, LLaMA 3.1 neglects the information at the beginning and end of the prompt to a greater extent. The starting third contributes 24.8% of all the information in the summary, which is a comparatively good value. However, the third end contributes only 12.3%, indicating that LLaMA 3.1 has the most difficulty with the late parts of the context window. Increasing the prompt length from 10k to 128k tokens changes the distribution over the prompt thirds. The overall evolution of the distribution tends toward the first third of the prompt, which

observes an increase of 51%. At the same time, the ending part suffers the most significant loss of 40.1%.

6.1.3 Qwen2.5-7B

The performance in the last sector of Qwen2.5 is overall the best compared to the other models. Qwen2.5 is able to retrieve, on average, 30.1% from the last third, with an outlier of 44.9% for XSum. The worst performing sector, with 19.3% on average, is the first third, suggesting that the model has a slight recency bias toward later positions in the context window. As with all other models, the middle sector is the most performant overall. As the number of input tokens gradually rises from 10k to 128k tokens, we notice a stark trend toward the last part of the context. It grows by 61.8% for BookSum and InfiniteBench and by a substantial 271.31% for XSum. We observe a coinciding loss of attention, particularly in the first part of the context window, of which the proportion decreases by 42.9%.

6.1.4 Phi-3-small-7B

Similarly to LLaMA 3.1, Phi-3-small extracts its summary information primarily from the early positions of the prompt. The first two sections share most of the sentence positions. The summaries for InfiniteBench and BookSum allocate only 8.5% to the first section while assigning 26.35% and 65.15% to the first and second sections, respectively. Likewise, for XSum, Phi-3 neglects the last section but distributes more evenly over the first two sections. The evolution of the distribution over the context window extension to up to 128k tokens shows an influx of 11.4% for InfiniteBench and BookSum and a significant increase of 141.5% for XSum. This has the consequence that the two outer parts of the context window receive increasingly less attention from Phi-3.

6.1.5 Overall Performance Trends

We conclude that all tested models deviate from an ideal informational distribution and observe recency or primacy biases. All models, except Qwen2.5, tend to favor earlier positions, with a strong preference for the middle and, to a lesser extent, the first section of the context window. LLaMA-3.1 centers around the middle section the most, and consequently neglects earlier and later parts of the context window. Qwen2.5 shows the most significant emphasis on later text positions. Phi-3, much like LLaMA-3.1, focuses its attention on the middle section; however, it is better able to also incorporate information from the other two parts.

6.2 Atomic Facts

The second evaluation stage of our benchmark tests whether the models can extract accurate information from all parts of its context window, depending on the prompt size. As for the other task, we gradually increase the context window size from 10k to 128k tokens and extract atomic facts from each summary sentence. This inference procedure results in 13 summaries.

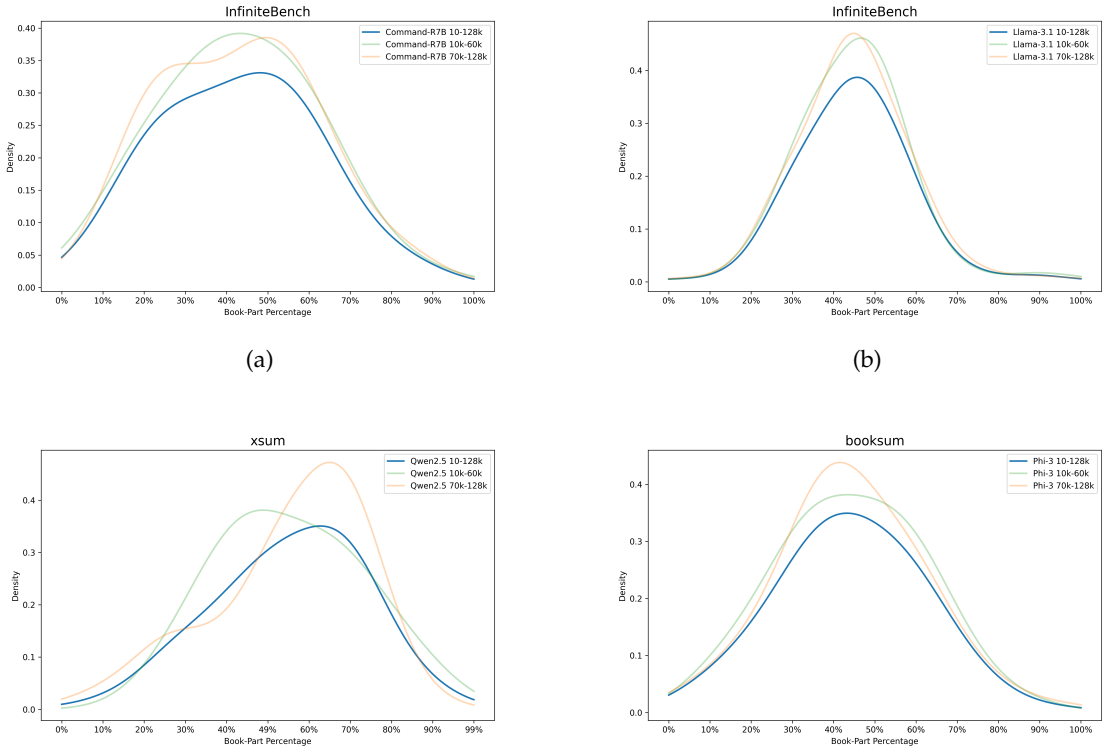


Figure 6.1: Displays the average density curves of the distribution of the retrieved information over the lengths of the documents. The three curves in each graph show the distribution over all prompt parts, parts from 10k to 60k, and parts from 70k to 128k. The data is averaged over prompt steps and normalized to a uniform scale. (a) The distribution of Command-R7B for InfiniteBench is relatively evenly spread across the context window. (b) The distribution of LLaMA-3.1 is mainly centered around the middle parts while slightly being shifted to early positions in the context window. LLaMA-3.1 observes no significant trend along increasing context window sizes. (c) Qwen2.5 displays the most skew to later positions. Its two additional curves display a further trend in this direction. (d) Much like LLaMA-3.1, the distribution of Phi-3 is centered around the middle; however, it slightly spreads out more to the beginning and ending parts of the context window.

We iterate through all these summaries, starting with the summary derived from the first 10 thousand tokens of the book and compare it with the subsequent summary derived from 20k tokens. With a step size of 10k tokens, we go through each pair of prompt steps until we reach the last context window of 128k tokens. For each pair of summaries, we classify the identified facts as *matching, new, and lost* facts compared to the following summary in line. We will refer to each step in this iteration as a prompt step. As in the previous metric, we split the prompt into a beginning, middle, and ending section to granularly distinguish the model performance depending on the position in the context window. For each fact we extract from the summary, we determine its informational origin inside the book from the summary sentence-to-prompt mapping established by the first metric. Given this positional information, we can evaluate whether the model can attend to the newly provided context window and whether it can retain early information and equal attention over the entire length of the context window. We base these performance indicators on the following criteria:

- If the model can attend to all parts of its context window, the matching facts are evenly distributed over the whole context window.
- The model can retain early information if we detect only a few lost facts at the beginning of the context window.
- The model can extract information from newly added context if we extract new facts from later positions inside the prompt.

The expected number of each fact type is proportional to the increase in prompt size at each iteration step of the evaluation. We can assume this proportionality in our datasets since they observe a high uniformity score, which signifies an even distribution of information over the entire length of the dataset entry. Hence, adding context to the prompt would yield new information proportionally to the relative extension amount.

Based on our assumption, the amount of additional information that is provided by extending the context window with each step by a constant number of additional tokens proportionally diminishes with larger context window sizes. In the first iteration step, from 10k to 20k tokens, we provide the model with 200% more tokens; this gradually decreases to only 6% additional information in the last step from 120k to 128k. As the number of extractable new information decreases with each step, the number of matching facts, that is, the facts that are retained from one step to the other, should ideally increase with the same amount. At the same time, the number of lost facts should decrease since the proportion of information the model has encountered before increases. However, due to the abundance of information contained in the book, the scale of the context window, and the limited length of the summary, we focus on the change of distribution of the different fact types over the starting, middle, and ending parts of the book.

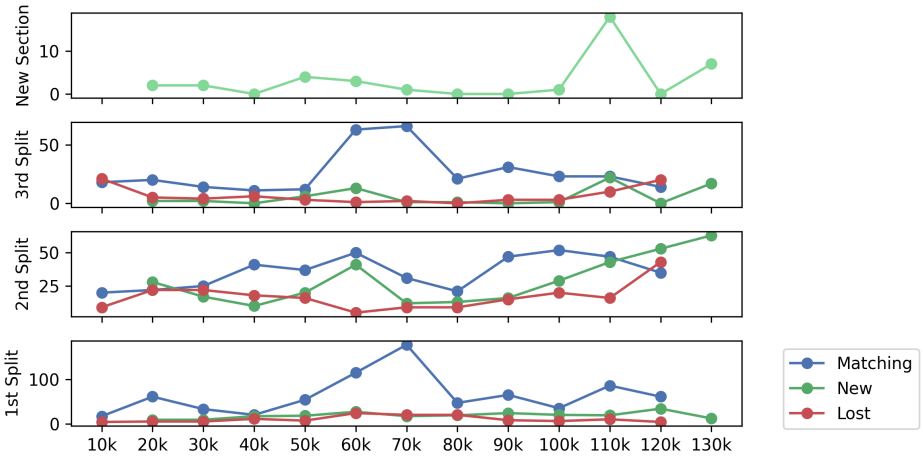


Figure 6.2: The figure shows the distribution of fact types across the starting, middle, ending, and new sections of the 10k tokens that are gradually added to the prompt in each iteration for Command-R7B. The results are from the InfiniteBench dataset. In the first section, we only have a few lost and new facts and a high and constant amount of matching facts. The second sector sees the most new facts. In the third sector, the model extracts very few new facts, which decreases even more in the added section, which provides the model with unseen information and should see a lot of new facts.

Caveats

As outlined in section 5.7.1, we evaluate each fact using two different scoring methods. The first method compares extracted facts to those from the reference summary. The second assesses accuracy by measuring the similarity between the candidate facts and those extracted from the corresponding paragraph in the book. However, we recognize that the second approach is relatively ineffective, as the maximum similarity scores remain low, with an average cosine similarity of just 0.45. Therefore, we focus solely on fact accuracies derived from the first method.

6.2.1 Command-R7B

The iteration steps up to 70,000 tokens from InfiniteBench, Command-R7B see a gradual increase in matching facts from 66.6% to 78.6%. This trend correlates with the ideal performance of gradually retaining more and more information as we increase the context window size. The distribution of matching facts over the three-thirds of the context window is relatively even. This trend coincides with a low fraction of lost facts, accounting for only 0.8% of all facts; this indicates that the model retains information relatively well, up to 70,000. However, after this threshold of 70k tokens, the percentage of matching facts decreases to an average of 49.8% of all facts, most significantly losing traction in the last third of the context window, where we observe 10% more lost facts than in the previous iteration steps. This gap is mainly

filled by new facts, which see an increase of 8 percentage points. This indicates that the model loses track of previously extracted information and substitutes it with new information it finds primarily extracted from the middle section of the context window. The fraction of new facts originating from the last section of the context window is only 11.05%, meaning that the model cannot extract much new information from the additional context window for each iteration step. The results of BookSum and XSum reveal similar trends.

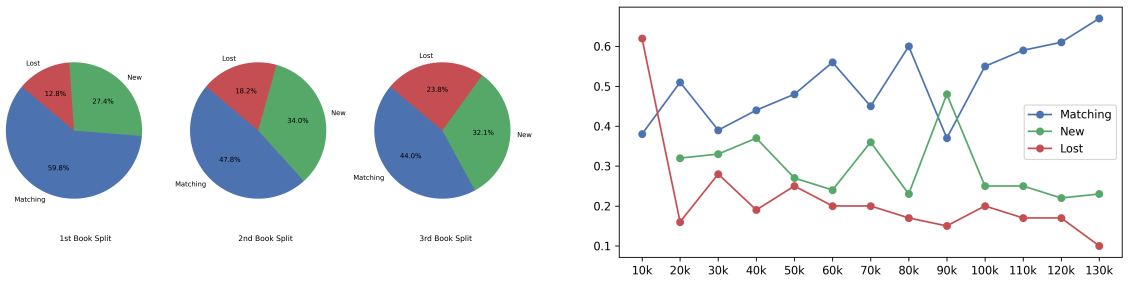
6.2.2 LLaMA 3.1

The performance of LLaMA 3.1 is more volatile than the performance of Command-R7B. The model retains facts from one iteration step to the other relatively badly, this is highlighted by the low overall percentage of matching facts of 45.0% compared to 58.9% of Command-R7B. Particularly in the second and third part of the context window LLaMA 3.1 has a small proportion of matching facts, coinciding with a high number of new facts. LLaMA 3.1 extracts new information comparatively well from later parts of the context window. The number of new facts steadily increases with the length of the context window. The model output is relatively in early parts, and keeps changing things in later parts.

The BookSum results exhibit similar performance trends to InfiniteBench regarding matching, new, and lost facts within the first two prompt sections. We observe a gradual increase in the matching fact ratio, rising from 51% at a context window of 10k tokens to 67% at 128k tokens, along with a decrease in lost facts by approximately 6 percentage points. However, performance in the section with the newly added 10k tokens at the end of the context window is stronger, as the model extracts a relatively high proportion of new facts from this segment. Fact accuracy, measured by similarity to the facts in the reference summary, is highest in the first part of the context window, with an average cosine similarity of 0.62. In contrast, the average fact accuracy in the remaining two-thirds of the context window declines slightly by 0.05 in cosine similarity. This suggests that the model retrieves early information with slightly higher factual accuracy than information from later positions in the context window.

6.2.3 Qwen2.5-7B

Qwen2.5 shows a low rate of fact retention, with only 30.3% of all extracted facts being encountered by previous prompt steps and classified as matching facts. The progression to larger context window sizes leads to a decreased proportion of matching facts and a increase in the number of new facts. Qwen2.5 predominantly extracts new facts from the latter sections of the context window, where they constitute 43.4% of all facts. Moreover, the proportion of new facts rises by 4.2 percentage points in the second sector and by 6.9 percentage points in the third sector over the progression from 10k to 128k tokens, indicating a positive performance with newly encountered context. The model exhibits similar performance in terms of fact distribution across BookSum as it does under InfiniteBench. The proportion of new and lost facts, primarily new facts, increases with their position within the context window



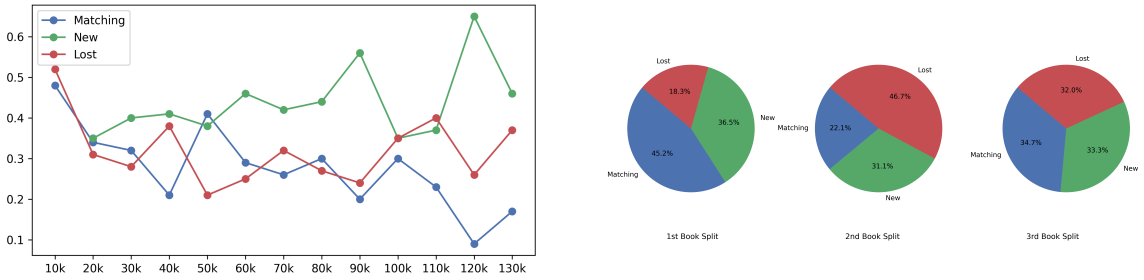
(a) InfiniteBench (b) BookSum

Figure 6.3: (a) shows the distribution of the facts types across the prompt's beginning, middle, and ending sections for LLaMA 3.1 and InfiniteBench. Compared to other models, LLaMA 3.1 retains relatively few matching facts and extracts more new facts, particularly from later parts of the prompt. (b) illustrates the trend in the proportions of matching, new, and lost facts as the prompt expands step by step. Matching facts account for approximately 50.9% of all facts and see a slight increase. The number of new facts remain consistently around 30%. Meanwhile, lost facts gradually decline as the context window increases.

and as the overall size of the prompt expands. From a context window size of 10k to 40k, the fraction of matching facts remains relatively high at 45.2%. However, this declines to approximately 28.4% for the remaining prompt steps leading up to 128k and is accompanied by an increase in lost and new facts to 39.4% and 32.2%, respectively. This trend indicates that the model's output becomes increasingly volatile, retaining significantly less information from previous prompt iterations and losing the majority of previously extracted facts.

6.2.4 Phi-3-small-8B

Phi-3-small has the lowest percentage of matching facts across all sections of its context window, with matching facts accounting for only 17.3% of all extracted facts across prompt steps. As the context window expands, the model gradually loses track of existing facts, particularly in the middle section, where lost facts make up 44% of all facts. At the same time, Phi-3 increasingly generates new facts from the first and second sections of the context window. However, it struggles to extract new information from later parts. From the 50k token mark onward, the model fails to extract new facts from the additional 10k tokens introduced at each step. As a result, Phi-3-small exhibits the most volatile performance, with a high proportion of both new and lost facts, meaning that the summaries differ significantly in each iteration step. Overall, the model prioritizes early positions, retrieves very few new facts from later sections of the prompt, and retains relatively little information over time.



(a) InfiniteBench (b) BookSum

Figure 6.4: (a) illustrates how Qwen2.5 processes facts from the InfiniteBench dataset as the context window expands, showing the changing proportions of matching and lost facts throughout this progression. The portion of matching facts decreases, while the portion of new and lost facts increases with a longer context window. (b) presents three pie charts illustrating the distribution of matching, new, and lost facts across different prompt steps: 10k–40k, 50k–80k, and 90k–128k. The charts reveal a significant decline in matching facts, while the proportions of lost and new facts steadily increase as the context window expands.

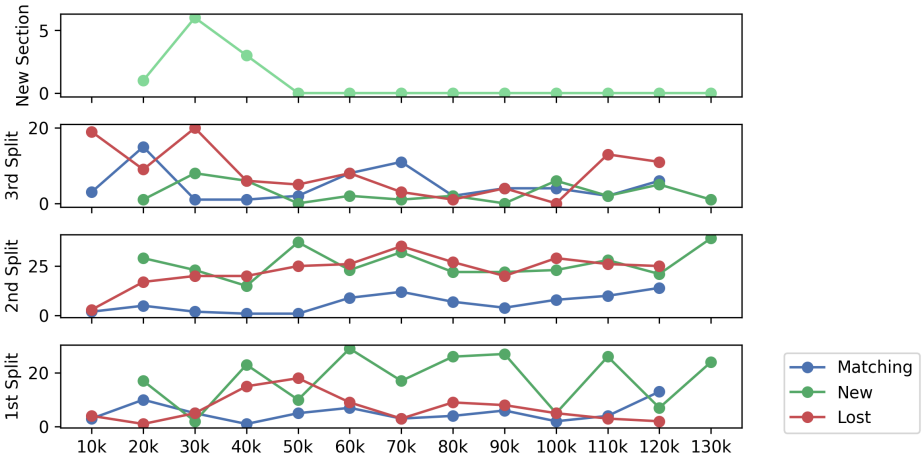


Figure 6.5: The figure illustrates the distribution of fact types across the starting, middle, ending, and new sections, where the new section represents the 10k tokens incrementally added to the prompt in each iteration for Phi-3-small on the dataset InfiniteBench. The ratio of matching facts remains relatively low across all three prompt sections, with most extracted facts being either new or lost. The first row indicates that, from 50k tokens onward, the model fails to extract new facts from the additional 10k tokens introduced at each step.

6.3 Question Answering

The third and final evaluation metric assesses whether the models can focus on a specific part of the context window to accurately answer questions based on information found in that location. We evaluate question-answering performance based on the location within the context window and track the model performance as the context window size gradually increases from 10k to 128k tokens. As described in Chapter 5.8, we divide the context window into three sections and generate questions for each section based on a specific paragraph within that section. The questions cover four distinct types. *abtractive*, *extractive*, *binary*, and *unanswerable* questions.

6.3.1 Command-R7B

The question-answering performance of Command-R7B for InfiniteBench is strongly influenced by the position within the context window and its overall length. Questions appearing in the first section of the context window achieve the highest average accuracy (0.51), followed by those in the second (0.42) and third sections (0.37), respectively. The bad performance on binary and extractive questions mainly dampens these low average accuracies. Binary questions perform the worst of all question types, with accuracies of 0.29, 0.17, and 0.19 across the three sections. Extractive questions show significantly higher performance in the first section, with scores of 0.58, 0.32, and 0.28, respectively. Abstractive and unanswerable questions perform significantly better, with average accuracies of 0.63 in the first section, 0.58 in the second, and 0.50 in the third. Notably, the accuracy of all question types is, on average, the highest in the first prompt section, gradually declining in accuracy for the subsequent part. This finding shows that Command-R7B can better comprehend information from earlier parts of the document than from later. A negative trend is also observed as the context window size increases, with binary questions in the first section experiencing the most notable decline in accuracy, averaging a -0.036 loss in accuracy. We find that the results from the other two datasets, BookSum and Xsum, strongly correlate with the findings of InfiniteBench.

6.3.2 LLaMA 3.1

LLaMA 3.1 answers questions from BookSum with relatively consistent accuracy across all three book sections, with only a slight advantage in the first prompt split (0.64) compared to the second (0.62) and third (0.61). Abstractive questions achieve notably high accuracy, scoring 0.73, 0.72, and 0.70 across the sections, with only a slight decline as the context window expands. The model struggles with extractive questions, answering them with lower accuracy rates of 0.54, 0.50, and 0.49. The most notable aspect of its performance is that answer accuracy does not degrade significantly over extending context windows. As with Command-R7B, we do not find significant performance differences between the other two datasets.

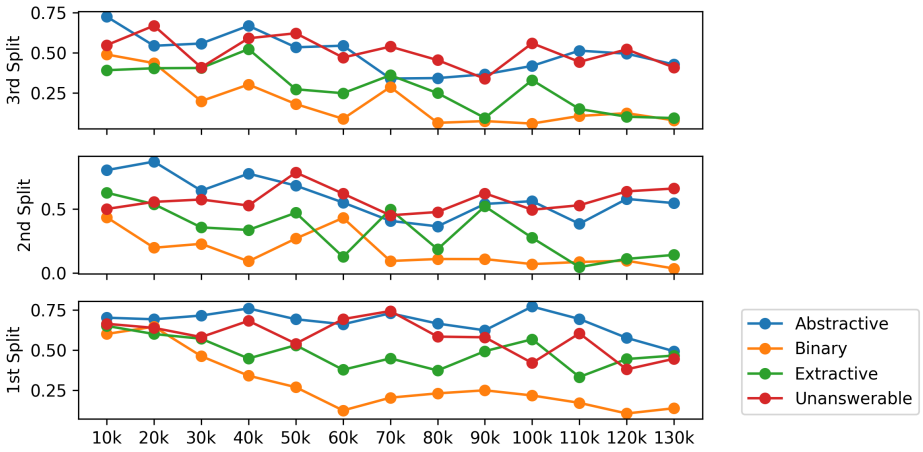


Figure 6.6: Graph of the question answering performance of Command-R7B for BookSum depending on context position and the length of the context window. A negative trend along the increase of the context window for all parts of the context window is observable.

6.3.3 Qwen2.5

The performance of Qwen2.5 for IninfiteBench, like LLaMA 3.1, remains stable even with longer context windows. The model performs strongly on abstractive, binary, and unanswerable questions, maintaining consistent accuracy rates around 0.72-0.73 across all prompt sections regardless of context length. However, it struggles with extractive questions, where accuracy drops to 0.56, 0.51, and 0.54 for each prompt part, respectively. Accuracy is particularly low for binary questions in the 10k to 30k token range. This is because the model provides only yes/no responses without explanations, while the ground truth answers require supporting reasoning.

6.3.4 Phi-3

Phi-3, like the other models, performs best on abstract questions, with accuracies of 0.7, 0.71, and 0.67. Unanswerable questions are the second best-performing questions, particularly on Inifnite, with an average accuracy of 0.75.

6.3.5 Overall Performance Trends

All tested models show comparable performance patterns across the datasets, with some notable distinctions. LLaMA 3.1 and Qwen2.5 demonstrate the best handling of extended context windows, showing minimal accuracy degradation as the context length increases. In contrast, Command-R7B's accuracy declines substantially across all question types as the context window expands. A typical pattern among all models is their stronger performance on questions related to content that appears early in the context window. When comparing

Model	Abstractive	Binary	Extractive	Unanswerable	
CommandR7B	0.67/0.59/0.49	0.29/0.17/0.19	0.48/0.32/0.28	0.58/0.57/0.51	(Inf)
	0.74/0.66/0.58	0.30/0.23/0.26	0.58/0.32/0.26	0.60/0.62/0.55	(Bk)
	0.68/0.56/0.61	0.4/0.27/0.29	0.56/0.41/0.43	0.52/0.4/0.41	(Xs)
LLaMA 3.1	0.72/0.71/0.69	0.56/0.54/0.54	0.49/0.46/0.47	0.65/0.66/0.65	(Inf)
	0.73/0.72/0.70	0.60/0.57/0.57	0.54/0.50/0.49	0.68/0.68/0.69	(Bk)
	0.73/0.70/0.69	0.55/0.51/0.53	0.60/0.61/0.62	0.58/0.59/0.63	(Xs)
Qwen2.5	0.7/0.71/0.73	0.72/0.71/0.71	0.49/0.45/0.51	0.63/0.64/0.61	(Inf)
	0.76/0.75/0.76	0.71/0.75/0.74	0.56/0.51/0.54	0.7/0.68/0.67	(Bk)
	0.72/0.72/0.73	0.73/0.75/0.77	0.56/0.58/0.6	0.62/0.65/0.63	(Xs)
Phi-3	0.70/0.71/0.67	0.56/0.45/0.55	0.5/0.44/0.45	0.70/0.70/0.71	(Inf)
	0.72/0.71/0.72	0.48/0.56/0.62	0.58/0.48/0.50	0.57/0.69/0.67	(Bk)
	0.76/0.73/0.76	0.66/0.66/0.65	0.69/0.66/0.65	0.65/0.60/0.63	(Xs)

Table 6.2: Lists the average cosine similarity of the candidate to reference the answer for each model, dataset, and question type. Higher scores indicate accurate answers. The entries in the last column refer to the datasets: Inf=InfinteBench, Bk=BookSum, Xs=Xsum

question types, *abstractive* questions consistently yield the highest accuracy scores, followed by *unanswerable* and *binary* questions. Command-R7B diverges from this pattern, showing particularly weak performance on binary questions compared to the other models. We explain the overall low accuracy of extractive answers with the nature of our datasets, which are highly abstractive, and feature little concrete, easily extractable information.
The overall most accurate model, with a score of 0.66 across all datasets and question types, is Qwen2.5, which is followed by Phi-3 (0.63) and Llama-3.1 (0.61). With a notable distance, Command-R7B performs the worst overall, with an average accuracy of 0.46.

Caveats

We acknowledge that binary questions pose challenges in evaluation. False negatives occur when the model provides a correct yes/no answer but omits the reasoning behind its decision, leading to a lower score despite accuracy. Conversely, the model may receive partial credit even when answering incorrectly since providing a reason, despite the answer being incorrect, sometimes scores higher than expected.

7 Discussion

This chapter reiterates the key findings of our thesis and point out its limitations.

7.1 Key Findings

Our first research question asked—***How can we adequately test the quality of text summarization and question answering produced by LLMs? Does the quality of the generated summary or answer improve when more content from the article is provided to the model?***—To investigate this, we successfully implemented a benchmarking pipeline comprising three metrics to assess LLMs' text summarization and question-answering capabilities in a setting of extremely long context window sizes. For each of the metrics, the evaluation framework tests how two key factors affect model performance across all metrics: the position of salient information within the context window and the overall context window size. The first metric introduced a methodology to determine the information distribution of candidate summaries. The positional examination provided a comprehensive understanding of how the models use information differently within the context window. The second metric comprehensively examines how effectively models extract information across different context window sizes. It tests how models perform in three key areas: their ability to capture new information from expanded context windows, any potential loss of information from previously covered sections, and their capacity to retain critical information from one prompt step to another. This approach allows us to assess the gains and potential trade-offs extending the context window of LLMs. The third evaluation metric assesses models' ability to synthesize information from source documents. We tested this through four distinct question types: abstractive questions to evaluate information synthesis and inference capabilities, extractive questions to measure direct information retrieval, binary questions to assess decision-making based on complex dependencies, and unanswerable questions to evaluate the models' ability to recognize when they cannot provide valid answers, thereby helping identify potential hallucinations. Through these metrics, we contribute a comprehensive study of the long context window understanding of LLMs.

To address the second research question:***What are the most effective techniques for extending the context window of LLMs?***, we first reviewed various existing techniques designed to extend the context window of LLMs. These techniques included efficient transformer modifications at the memory and computational level, adapted attention mechanisms, and positional encoding modifications. We concluded this review by selecting three modern LLMs, each adopting a different context window extension technique—RoPE, YaRN, and

LongRoPE. We tested these models with our evaluation pipeline to determine if there were performance variations based on the different methods. We arrived at the conclusion that all tested models, and by extension, all techniques, observe some challenges and only marginal performance differences between each other in our setting of extremely long context window sizes. Regarding the context window extension techniques, LLaMA 3.1 can be viewed as the baseline model for Qwen2.5 and Phi-3 as it uses RoPE, the foundational technique for the subsequent iterations, YaRN, and LongRoPE, which the other two models use. As such, LLaMA 3.1 is an interesting indicator of the success of YaRN and LongRoPE. However, our evaluation results suggest that the models perform relatively equally across all extension techniques, with no clear standout. Interestingly, the only closed-source model exhibited the weakest overall performance.

Finally, we explored the third research question— ***How do LLMs utilize the information within their context window during text summarization and question-answering tasks? Can LLMs distribute their attention uniformly across the entire document, or is their attention clustered towards specific sections?***—The results of our evaluation pipeline lead us to the following conclusions. Using attention visualization techniques and performance analysis, we observed that all tested models do not distribute their attention uniformly across an entire document. Instead, their attention tends to concentrate on specific sections. For instance, Command-R7B primarily focused on the introduction, LLaMA-3.1 and Phi-3 showed the strongest attention to the middle section, and Qwen2.5 directed the most focus toward the ending sections of the prompt. Secondly, in our methodology for atomic facts extraction and comparison, we observed that as the context window size gradually extends, the models struggle to retrieve new information from certain sections, lose track of details from others, and fail to retain the expected amount of information, particularly at the beginning or end of a document. The question-answering metric could determine the accuracy of the questions significantly depending on the position of the question and the context window size.

7.2 Limitations

For our first research question, we examined four models using different approaches to extend their context windows to determine the most potent performing technique. However, this comparison is inherently flawed since the models differ in many more aspects of their architecture and training than merely in their context window extension technique. This makes it difficult to isolate the specific impact of the context extension techniques on the model's performance. The only accurate baseline for comparison is the standard transformer architecture. Future iterations of our experiments should employ different versions of a single LLM, that are trained or fine-tuned on different context window extension techniques, in order to facilitate a direct comparison.

8 Conclusion and Outlook

This thesis encompassed an in-depth exploration of how contemporary LLMs adapt to extended context window sizes and a comprehensive evaluation pipeline, testing LLMs on test summarization and question-answering tasks, specifically emphasizing long-context capabilities. We drew the relevancy of our approach from the increasing importance of long-document inference in the digital age, where the rapid growth of textual data challenges LLMs to process more and more information, and emphasized the necessity of evaluating the performance of LLMs in such settings. After reviewing a suite of different approaches for benchmarking LLMs on text summarization and question-answering tasks, we concluded that most of the benchmarks only tested for a limited context length and contributing a custom evaluation tool would provide tremendous value to more adequately test for context window sizes of up to 128,000 tokens. Our evaluation resulted in a comprehensive understanding of how different LLMs use their context window differently, depending on the position of salient information within it, and the length of it. In order to make these evaluation data more accessible, we then expanded our pipeline for graph visualization and better user interaction through our demo app.

Going forward with our findings, we identify potential directions for future research. Firstly, the evaluation can be expanded to include more models to give a more holistic performance analysis of the landscape of countless LLMs. This expansion would allow for the identification of differences between open -and closed-source models, for example. Additionally, as discussed in the *Limitations*, our evaluation pipeline could be used to examine the performance differences of altercations of one model that only vary in one aspect of the model architecture, like their context window extension technique, to identify the impact of that particular modification the overall performance.

List of Figures

List of Tables

Bibliography

[1] M. Fire and C. Guestrin. "Over-optimization of academic publishing metrics: observing Goodhart's Law in action". In: *GigaScience* 8.6 (2019), giz053.

[2] B. Schneier. *Data and Goliath: The hidden battles to collect your data and control your world*. WW Norton & Company, 2015.

[3] Y. Li, S. Miao, H. Huang, and Y. Gao. "Word Matters: What Influences Domain Adaptation in Summarization?" In: *arXiv preprint arXiv:2406.14828* (2024).

[4] L. Basyal and M. Sanghvi. "Text summarization using large language models: a comparative study of MPT-7B-instruct, Falcon-7b-instruct, and OpenAI Chat-GPT models". In: *arXiv preprint arXiv:2310.10449* (2023).

[5] S. Robertson. "Understanding inverse document frequency: on theoretical arguments for IDF". In: *Journal of documentation* 60.5 (2004), pp. 503–520.

[6] R. Mihalcea and P. Tarau. "Textrank: Bringing order into text". In: *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004, pp. 404–411.

[7] H. Shakil, A. Farooq, and J. Kalita. "Abstractive text summarization: State of the art, challenges, and improvements". In: *Neurocomputing* (2024), p. 128255.

[8] K. Ishwari, A. Aneeze, S. Sudheesan, H. Karunaratne, A. Nugaliyadde, and Y. Mallawarrachchi. "Advances in natural language question answering: A review". In: *arXiv preprint arXiv:1904.05276* (2019).

[9] J. Saad-Falcon, J. Barrow, A. Siu, A. Nenkova, D. S. Yoon, R. A. Rossi, and F. Dernoncourt. "Pdftriage: Question answering over long, structured documents". In: *arXiv preprint arXiv:2309.08872* (2023).

[10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[11] M. Raeini. "The Evolution of Language Models: From N-Grams to LLMs, and Beyond". In: *SSRN: https://ssrn.com/abstract=4625356* (2023).

[12] S. Jana, K. Pal, S. Biswas, and K. Roy. "The Evolution and Impact of Large Language Model Systems: A Comprehensive Analysis". In: *SSRN* (Mar. 2024).

[13] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur. "Recurrent neural network based language model". In: *Interspeech* (2010), pp. 1045–1048. ISSN: 2958-1796. DOI: 10.21437/Interspeech.2010-343.

[14] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[15] A. Vaswani. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017).

[16] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun. "Pre-Trained Language Models and Their Applications". In: *Engineering* 25 (2023), pp. 51–65. ISSN: 2095-8099. DOI: 10.1016/j.eng.2022.04.024. URL: https://www.sciencedirect.com/science/article/pii/S2095809922006324.

[17] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, et al. "Instruction tuning for large language models: A survey". In: *arXiv preprint arXiv:2308.10792* (2023).

[18] F. D. Keles, P. M. Wijewardena, and C. Hegde. "On the computational complexity of self-attention". In: *International Conference on Algorithmic Learning Theory*. PMLR. 2023, pp. 597–619.

[19] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. "Efficient Transformers: A Survey". In: *ACM Computing Surveys* 55 (2020), pp. 1–28. URL: https://api.semanticscholar.org/CorpusID:221702858.

[20] I. Beltagy, M. E. Peters, and A. Cohan. "Longformer: The long-document transformer". In: *arXiv preprint arXiv:2004.05150* (2020).

[21] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. "Linformer: Self-attention with linear complexity". In: *arXiv preprint arXiv:2006.04768* (2020).

[22] X. Wang, M. Salmani, P. Omidi, X. Ren, M. Rezagholizadeh, and A. Eshaghi. "Beyond the limits: A survey of techniques to extend the context length in large language models". In: *arXiv preprint arXiv:2402.02244* (2024).

[23] N. Kitaev, Ł. Kaiser, and A. Levskaya. "Reformer: The efficient transformer". In: *arXiv preprint arXiv:2001.04451* (2020).

[24] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Q. V. Le, J. Laudon, et al. "Mixture-of-experts with expert choice routing". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 7103–7114.

[25] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* (2023).

[26] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. "Gqa: Training generalized multi-query transformer models from multi-head checkpoints". In: *arXiv preprint arXiv:2305.13245* (2023).

[27] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. "The llama 3 herd of models". In: *arXiv preprint arXiv:2407.21783* (2024).

[28] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, et al. "Qwen2. 5 technical report". In: *arXiv preprint arXiv:2412.15115* (2024).

[29] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. "Flashattention: Fast and memory-efficient exact attention with io-awareness". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 16344–16359.

[30] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. "AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration". In: *Proceedings of Machine Learning and Systems* 6 (2024), pp. 87–100.

[31] H. Y. Koh, J. Ju, M. Liu, and S. Pan. "An empirical survey on long document summarization: Datasets, models, and metrics". In: *ACM computing surveys* 55.8 (2022), pp. 1–35.

[32] C. Yun, S. Bhojanapalli, A. S. Rawat, S. J. Reddi, and S. Kumar. "Are transformers universal approximators of sequence-to-sequence functions?" In: *arXiv preprint arXiv:1912.10077* (2019).

[33] S. Pawar, S. Tonmoy, S. Zaman, V. Jain, A. Chadha, and A. Das. "The What, Why, and How of Context Length Extension Techniques in Large Language Models–A Detailed Survey". In: *arXiv preprint arXiv:2401.07872* (2024).

[34] S. Chen, S. Wong, L. Chen, and Y. Tian. "Extending Context Window of Large Language Models via Positional Interpolation, 2023". In: *URL https://arxiv. org/abs/2306.15595* ().

[35] O. Press, N. A. Smith, and M. Lewis. "Train short, test long: Attention with linear biases enables input length extrapolation". In: *arXiv preprint arXiv:2108.12409* (2021).

[36] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. "Roformer: Enhanced transformer with rotary position embedding". In: *Neurocomputing* 568 (2024), p. 127063.

[37] S. Chen, S. Wong, L. Chen, and Y. Tian. "Extending context window of large language models via positional interpolation". In: *arXiv preprint arXiv:2306.15595* (2023).

[38] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. "Mistral 7B". In: *arXiv preprint arXiv:2310.06825* (2023).

[39] Y. Sun, L. Dong, B. Patra, S. Ma, S. Huang, A. Benhaim, V. Chaudhary, X. Song, and F. Wei. "A length-extrapolatable transformer". In: *arXiv preprint arXiv:2212.10554* (2022).

[40] B. Peng, J. Quesnelle, H. Fan, and E. Shippole. "Yarn: Efficient context window extension of large language models". In: *arXiv preprint arXiv:2309.00071* (2023).

[41] Y. Ding, L. L. Zhang, C. Zhang, Y. Xu, N. Shang, J. Xu, F. Yang, and M. Yang. "Longrope: Extending llm context window beyond 2 million tokens". In: *arXiv preprint arXiv:2402.13753* (2024).

[42] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker. "Perplexity—a measure of the difficulty of speech recognition tasks". In: *The Journal of the Acoustical Society of America* 62.S1 (1977), S63–S63.

[43] F. L. Bronnec, A. Verine, B. Negrevergne, Y. Chevaleyre, and A. Allauzen. "Exploring precision and recall to assess the quality and diversity of LLMs". In: *arXiv preprint arXiv:2402.10693* (2024).

[44] C.-Y. Lin. "Rouge: A package for automatic evaluation of summaries". In: *Text summarization branches out*. 2004, pp. 74–81.

[45] J. P. Jiawei Han Micheline Kamber. "2 - Getting to Know Your Data". In: *he Morgan Kaufmann Series in Data Management Systems, Data Mining (Third Edition)* (2012), pp. 39–82. URL: https://doi.org/10.1016/B978-0-12-381479-1.00002-2.

[46] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. "Bertscore: Evaluating text generation with bert". In: *arXiv preprint arXiv:1904.09675* (2019).

[47] J. Devlin. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[48] N. Reimers. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[49] H. Face and S.-T. Team. *all-MiniLM-L6-v2*. Accessed: 2025-02-13. 2020. URL: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

[50] B. Chen, Z. Zhang, N. Langrené, and S. Zhu. "Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review". In: *arXiv preprint arXiv:2310.14735* (2023).

[51] X. Liu, J. Wang, J. Sun, X. Yuan, G. Dong, P. Di, W. Wang, and D. Wang. "Prompting frameworks for large language models: A survey". In: *arXiv preprint arXiv:2311.12785* (2023).

[52] U. Shaham, E. Segal, M. Ivgi, A. Efrat, O. Yoran, A. Haviv, A. Gupta, W. Xiong, M. Geva, J. Berant, et al. "Scrolls: Standardized comparison over long language sequences". In: *arXiv preprint arXiv:2201.03533* (2022).

[53] Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou, et al. "Longbench: A bilingual, multitask benchmark for long context understanding". In: *arXiv preprint arXiv:2308.14508* (2023).

[54] C. An, S. Gong, M. Zhong, X. Zhao, M. Li, J. Zhang, L. Kong, and X. Qiu. "L-eval: Instituting standardized evaluation for long context language models". In: *arXiv preprint arXiv:2307.11088* (2023).

[55] X. Zhang, Y. Chen, S. Hu, Z. Xu, J. Chen, M. K. Hao, X. Han, Z. L. Thai, S. Wang, Z. Liu, et al. "InfinityBench: Extending Long Context Evaluation Beyond 100K Tokens". In: *arXiv preprint arXiv:2402.13718* (2024).

[56] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. "Lost in the middle: How language models use long contexts". In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 157–173.

[57] H. Lee, S. Yoon, F. Dernoncourt, D. S. Kim, T. Bui, J. Shin, and K. Jung. "KPQA: A Metric for Generative Question Answering Using Keyphrase Weights". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou. Online: Association for Computational Linguistics, June 2021, pp. 2105–2115. DOI: 10.18653/v1/2021.naacl-main.170. URL: https://aclanthology.org/2021.naacl-main.170/.

[58] W. Kryściński, N. Rajani, D. Agarwal, C. Xiong, and D. Radev. "Booksum: A collection of datasets for long-form narrative summarization". In: *arXiv preprint arXiv:2105.08209* (2021).

[59] H. Face. *xinrongzhang2022/InfiniteBench*. Accessed: 2024-12-06. 2024. URL: https://huggingface.co/datasets/xinrongzhang2022/InfiniteBench/viewer/default/passkey.

[60] S. Narayan, S. B. Cohen, and M. Lapata. "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization". In: *arXiv preprint arXiv:1808.08745* (2018).

[61] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. W. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi. "Factscore: Fine-grained atomic evaluation of factual precision in long form text generation". In: *arXiv preprint arXiv:2305.14251* (2023).

[62] L. Hilgert, D. Liu, and J. Niehues. "Evaluating and Training Long-Context Large Language Models for Question Answering on Scientific Papers". In: *Proceedings of the 1st Workshop on Customizable NLP: Progress and Challenges in Customizing NLP for a Domain, Application, Group, or Individual (CustomNLP4U)*. 2024, pp. 220–236.

[63] A. Gomez. *Introducing Command R7B: Fast and efficient generative AI*. Accessed: 2025-02-05. 2024. URL: https://cohere.com/blog/command-r7b.

[64] M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl, et al. "Phi-3 technical report: A highly capable language model locally on your phone, 2024". In: *URL https://arxiv. org/abs/2404.14219* (2024).