Reproducing and Analyzing the "Lost in the Middle" and "The Power of Noise" Phenomenon in Retrieval-Augmented Generation

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Kousik Samanta
Roll No: CS2417

under the guidance of
Prof. Mandar Mitra
Professor
Computer Vision and Pattern Recognition

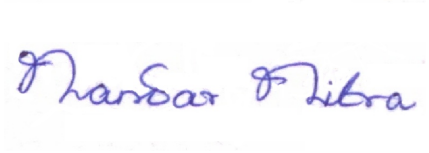
Indian Statistical Institute
Kolkata-700108, India

May 2026

CERTIFICATE

This is to certify that the dissertation entitled "Reproducing and Analyzing the 'Lost in the Middle' and 'Power of Noise' Phenomena in Retrieval-Augmented Generation" submitted by Kousik Samanta to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of Master of Technology in Computer Science, is a bonafide record of work carried out by him under my supervision and guidance.

The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.



Mandar Mitra
Professor
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to express my heartfelt gratitude to my supervisor, Prof. Mandar Mitra, for his constant guidance, encouragement, and insightful feedback throughout the course of this dissertation. His expertise and support have been invaluable. Sir, suggested the idea of isolating the experiment in Document-Grounded and Answer-Grounded setups. I am deeply thankful to Sourav Saha, Senior Research Fellow at ISI Kolkata, for his invaluable guidance and mentorship throughout this dissertation. His assistance in resolving technical challenges, sourcing datasets, and structuring the experimental design was truly instrumental to this work.

I also thank the faculty and my peers at the Indian Statistical Institute for providing an intellectually stimulating environment throughout my time here.

Kousik Samanta
Indian Statistical Institute,
Kolkata-700108, INDIA.

Abstract

Retrieval-Augmented Generation has become the way to improve Large Language Models. They help with problems like knowledge and hallucinations. Recent studies show that these models still have limitations.

One big problem is the "Lost in the Middle" phenomenon. Models can't access information in the middle of contexts properly. Another counterintuitive observation is the "Power of Noise" paradigm, which suggests adding unrelated documents can actually make the generation better.

We know these happen in extractive QA tasks, but we don't know if they happen in tasks that need complex reasoning. This dissertation looks into how position and noise affect Long-Form Question Answering. We use the ELI5 dataset and test three models. We give them varying amounts of context and see how they do. We also change the location of the correct information and add distracting or random information to observe the effects of these perturbations.

Traditional metrics for evaluating model-generated answers aren't very effective for long-form responses. We introduce two new metrics of evaluation, PropScore and SentenceScore.

Our experiments give us three findings. First, the "Lost in the Middle" issue still happen to a certain degree in Long-Form QA. Second, we confirm that noise can actually improve generation. Third, we hypothesize the reasons of persistence of the "Lost in the Middle" phenomenon and the "power of noise" paradigm in Long-Form QA.

Contents

Chapter 1

Introduction

Following the seminal works of [3] in 2020, Retrieval-Augmented Generation (RAG) became one of the most widely studied and deployed techniques across industry and academia. Its popularity rises due to its ability to compensate for some key limitations of pre-trained Large Language Models (LLMs).
In the works of [1] in 2023 and [2] in 2024, we observed that the position of the relevant information in the prompt heavily influences the output quality of LLMs in an RAG pipeline. [2] also showed that including noise along with relevant information in the prompt improves the quality of the output. These works mostly focused on extractive QA task. In this paper, we examine whether similar phenomena persist in the case of Long Form QA tasks.

1.1 Retrieval-Augmented Generation

RAG is a modern architecture built to optimize the output quality of LLMs. It consists of two main components, **retrieval component** and **generation component**.
The retrieval component retrieves documents (or passages) from a database (mainly vector databases) using the query, and then it augments the prompt with the retrieved documents, along with the query and system instructions to provide additional or necessary knowledge to the LLM.

Prompt 1

SYSTEM-INSTRUCTION
Context:
DOCUMENT-1
DOCUMENT-2
.....
DOCUMENT-N
Question: *QUESTION*
Answer:

The generation component is fed these retrieved-document-augmented prompts. These documents provide extra knowledge to the LLM that it might not have, the underlying assumption being that the better the retrieval, the better the output quality.

1.2 Why is RAG useful?

LLMs hold a knowledge base via encoding knowledge in their parameters during training. Consequently, LLMs heavily rely on the data it was trained on, resulting in some severe limitations:

- LLM's knowledge can become outdated.
- Its output might be wrong if the query is related to a domain for which it didn't had data during training.
- Due to the sheer size of its knowledge base, LLM might struggle to fetch a proper response, and may hallucinate, generating irrelevant information in output.

To resolve the first two limitations, LLMs need to be fine-tuned with new data each time, which is expensive and time-consuming. RAG addresses all three issues to some degree without needing the LLMs to be fine-tuned:

- updated data can be added in the database, retrieved during a query, and augmented in a prompt, addressing the first issue.
- the external domain knowledge can be included in the prompt, providing context for the out-of-domain queries.

- providing context in the prompt along with query can help the LLM to put attention to correct information, reducing chances of hallucination.

So, RAG provides a cost-effective solution for the limitations of LLMs, makes the usage of pre-trained LLM across domains more reliable, and improves the generation quality of LLMs.

1.3 Lost in the Middle

The paper [1] analyzed how the performance of an LLM varies across multi-document QA and key-value retrieval tasks, based on the context length of a prompt(i.e., information volume) and the position of relevant information within the context of a prompt.
The authors of the paper found that an LLM performs best when the relevant information appears either at the very beginning (primacy bias) or at the end of its input context (recency bias). On the other hand, the performance of an LLM drops significantly when the relevant information appears in the middle of the input context. It demonstrates that LLMs fail to retrieve information properly from the middle of a long context, and a lot of information from the middle of a context is lost. The authors refer to this as the **"Lost in the Middle"** phenomenon in LLMs.
The authors of [2] also verifies this phenomenon using different datasets and different models.

1.4 Power of Noise

The paper [2] (i) verified the **"Lost in the Middle"** phenomenon with different datasets and different models; (ii) demonstrated that introducing controlled noise in the prompt context can actually improve the generation quality of LLMs.
The authors showed that augmenting the query with only retrieved documents, along with one or more relevant documents, can actually be counterproductive, as cheaper retrieval methods rely mostly on Lexical or semantic similarity. As a result, documents that contain misinformation or claims directly opposing the claims of relevant documents (gold documents) can be scored high, due to high lexical or semantic similarity. For example, suppose our query is "When was Bob born?". The relevant (gold) document has this sentence: "Bob's date of birth is 16th June, 2026". But another document has the sentence: "Bob wasn't born in 1980". The latter document can be ranked high if retrieval relies only on lexical matching. Contextual

similarity has similar pitfalls. The authors showed that controlled injection of noise (random documents) along with high-ranking retrieved documents and relevant documents in the prompt can improve the generation quality of an LLM. The noise helps LLM recognize the right patterns in the context. So, as controlled inclusion of noise can increase the model's generative performance, the authors termed this phenomenon as **"Power of Noise"**.

1.5 Problem Statement

While RAG effectively mitigates the frozen knowledge and hallucination in LLM, recent works demonstrate that these models suffer from positional biases. Specifically, the "Lost in the Middle" phenomenon suggests these models struggle to utilize relevant information from the middle of long-context prompts. Additionally, the "Power of Noise" paradigm argues that controlled injection of noise (random documents) can counterintuitively improve generation quality.
However, existing research has verified these phenomena in Extractive QA tasks, leaving the question of whether these structural vulnerabilities and noise-induced performance gain persist in Long Form QA tasks, where models need to synthesize output and perform complex reasoning.
So, our primary objective for this dissertation is to reproduce the two mentioned phenomena in the scope of Long-Form QA tasks using the ELI5 dataset. We ask the following questions in this work:

- **RQ1:** To what extent does the "Lost in the Middle" phenomenon manifest in the scope of Long-Form QA task?
- **RQ2:** How does the injection of random noise influence the generation and reasoning capabilities of LLMs in Long-Form answer tasks?
- **RQ3:** What causes these kinds of phenomena to persist in Long-Form QA tasks, if they does persist?

Chapter 2

Background

This chapter provides the theoretical and empirical foundation for the experiments described in this dissertation. We begin with a deeper treatment of Retrieval-Augmented Generation (RAG), covering its architecture, retrieval strategies, and known failure modes. We discuss the "lost in the middle" phenomenon and the "power of noise" paradigm, followed by a brief review of related literature on long-context language models, positional biases, and evaluation frameworks for generative QA.

2.1 More on Retrieval-Augmented Generation

2.1.1 Architecture and Components

Retrieval-Augmented Generation (RAG), introduced by [3], frames knowledge-intensive NLP tasks as a two-stage pipeline: a *retriever* and a *generator*. Given a query q, the retriever selects a set of k documents $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$ from a large external corpus, and the generator — a pre-trained language model — produces an answer conditioned on both q and $\mathcal{D}$.

Formally, the generation process computes:

$$P(a \mid q) = \sum_{d \in \mathcal{D}} P(a \mid q, d) \cdot P(d \mid q)$$

where $P(d \mid q)$ is the retriever's relevance score for document d given query q, and $P(a \mid q, d)$ is the generator's likelihood of producing answer a given the query and document. In practice, the summation is approximated by concatenating the top-k retrieved documents into a single prompt context, as illustrated in Prompt 1 in Chapter 1.

2.1.2 Retrieval Strategies

The retriever is the first critical component of any RAG pipeline. Retrieval methods broadly fall into three categories:

Sparse Retrieval (Lexical): Methods such as BM25 [46] represent queries and documents as sparse term-frequency vectors and score matches via term overlap. These are computationally efficient and highly interpretable, but fail to capture synonymy and paraphrasing.

Dense Retrieval (Semantic): Methods such as DPR [40] and Contriever [41] encode queries and documents into dense vector embeddings using bi-encoder neural networks. Retrieval is then performed via maximum inner-product search (MIPS) over an index (e.g., FAISS). Dense retrievers capture semantic similarity more flexibly, but the very flexibility that benefits them can also surface misleading documents: a passage that discusses the same topic but directly contradicts the gold answer may receive a high similarity score.

Hybrid Retrieval: Systems combining sparse and dense signals, such as SPLADE [22], aim to capture the complementary strengths of both approaches.

A key limitation relevant to this work is that both sparse and dense retrieval are *similarity-based*: they rank documents by their surface-level or semantic proximity to the query, not by the *factual correctness* of the information they contain. This means that a document making a claim directly opposed to the correct answer (e.g., negating a fact with high lexical overlap to the gold document) can still receive a high retrieval rank. As demonstrated by [2], this property makes the composition of the retrieved context — how many relevant documents, irrelevant documents, and noise documents are included — a critical design decision.

2.1.3 The Augmented Prompt and Context Length

The augmented prompt fed to the generator typically follows a template such as:

[System Instruction]
Document 1 ... Document k
Question: Q
Answer:

As k increases, the generator must reason over progressively longer contexts. Modern instruction-tuned models support large context windows (e.g., 128K tokens

for Qwen2.5 and Phi-3-mini), but a large context window does not guarantee that the model effectively *attends* to all parts of that context with equal fidelity. The central empirical observation this dissertation investigates — that the position of the relevant document within this prompt structure systematically affects output quality — is precisely the "Lost in the Middle" phenomenon.

2.2 More on the "Lost in the Middle" Phenomenon

2.2.1 The Core Finding

The "Lost in the Middle" phenomenon was formally characterized by [1]. Through controlled experiments on multi-document question answering (using NaturalQuestions-Open) and a synthetic key-value retrieval task, the authors demonstrated that language model performance follows a characteristic **U-shaped curve** as a function of the position of the relevant (gold) document within the input context. Performance peaks when the gold document appears at the *beginning* (primacy bias) or at the *end* (recency bias) of the context, and degrades substantially when the gold document is placed in the *middle*, even for models with explicitly extended context windows.

The magnitude of the effect is notable. For GPT-3.5-Turbo on the multi-document QA task with 20 or 30 total documents, worst-case performance (gold in the middle) falls *below* closed-book performance — i.e., the model answers more accurately when given *no documents at all* than when given 20 documents with the gold one buried in the middle [1]. This suggests that the presence of distracting context actively harms the retrieval of relevant information from the middle of the prompt.

2.2.2 Extended-Context Models Are Not Immune

A particularly important finding from [1] is that extended-context models (e.g., GPT-3.5-Turbo 16K vs. GPT-3.5-Turbo 4K; Claude-1.3 100K vs. Claude-1.3 8K) show nearly identical performance curves when evaluated on prompts that fit within both models' context windows. This indicates that the U-shaped degradation is not simply a consequence of a model operating beyond its context limit — it is a structural property of how these models attend to long input sequences. Having a *larger* context window does not translate to *better* utilization of that window.

2.2.3 Architectural and Training Factors

[1] performed a preliminary investigation into the mechanistic causes of this phenomenon, focusing on three factors:

Model Architecture (Decoder-only vs. Encoder-Decoder). Decoder-only models (such as GPT-3.5-Turbo and Llama) process the input context causally, attending to each token only from previous tokens. As a result, when the query appears at the end of the prompt, the model cannot attend to the query tokens while contextualizing the documents that precede it. In contrast, encoder-decoder models (such as Flan-T5-XXL and Flan-UL2) use a bidirectional encoder, enabling every document position to be contextualized with awareness of the full input, including the query. [1] found that encoder-decoder models are relatively robust to changes in the position of relevant information — but only when evaluated on sequences within their training-time context length. When sequences exceed the training-time length, these models, too, begin to show a U-shaped degradation curve.

Query-Aware Contextualization. A practical mitigation proposed and evaluated by [1] is to place the query *both before and after* the documents in the prompt, enabling decoder-only models to partially simulate the bidirectional attention of encoder-decoder models. This technique yields near-perfect performance on the synthetic key-value retrieval task, indicating that basic token-level retrieval benefits greatly from query-awareness. However, it provides only minimal improvement on the multi-document QA task, suggesting that more complex multi-hop reasoning over long contexts requires deeper architectural solutions.

Instruction Fine-Tuning. Comparing MPT-30B (base model) and MPT-30B-Instruct, [1] found that both models exhibit the U-shaped performance curve, indicating that the phenomenon is not a byproduct of the instruction fine-tuning procedure. Instruction fine-tuning slightly reduces the worst-case performance gap (from $\sim$10% to $\sim$4% for MPT-30B), but does not eliminate the positional bias. Crucially, even base models without instruction fine-tuning exhibit primacy and recency biases when prompted with task-formatted data, suggesting these biases arise during pre-training from the statistical structure of web text (e.g., patterns in Stack Overflow question-answer pairs, where context-setting often appears at document boundaries).

Model Scale. Experiments with Llama-2 at 7B, 13B, and 70B parameter scales revealed a noteworthy scaling effect: the 7B model shows only *recency* bias (a purely monotonically increasing performance curve as the gold document moves toward the end), while the 13B and 70B models exhibit the full U-shaped curve with both primacy and recency bias. This suggests that primacy bias emerges as a property of larger models, and that the U-shaped curve is not universal across model scales — smaller models may simply be recency-biased without meaningful primacy bias [1].

2.3 The Power of Noise

Building on the "Lost in the Middle" findings, [2] extended the investigation in two important directions. First, they verified the U-shaped positional degradation phenomenon using different datasets and different model architectures, corroborating the finding's generality. Second, and more counterintuitively, they demonstrated that injecting *random, unrelated documents* (noise) into the prompt context alongside retrieved documents can *improve* generation quality under certain conditions.

The rationale for the latter finding is as follows. Pure retrieval-based context construction — selecting the top-k documents by relevance score — tends to fill the context with documents that are highly topically related to the query. While this includes the gold document, it also includes many *distractor* documents: passages that are lexically or semantically similar to the query but either fail to answer it or, more dangerously, contain claims that directly oppose the correct answer. Because retrieval scoring does not distinguish between supporting and contradicting evidence, high-scoring distractors can actively mislead the generator.

Introducing random noise documents disrupts this cluster of topically similar but potentially contradictory contexts. The noise documents have no systematic relationship to the query, so they do not reinforce incorrect claims. The result is that the LLM can more easily identify the signal (the gold document) against a background of diverse, uncorrelated noise than against a background of coherent but partially adversarial distractors. [2] termed this the **"Power of Noise"** paradigm: controlled injection of noise acts as a form of implicit regularization for the retrieval context, improving the generator's ability to extract and synthesize relevant information.

2.4 Literature Review

2.4.1 Long-Context Language Models

The ability to process long input sequences is a fundamental capability for RAG systems. Early Transformer models [25] were trained with relatively small context windows (512–2048 tokens) due to the quadratic memory and compute requirements of full self-attention. A substantial body of work has sought to extend or approximate this capability.

Sparse attention mechanisms, such as Longformer [26] and BigBird [27], reduce the quadratic complexity by attending to local windows and global tokens, enabling processing of sequences up to 4K–8K tokens. *Memory-augmented recurrence*, as in Transformer-XL [28], allows models to condition on cached hidden states from prior segments. More recently, *FlashAttention* [29] provides exact attention with IO-aware CUDA kernels, reducing memory bandwidth costs without sacrificing the exactness of the computation, making large context windows practically tractable.

At the model level, techniques such as *positional interpolation* and *rotary position embedding (RoPE)* scaling (used, for example, in LongChat) allow models originally trained on shorter sequences to generalize to longer ones. The Qwen2.5 and Phi-3-mini models used in our experiments both employ such extended context capabilities, supporting windows of up to 128K tokens. Nevertheless, as the "Lost in the Middle" findings confirm, large context windows are a necessary but not sufficient condition for effective long-context reasoning.

2.4.2 How Language Models Use Context

Prior to the "Lost in the Middle" paper, several works had examined the degree to which language models effectively condition on their input context. [30] showed that small LSTM language models make progressively coarser use of longer-range context when performing next-word prediction, a finding echoed by [34] in dialogue models. [32] found that longer contexts improve token prediction only marginally for most positions, consistent with the theoretical result of [33] that sequence distributions with bounded mutual information yield diminishing returns from increasingly long contexts. [37] analyzed efficient Transformer variants across a battery of long-context NLP tasks, finding persistent recency bias and poor utilization of long-range context.

[38] were among the first to demonstrate the value of retrieval context for factual question answering, showing that attaching a retrieved passage to a pre-trained

LM's input significantly improves performance on unsupervised open-domain QA. [39] extended this line of work with Fusion-in-Decoder (FiD), which encodes each retrieved passage independently and fuses the encoded representations in the decoder, achieving strong performance on NaturalQuestions-Open and TriviaQA. FiD effectively sidesteps the positional interference problem by preventing retrieved documents from directly attending to each other in the encoder — a design choice that implicitly acknowledges the hazard of mixing many documents in a single context.

2.4.3 Retrieval-Augmented Generation Systems

Following [3], numerous systems have refined the RAG paradigm. REPLUG [42] treats the language model as a black box and trains only the retriever, using the language model's perplexity on the generated answer as a training signal. In-Context RALM [43] integrates retrieval directly into the generation loop, retrieving new documents at each decoding step. These systems demonstrate that the interaction between retrieval quality and generation quality is a rich research area — and that improvements to the retriever alone can yield substantial downstream gains.

For evaluation of RAG pipelines, RAGAS [14] and ARES [16] proposed automated frameworks that separately assess retrieval faithfulness and answer quality. These multi-faceted evaluation approaches motivate our own use of multiple metrics in Chapter 3, which are designed to capture different aspects of generation quality (lexical overlap, semantic similarity, and logical entailment).

2.4.4 Positional Bias and Attention Analysis

The empirical U-shaped curve documented by [1] is well-aligned with theoretical analyses of Transformer attention. [44] introduced ALiBi (Attention with Linear Biases), a positional encoding scheme that biases attention scores toward more recent tokens, which explicitly encodes a recency preference and allows length extrapolation. [45] showed that training on shorter sequences and then testing on longer sequences introduces a length generalization gap — consistent with the observation that extended-context models are not necessarily better at using their context.

Attention patterns in instruction-tuned models have been shown to cluster around specific structural positions in the prompt (system instructions, query tokens) rather than distributing evenly over document content [1]. This structural attention bias — a consequence of the training data distribution in instruction finetuning, where system prompts and queries occupy predictable positions — may contribute directly to primacy and recency bias.

2.4.5 Evaluation of Long-Form QA

Unlike extractive QA, where evaluation reduces to substring matching, long-form QA requires assessing multi-sentence, explanatory responses. The ELI5 dataset [4], used in this work, is a canonical long-form QA benchmark, comprising complex questions from the *r/explainlikeimfive* Reddit community paired with multi-sentence crowd-sourced answers and supporting web passages.

Standard evaluation metrics for long-form QA include ROUGE-L [8], BLEU [9], and BERTScore [10]. However, as noted by [12] and others, these metrics have well-documented failure modes: semantically opposed statements can score highly under n-gram overlap metrics, and contextual embedding similarity does not robustly distinguish between contradictory but topically similar claims. These limitations motivate the proposition-level and sentence-level evaluation approaches (PropScore and SentenceScore) introduced in Chapter 3.

Recent work has moved toward LLM-as-a-judge evaluation frameworks. [17] proposed MT-Bench and Chatbot Arena for preference-based evaluation using GPT-4 as a judge. [15] introduced G-Eval, a prompt-based NLG evaluation framework using chain-of-thought reasoning to assess fluency, coherence, consistency, and relevance. While these methods offer greater semantic fidelity than n-gram metrics, they require large (20B+ parameter) models and are computationally expensive for the scale of experiments in this dissertation, motivating our hybrid approach.

FActScore [12] proposed decomposing generated answers into atomic propositions and verifying each against a knowledge source, providing a fine-grained factual precision score. DenseX Retrieval [13] demonstrated that the granularity of retrieval units (passage-level vs. proposition-level) has a significant impact on downstream QA performance, with finer-grained proposition-level units often outperforming coarser passage-level retrieval. LongRecall [18] extended this line of work with a multi-layer nugget extraction and evaluation framework specifically designed for long-form text, directly motivating the PropScore metric we introduce in Chapter 3.

2.4.6 The ELI5 Dataset

The ELI5 dataset [4] contains approximately 270K question-answer pairs sourced from the *r/explainlikeimfive* subreddit, with answers written by Reddit users and supporting documents retrieved from Common Crawl web data. The questions are intentionally open-ended, requiring explanatory multi-sentence answers rather than short factual responses. This structure makes ELI5 a substantially more demanding

testbed for RAG evaluation than extractive benchmarks like NaturalQuestions-Open: the generator must not merely retrieve a verbatim span from the context, but synthesize a coherent explanation that may draw on multiple documents or on background knowledge.

In our experiments, we use the 1K-query version of ELI5 provided by DeclareLab's *trust-align* repository [19], which includes pre-retrieved and ranked passages (100 per query) from the Common Crawl-based web corpus. Crucially, neither the "Lost in the Middle" nor the "Power of Noise" experiments have previously been conducted on this dataset (as noted in Chapter 3), making our investigation a novel contribution to the empirical understanding of these phenomena in the long-form generative setting.

2.4.7 Summary

The literature reviewed in this chapter establishes several key points that inform the experimental design of this dissertation:

- RAG pipelines are vulnerable to positional biases in the generator, not only to retrieval quality degradation. Even perfect retrieval cannot guarantee correct generation if the gold document is placed in a suboptimal position within the prompt context.
- The U-shaped positional degradation observed by [1] has been replicated across model families and task types in the extractive QA setting. Whether it persists in the long-form QA setting — where the model must synthesize rather than extract — remains an open question that this dissertation addresses.
- Noise injection as a regularization strategy for the retrieval context is a novel and counterintuitive intervention with empirical support in the extractive setting [2]. Its effects in the long-form QA setting are similarly unexplored.
- Evaluation of long-form QA is a non-trivial problem. Standard n-gram and embedding-based metrics have known failure modes, and LLM-based evaluators are computationally prohibitive at scale. The hybrid metrics proposed in this work (PropScore, SentenceScore) aim to provide a principled middle ground.

These observations directly motivate the research questions RQ1–RQ3 posed in Chapter 1 and the experimental design described in Chapter 3.

Chapter 3

Experimental Setup

3.1 Implementation Overview

This series of experiments investigates "lost in the middle" and "power of noise" phenomena in LLMs in the scope of Long-Form QA. The design of this experiment setup is heavily inspired by the findings from the paper [2], which argues that the relevance, position, and number of passages in a prompt context influence the performance of LLMs. Though the code base and the outline of these experiments are designed from scratch. These are designed to evaluate how different retrieval strategies affect the quality of LLM's response in a RAG pipeline. The task is focused on Long-Form Question Answering.

To test the "lost in the middle" and "power of noise" phenomena rigorously, we scale the experiments across three distinct context loads: 9, 19, and 29 injected documents. We also vary the position of the most relevant information within the prompt context. Also, we inject random documents into each context load in a controlled way.

Let us define a few terminologies first:

- **Gold Document:** The documents that contains the answer. We denoted these documents by G. These documents hold the most relevant information.
- **Distractors:** The documents that are retrieved via the retriever and have high lexical and semantic similarity with the query, but doesn't answer the query. We denote them by S.
- **Noise:** The documents that are completely unrelated to the query. We denote them by R.

- **Reference Answer:** These are human-written answers written by /r/explainlikeiamfive platform users. We denote these by A.
- **Query:** It's the question asked by users. We denote Query by Q.

There are 10 experiments in total, and 3 different LLMs were used; each LLM was tested in all 10 experiments. These experiments are categorized into two main categories:

- **Category A (Document Grounded):** The LLMs are given the "Gold" documents, but they are not given the explicit answer. We designed 5 different experiments in this category.
- **Category B (Answer Grounded):** The LLMs are provided with the reference answer itself, rather than just Gold documents. This category also includes 5 distinct experiments, mirroring the category A experiments in the category B setup.

The 5 experiments in **category A** are:

Exp1a: We vary the number of distractor documents (only distractors) in the context. The prompt structure is [SYS, S*k, G, Q], where k is the distractor load. Here, the distractor-load cases are: k= 0, 4, 9, 14, 19, 24, 29.

Exp2a: We vary the gold position in the context of the prompt. The context has one gold and the rest are distractors. We also vary the number of distractors to understand how gold position affects the generation across different distractor loads. The example prompt structure is [SYS, S*i, G, S*j, Q], where i+j = k.
Here, k =9, 19, 29. For k=9; i = 0, 4,9. For k=19; i = 0, 4, 9, 14, 19. And finally for k=29; i =0, 3, 7, 10, 14, 18, 22, 25, 29.

Exp3a: Vary the number of noise documents (only noise) in the context. The prompt structure is [SYS, R*k, G, Q], where k is the distractor load. Here, the noise-load cases are: k= 0, 4, 9, 14, 19, 24, 29.

1. **Exp4a:** We vary the gold position in the context of the prompt. The context has one gold and the rest are noise. We vary the noise load as well to understand how gold position affects the generation across different noise loads. The example prompt structure is [SYS, R*i, G, R*j, Q], where i+j = k.
Here, k =9, 19, 29. For k=9; i = 0, 4,9. For k=19; i = 0, 4, 9, 14, 19. And finally for k=29; i =0, 3, 7, 10, 14, 18, 22, 25, 29.

Exp5a: In this experiment, we mix distractors and noises in the context, along with gold. Along with mixing distractors and noise, we also vary the context loads. An example prompt example is [SYS, S*i, S*j, G, Q], where i+j = k.
We choose, k= 9, 19, 29.
For k=9, the (distractor, noise) composition variants are (0,9)-all noise, (2,7), (5,4), (7,2), (9,0)-all distractors.
For k=19, the compositions are: (0, 19), (2,17), (4,15), (7,12), (9,10), (11,8), (14,5), (17,2), (19,0).
For k=29, the compositions are: (0,29), (2,27), (4, 25), (7,22), (9,20), (12, 17), (14, 15), (17,12), (20, 9), (22,7), (25,4), (27, 2), (29,0).

Here, k+1 is the total context load, and for category B experiments the answer is the gold document. Also, SYS stands for system instruction.

The experiments of category B are exactly like the experiments of category A, the only difference being that instead of putting a gold document G in context, we explicitly provide the reference answer A. We replace all the G placeholders with A and name the new series of experiments **exp1b, exp2b, exp3b, exp4b, exp5b**. Each of these experiments is a replica of its corresponding category A experiment, with a category B condition.

Here, we'd like to point out that the experiment design clearly helps us look into how different context load, Gold document position, and noise influence extraction of information from the long context prompt. It partially tells us about generation quality by pointing out which information is emphasized more. However, to see the full picture, we also need to look closely at how generation itself is impacted token by token due to changes in prompt, as done in the above experiments. Here, we focused mainly on the first part, which partially gives a picture of the output quality changes.

3.2 Collection Used

In this experiment, we used the dataset ELI5 released through the paper [4] in 2019. It contains 270K complex, diverse questions that require explanatory multi-sentence answers. However, the authors didn't release the entire dataset or the mapping file due to Facebook's company policies. So, instead we use Declare-Lab's ELI5 dataset stored in GitHub repository trust-align. This version of ELI5 dataset has 1k queries.

For each query, the dataset included one of many answers written by reddit users. The dataset also have 100 passages collected from web data provided by Common Crawl. These 100 passages are the best 100 passages ranked and retrieved by a retriever. The passages are ranked as well.
We specify further that the "lost in the middle" and "power of noise" experiments were never performed on this dataset before.

3.3 Models Used

The following models are used for inference in our experiment.

Qwen2.5-7b-instruct: It is an open-weight, instruction-tuned language model released by Alibaba Cloud in 2024. It consists of 7.61 billion parameters. The model supports a massive context window of size 128k and up to 8k generation tokens per response. It supports 29 different languages, including English.[7]

Phi-3-mini-128k-instruct: This model is from Microsoft's Phi-3-mini LLM series released in 2024. It's a lightweight, fine-tuned, open model supporting a massive 128k context window. It has 3.8 billion parameters. It was trained on the Phi-3 dataset. The model can generate 4k tokens in one response.[6]

Llama-2-7b-chat-hf: It is a fine-tuned, generative text model from the Llama-2 family of models released by Meta in 2022. The model supports a context window + generation window of size 4k. It was trained on nearly 5 trillion tokens and has approximately 7 billion parameters. The model is optimized for dialogue use cases. [5]

3.4 Metrics Used for Evaluation

Since the earlier experiments performed by [1] and [2] were done for Extractive Question answering, where we match the LLM-generated answer against the reference answer, our experiment also requires ground truth to evaluate the LLM-generated answer. The evaluation metrics in our setup compare the generated answer against the ground truth. We listed the evaluation metrics we used here.

3.4.1 ROUGE-L

Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence (ROUGE-L) [8] is an NLP metric that evaluates text by measuring the longest

sequence of words shared between a model-generated text and a reference text, while maintaining word order.

$$\text{ROUGE-L}_{recall} = \frac{LCS(reference, generated)}{\text{length of reference answer}}$$

$$\text{ROUGE-L}_{precision} = \frac{LCS(reference, generated)}{\text{length of generated answer}}$$

and

$$\text{ROUGE-L}_{F1} = \frac{2 \times \text{ROUGE-L}_{recall} \times \text{ROUGE-L}_{precision}}{\text{ROUGE-L}_{recall} + \text{ROUGE-L}_{precision}}$$

For the evaluation in our experiment, we use $ROUGE - L_{F1}$ score, as both recall and precision are equally important here. Recall in important because we want to see how well the LLM extracts information from a long context prompt, and Precision tells us how much hallucination the model is making during generation using the retrieved information.

3.4.2 BERTScore

BERTScore computes token similarity using contextual embeddings from pre-trained BERT models, instead of relying on exact word matching.[10]
Suppose, $X =< x_1, x_2, ..., x_m >$ is a reference sentence and $Y = < y_1, y_2, ..., y_n >$ be a generated sentence. Also, let, $\bar{x}_i$ and $\bar{y}_i$ be the embeddings of the i-th tokens of sentence X and Y, respectively. We additionally assume, $\bar{x}_i$ and $\bar{y}_i$ are normalized, i.e., $||\bar{x}_i|| = 1 = ||\bar{y}_j||$ for all i and j.
Now, pairwise token similarity is

$$Sim(\bar{x}_i, \bar{y}_j) = x_i^T.y_j$$

The BERTScore metrics are defined as:

$$\text{BERTScore Recall: } R_{BERT} = \frac{\sum_{x_i \in X} \max_{y_j \in Y} x_i^T.y_j}{m}$$

$$\text{BERTScore Precision: } P_{BERT} = \frac{\sum_{y_j \in X} \max_{x_i \in X} x_i^T.y_j}{n}$$

and

$$\text{BERTScore F1: } F1_{BERT} = \frac{2 \times R_{BERT} \times P_{BERT}}{R_{BERT} + P_{BERT}}$$

For the same reason as mentioned in 3.4.1, we choose $F1_{BERT}$ as the metric of

interest.

3.4.3 BLEU

Bilingual Evaluation Understudy (BLEU) is an automated metric designed to evaluate the quality of model-generated text by measuring the precision of n-grams against a ground-truth reference text[9]. It factors in structural fluency and vocabulary alignment across various sequence lengths, yielding a score bounded between 0 and 1.

Given a generated candidate sentence Y and a reference sentence X, the modified n-gram precision p_n is computed by clipping candidate n-gram counts to the maximum frequency with which they appear in the reference text:

$$p_n = \frac{\sum_{\text{n-gram} \in Y} \text{Count}_{\text{clip}}(\text{n-gram})}{\sum_{\text{n-gram} \in Y} \text{Count}(\text{n-gram})}$$

To prevent short candidate generations from artificially inflating precision scores, BLEU incorporates a Brevity Penalty (BP). Let c denote the total length (word token count) of the candidate text, and r denote the length of the reference text. The penalty is defined as follows:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{\left(1 - \frac{r}{c}\right)} & \text{if } c \leq r \end{cases}$$

The standard cumulative metric (typically BLEU-4) is computed by taking the geometric mean of the modified precisions up to a maximum n-gram order N (where $N = 4$), scaled by uniform weights $w_n = \frac{1}{N}$:

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

In long-form open-ended generations, it is frequent for higher-order precisions (e.g., p_3 or p_4) to result in a zero count due to word order variations. Because a single zero value causes the geometric mean to fall to 0, we implement *Chencherry Smoothing (Method 1)*. This modification adds a microscopic dynamic value ϵ to zero-count n-grams, ensuring partial credit for valid lower-order matches is preserved.

3.4.4 Sentence-BERTScore (SBERT)

Sentence-BERTScore computes sentence similarity using contextual embedding from pre-trained models[11]. It checks the similarity of the reference answer and the generated answer at sentence level.

Suppose, $X =< x_1, x_2, ..., x_m >$ is the reference answer and $Y =< y_1, y_2, ..., y_n >$ is model generated answer. Here, x_i and y_j are sentences. SBERT processes sentences through a Siamese network architecture and applies a pooling operation to output a token embedding, resulting in a fixed-length embedding for each sentences. Let $\bar{x}_i$ and $\bar{y}_i$ be the contextual embeddings of the i-th sentence of X and Y, respectively. For the sake of simplicity, we assume, just like we did in BERTScore, that these sentence embeddings are normalized.

Now, pairwise sentence similarity is

$$Sim(\bar{x}_i, \bar{y}_j) = \bar{x}_i^T . \bar{y}_j$$

The Sentence-BERTScore metrics are defined as

$$R_{SBERT} = \frac{\sum_{x_i \in X} max_{y_j \in Y} \bar{x}_i^T . \bar{y}_j}{m}$$

$$P_{SBERT} = \frac{\sum_{y_i \in X} max_{x_j \in Y} \bar{x}_i^T . \bar{y}_j}{n}$$

$$F1_{SBERT} = \frac{2 \times R_{SBERT} \times P_{SBERT}}{R_{SBERT} + P_{SBERT}}$$

For the reasons mentioned in 3.4.4, we again choose $F1_{SBERT}$ as our metric of interest.

3.4.5 PropScore* and SentenceScore*

Motivation:

The metrics described in the earlier sections are fast, but they rely on n-gram matching or contextual similarity. Often two directly opposing statements can be scored very high based on lexical matching or contextual matching. For example "The vaccine was effective in preventing infection" and "The vaccine was ineffective in preventing infection", these two sentences will score very high in ROUGE-L and BLEU. Similarly, the two sentences have very similar contexts. So, their contextual embeddings will be close as well. The problem worsens further when we need to

evaluate the similarity of two answers, consisting of multiple sentences.
To mitigate this, authors of FactScore[12] suggested breaking the answers down in propositions or nuggets and then evaluating the quality of propositions in model-generated answers. In DenseX Retrieval[13] we see how granularity of information impacts retrieval via LLMs. Systems such as RAGAS [14] and ARES [16] isolate the performance of the retriever from the faithfulness of the generator, establishing a multi-faceted approach to RAG assessment. Building upon these works, LongRecall[18] builds an elegant multi-layer architecture that extracts nuggets, filters, and evaluates these nuggets using an LLM.
Many such LLM-based evaluation metrics evolved, making incremental improvements in this area, such as LLM-as-a-judge [17], G-eval[15]. Still, a serious bottleneck remains in all LLM-based approaches, when we feed the propositions into LLMs, i.e., to get a reasonably good score, we need models with 20-30 billion parameters at least, making the task computationally expensive.
Keeping these bottlenecks in view, we introduce a new evaluation approach that provides a middle ground between the computational simplicity of non-LLM-based metrics and the rigorous semantic grounding of LLM-based evaluators. We introduce **PropScore**, which works on the nugget level and its sentence version **SentenceScore** in the following discussion.

** *We use the terms nuggets and propositions alternatively here. They mean the same thing.*

PropScore:

The calculation of PropScore for a set of N evaluation queries, $\mathcal{Q} = \{q_i\}_{i=1}^{N}$, follows a structured pipeline as detailed below.

1. Setup and Notations: For each query q_i, let:

- $R_i = \{r_{i1}, r_{i2}, ..., r_{in_i}\}$ denote the set of n_i nuggets extracted from the reference answer.
- $G_i = \{g_{i1}, g_{i2}, ..., g_{im_i}\}$ denote the set of m_i nuggets extracted from the LLM-generated answer.

. The nuggets are extracted using a Language Model, as done in earlier works. To extract nuggets we used Qwen2.5-3B-Instruct.[7]

2. Pairwise Proposition Scoring: For every pair of reference and generated propositions (r_{ij}, g_{ik}), a scalar Proposition Score $PS(r_{ij}, g_{ik}) \in [0, 1]$ is calculated to fuse semantic similarity with bidirectional logical entailment.

2.1 Cosine Similarity:

First, the raw cosine similarity between the embeddings of the proposition pair is computed using a pretrained dense embedding model E:

$$Sim(r_{ij}, g_{ik}) = \frac{E(r_{ij}) \cdot E(g_{ik})}{||E(r_{ij})||_2 \cdot ||E(g_{ik})||_2}$$

We used the embedding model BAAI/bge-large-en-v1.5, which is specifically optimized for English. [24]

2.2 Adjusted Similarity:

To discard semantically opposed or orthogonal pairs, this similarity is adjusted with a hard floor at zero:

$$AdjSim(r_{ij}, g_{ik}) = \begin{cases} Sim(r_{ij}, g_{ik}) & \text{if } Sim(r_{ij}, g_{ik}) > 0 \\ 0 & \text{otherwise} \end{cases}$$

2.3 Symmetric Entailment Score:

Next, a Natural Language Inference (NLI) cross-encoder evaluates logical consistency to compute the bidirectional entailment probabilities P_f (forward) and P_b (backward).

$$P_f(r_{ij}, g_{ik}) = \mathbb{P}(Entail|\ r_{ij} \rightarrow g_{ik})$$

and

$$P_b(r_{ij}, g_{ik}) = \mathbb{P}(Entail|\ r_{ij} \leftarrow g_{ik})$$

The symmetric entailment score is defined conservatively as the minimum of both directions, ensuring one-sided entailments are appropriately penalized:

$$P_{sym}(r_{ij}, g_{ik}) = \min\{P_f(r_{ij}, g_{ik}), P_b(r_{ij}, g_{ik})\}$$

We used cross-encoder/nli-deberta-v3-large for Natural Language Inference (NLI) task.[23]

2.4 Proposition Score:

The final pairwise Proposition Score uses a non-linear exponential penalty to

combine the adjusted similarity and symmetric entailment:

$$PS(r_{ij}, g_{ik}) = \left[\frac{AdjSim(r_{ij}, g_{ik}) \cdot (1 + P_{sym}(r_{ij}, g_{ik}))}{2}\right]^{1+\gamma(1-P_{sym}(r_{ij},g_{ik}))}$$

,where $\gamma \geq 0$ is a tunable parameter. When $P_{sym} = 1$ (full bidirectional entailment), the base equals AdjSim and the exponent collapses to 1, so $PS = AdjSim$. When $P_{sym} = 0$ (no entailment), the base is halved and the exponent becomes $1 + \gamma = 1$ and the base $\frac{1+P_{sym}(r_{ij},g_{jk})}{2} = 0.5$, aggressively suppressing the score even for semantically close but logically ungrounded pairs.

3. Top-p DCG Aggregation: To evaluate how well a specific reference proposition r_{ij} is covered by the generated text, we aggregate its pairwise scores against all generated propositions.

3.1 Ranked Score Sequence:

For a fixed reference proposition r_{ij}, compute its PS against every generated proposition and sort the resulting scores in **descending** order:

$$(t_1, t_2, ..., t_{m_i}), \text{ where } t_1 \geq t_2 \geq .... \geq t_{m_i} \text{ and } t_k = PS(r_{ij}, g_{i\sigma(k)})$$

. Here, σ is a permutation of the set $\{1, 2, ..., m_i\}$

3.2 Aggregated Proposition Score:

The Aggregated Proposition Score is calculated using a Discounted Cumulative Gain (DCG) sum over the top-p matches:

$$AggPS(r_{ij}) = \sum_{k=1}^{p} \frac{t_k}{\log_2(k+1)}$$

The discount $1/\log_2(1+k)$ assigns the highest weight to the best-matching generated proposition (weight $1/\log_2(2) = 1$ at rank 1) with diminishing returns for subsequent matches. This handles the case where one reference fact is covered by multiple generated propositions or duplicate proposition from the generated answer.

4. Query-Level Metric: The PropScore for query q_i averages the aggregated proposition scores over all n_i reference propositions:

$$PropScore(q_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} AggPS(r_{ij})$$

In PropScore, we have two hyperparameters: **(a) γ and (b) p**. We suggest tuning γ on a sample of the data. For intuition, γ represents degree of strictness in judgment. In our experiments, we used $\gamma = 9$, as the metric performed best on a sample for this value of γ The hyper-parameter p can also be tuned as per use case, but we used $p = \sqrt{m_i}$ for our experiments.

Concerns and Limitations: Even though our proposed metric shows promise theoretically, there are a few common concerns and limitations, we must notice.

- The metric itself is heavily reliant on the quality of propositions generated by Language Models. If the quality of proposition is poor the metric itself will suffer inaccuracy in scoring. In the next section we will see that *Propscore* failed to outperform the baselines. *PropScore* underperformed in both Kendall τ and Pearson Correlation ρ . But, we did an ablation study that led us to believe that the quality of extracted proposition caused poor performance, not the metric itself.
- The metric can be too aggressively penalize extra, but helpful details generated by a LLM, due to the symmetric entailment probability.
- The metric requires ground truth for evaluation, and Long-Form QA paradigm factual correctness independent of ground truth is often considered more useful.

SentenceScore:

The computation pipeline for SentenceScore is same as PropScore, but here we break the reference and generated answers into sentences, instead of extracting propositions using a Language Model, i.e.,

R_i = collection of all the sentences in the reference answer of the query q_i

and

G_i = collection of all the sentences in the model generated answer of the query q_i

Mimic the rest of the pipeline for computing $PropScore(q_i)$ to compute $SentenceScore(q_i)$ for query q_i.

Concerns and Limitations:

- SentenceScore is significantly faster than PropScore, since, we remove the LLM heavy proposition extraction part, however, it can become blind to acceptable abstraction, i.e., different phrasing explaining the same facts. Even then, SentenceScore performed significantly better at ranking than all the baselines in our experimental setup. We will see in the next section that SentenceScore improved the **Kendall** τ significantly compared to all other baselines; while it fell short in **Pearson Correlation** ρ. We clarify, that we didn't check it's performance in other datasets. We can't claim the performance enhancement to be global yet. We are still working on this task.
- SentenceScore too can penalize useful information absent in ground truth.
- SentenceScore too requires Ground truth, just like PropScore. We are also working on modifying the metric, so the metric can combine ground truth along with factual correctness and usefulness, without relying on the ground truth entirely.

Chapter 4

Results and Discussion

I this section, we will discuss the findings of our work, and try to understand why these findings appeared in our results.
Earlier, we discussed isolating our experiments into two categories:

- Category A
- Category B

We will look at the results in this fashion: expia and expib results will be shown together for better comparison

4.1 exp1a and exp1b Results

In Document-Grounded experiments, we feed the LLMs Gold-documents, and we explicitly keep the reference answer held out.

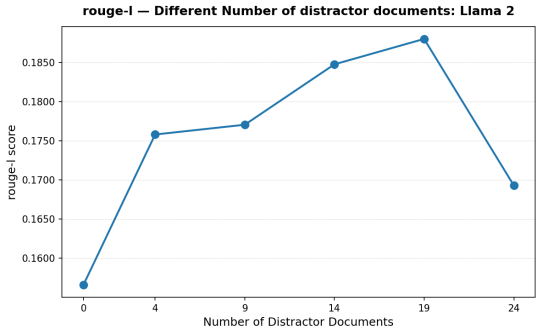
4.1.1 Model: Llama-2-7b-chat-hf

Distractors	ROUGE-L	BERTScore	SBERT	BLEU	PropScore-7	SentenceScore-9
0	0.1566	0.8492	0.6607	0.0195	0.0391	0.0123
4	0.1758	0.8571	0.7022	0.0285	0.0530	0.0223
9	0.1770	0.8578	0.7084	0.0301	0.0538	0.0206
14	0.1848	0.8636	0.7300	0.0319	0.0541	0.0191
19	0.1880	0.8659	0.7338	0.0323	0.0616	0.0209
24	0.1693	0.8589	0.7277	0.0254	0.0510	0.0537

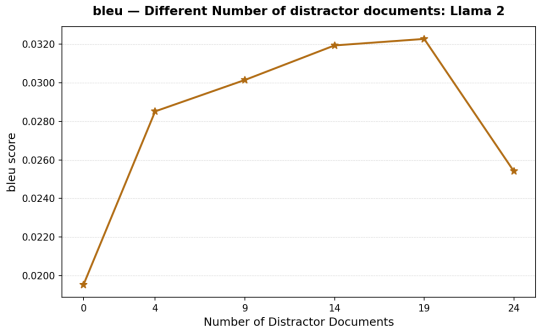
Table 4.1: **Exp1a Results -** Performance metrics across varying distractor counts in Document-Grounded Scenario. PropScore is reported at penalty strictness $\gamma = 7$ and SentenceScore at $\gamma = 9$.

Distractors	ROUGE-L	BERTScore	SBERT	BLEU	PropScore ($\gamma = 7$)	SentenceScore ($\gamma =$
0	0.2122	0.8680	0.6405	0.0471	0.0688	0.0461
4	0.3323	0.8932	0.7297	0.1516	0.1561	0.1316
9	0.3907	0.9018	0.7479	0.2260	0.2220	0.1974
14	0.4175	0.9067	0.7661	0.2606	0.2435	0.2331
19	0.4082	0.9042	0.7681	0.2569	0.2548	0.2409
24	0.4280	0.9060	0.7618	0.2798	0.2475	0.2410

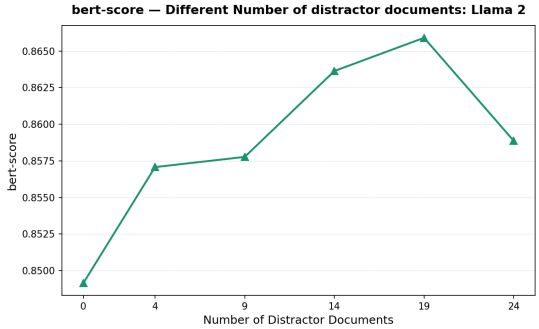
Table 4.2: **Exp1b Results -** Performance metrics across varying distractor counts in Answer-Grounded scenario. PropScore is reported at penalty strictness $\gamma = 7$ and SentenceScore at $\gamma = 9$.



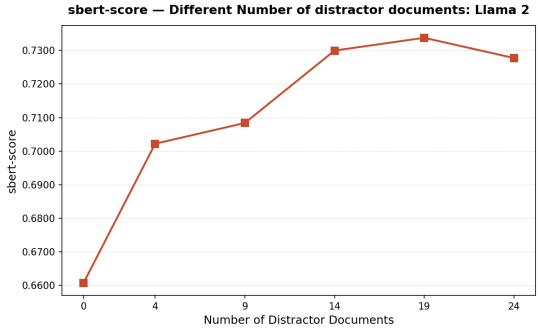
(a) ROUGE-L



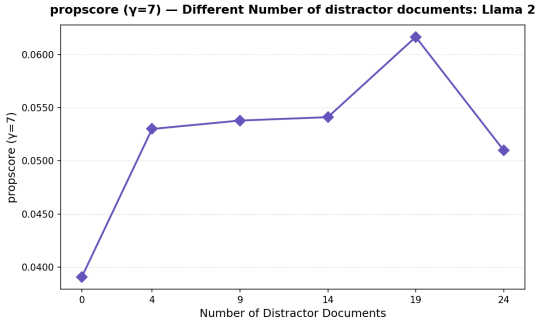
(b) BLEU



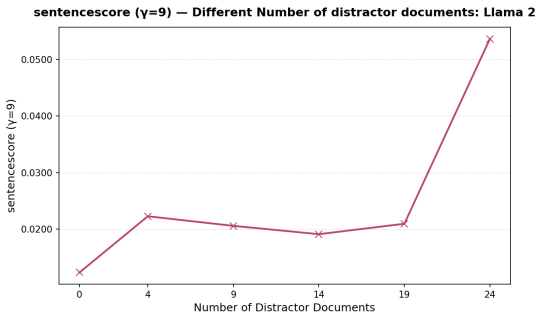
(c) BERTScore



(d) Sentence-BERT (SBERT)

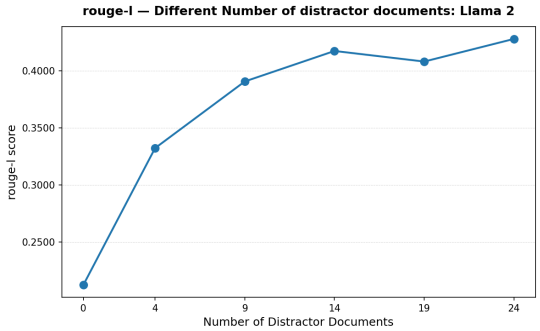


(e) PropScore ($\gamma = 7$)

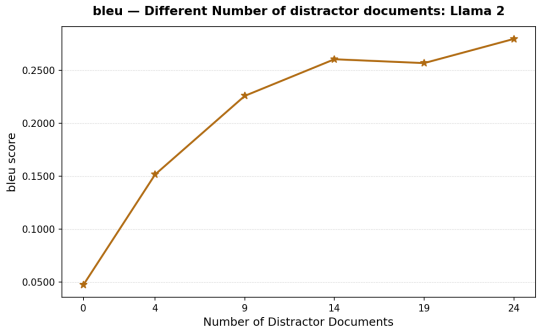


(f) SentenceScore ($\gamma = 9$)

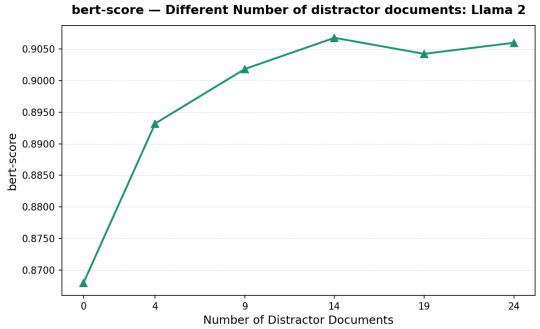
Figure 4.1: Category A (Document Grounded): Comprehensive metric comparison as distractor count increases. All metrics increase as distractor count increases, most peaks at 14 or 19, then flattens (except sentence score).



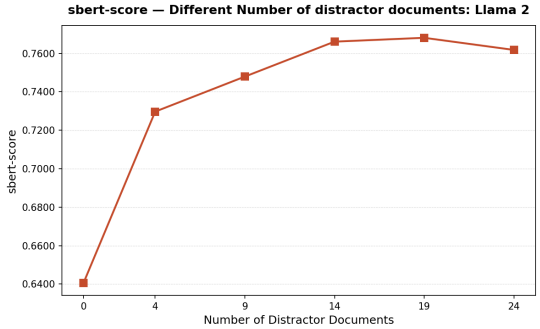
(a) ROUGE-L



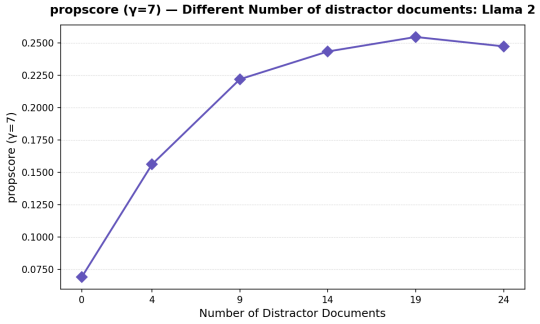
(b) BLEU



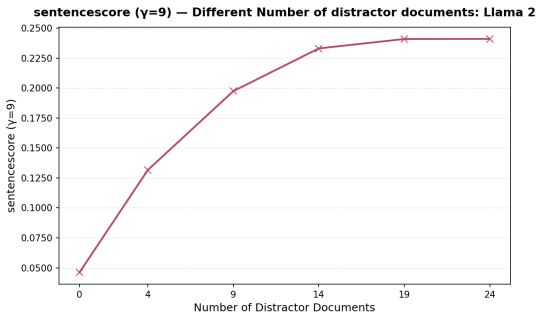
(c) BERTScore



(d) Sentence-BERT (SBERT)



(e) PropScore ($\gamma = 7$)



(f) SentenceScore ($\gamma = 9$)

Figure 4.2: Category B (Answer Grounded): Comprehensive metric comparison as distractor count increases. All metrics increase as distractor count increases, then flattens.

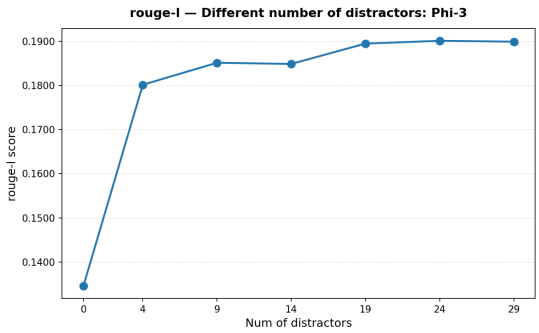
4.1.2 Model: Phi-3-mini

Distractors	ROUGE-L	BERTScore	SBERT	BLEU	PropScore-7	SentenceScore-9
0	0.1346	0.8416	0.6848	0.0177	0.0676	0.0393
4	0.1801	0.8649	0.6793	0.0268	0.0505	0.0123
9	0.1851	0.8649	0.6891	0.0280	0.0460	0.0132
14	0.1849	0.8685	0.6912	0.0259	0.0515	0.0130
19	0.1895	0.8665	0.6853	0.0307	0.0461	0.0130
24	0.1901	0.8666	0.6898	0.0285	0.0429	0.0112
29	0.1899	0.8682	0.6864	0.0276	0.0447	0.0101

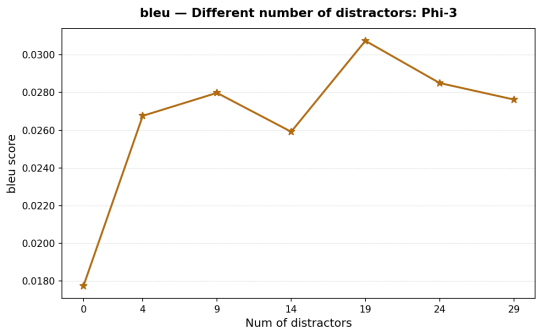
Table 4.3: **Exp1a Result -** Performance metrics across varying distractor document counts in Document-Grounded scenario. PropScore is reported at penalty strictness $\gamma = 7$ and SentenceScore at $\gamma = 9$.

Noise Docs	ROUGE-L	BERTScore	SBERT	BLEU	PropScore-7	SentenceScore-9
0	0.1588	0.8497	0.7649	0.0510	0.2483	0.3286
4	0.2875	0.8868	0.6769	0.0906	0.1067	0.0680
9	0.3108	0.8930	0.6927	0.1146	0.1159	0.0864
14	0.3165	0.8911	0.6894	0.1246	0.1381	0.0948
19	0.3217	0.8924	0.6904	0.1342	0.1411	0.1118
24	0.3414	0.8944	0.7055	0.1530	0.1518	0.1224
29	0.3424	0.8941	0.7000	0.1566	0.1556	0.1248

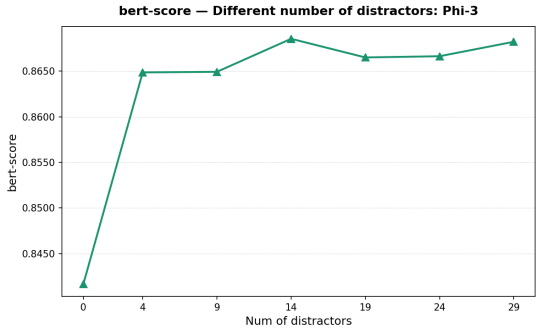
Table 4.4: **Exp1b Results-**Performance metrics across varying distractor document counts in Answer-Grounded scenarios. PropScore is reported at penalty strictness $\gamma = 7$ and SentenceScore at $\gamma = 9$.



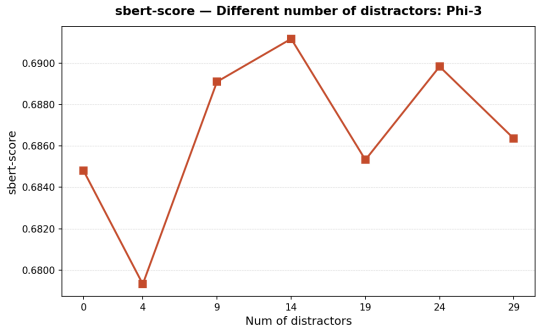
(a) ROUGE-L



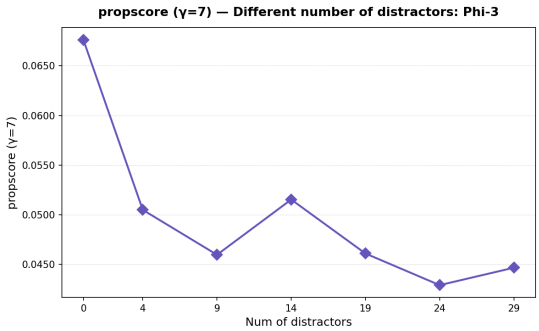
(b) BLEU



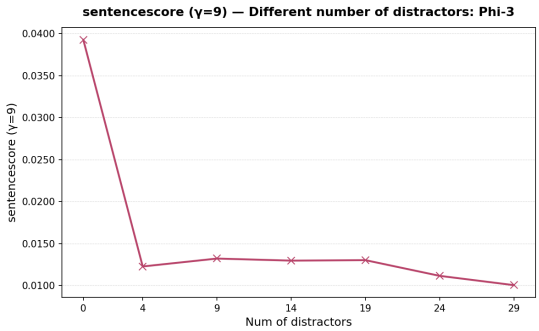
(c) BERTScore



(d) Sentence-BERT (SBERT)

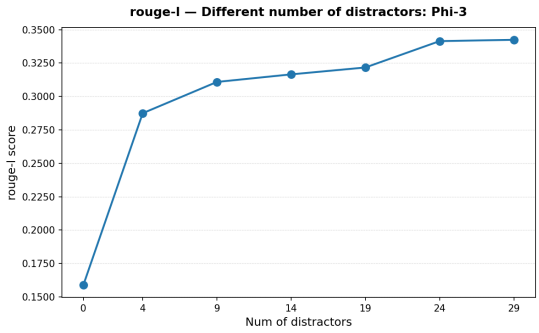


(e) PropScore ($\gamma = 7$)

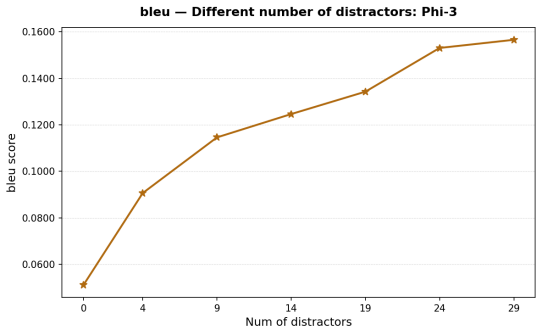


(f) SentenceScore ($\gamma = 9$)

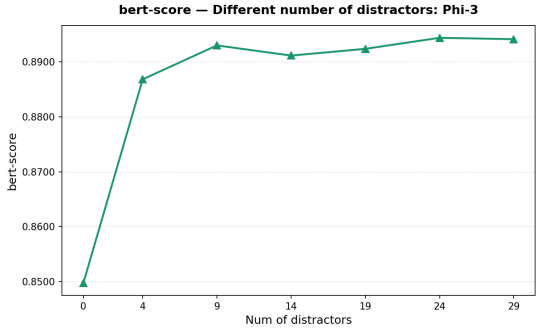
Figure 4.3: Category A (Document Grounded): Comprehensive metric comparison as distractor count increases. Traditional metrics (Rows 1 and 2) show artificial improvement, while logically constrained metrics (Row 3) accurately capture the context degradation.



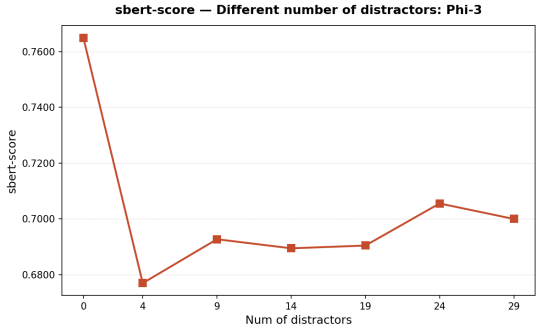
(a) ROUGE-L



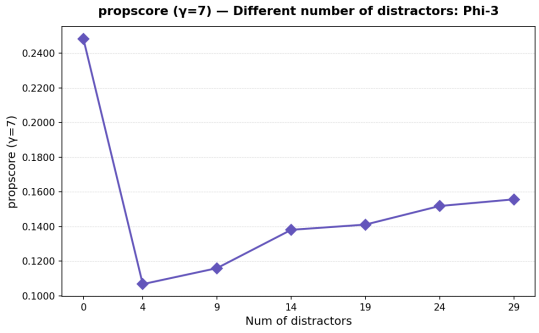
(b) BLEU



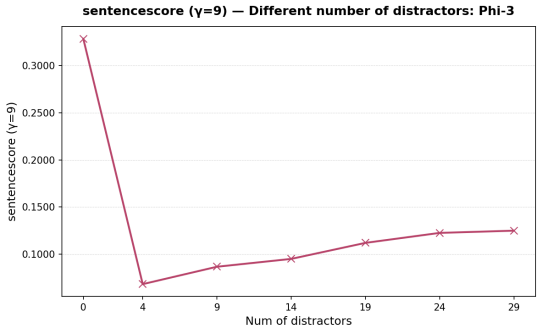
(c) BERTScore



(d) Sentence-BERT (SBERT)



(e) PropScore ($\gamma = 7$)



(f) SentenceScore ($\gamma = 9$)

Figure 4.4: Category B (Answer Grounded): Comprehensive metric comparison as distractor count increases. Traditional metrics (Rows 1 and 2) show artificial improvement, while logically constrained metrics (Row 3) accurately capture the context degradation. We also witness sbert to change it's behaviour and show the degradation, which is closer to logically constrained metrics.

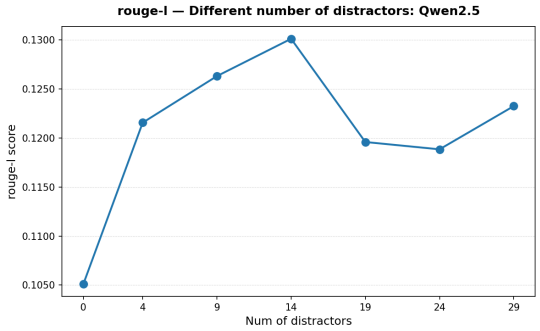
4.1.3 Model: Qwen2.5-7b-instruct

Distractors	ROUGE-L	BERTScore	SBERT	BLEU	PropScore-7	SentenceScore-7
0	0.1051	0.8300	0.7098	0.0131	0.0744	0.0410
4	0.1216	0.8390	0.7068	0.0152	0.0646	0.0241
9	0.1263	0.8414	0.7155	0.0177	0.0706	0.0303
14	0.1301	0.8433	0.7202	0.0185	0.0724	0.0267
19	0.1196	0.8384	0.7175	0.0154	0.0720	0.0319
24	0.1188	0.8375	0.7192	0.0165	0.0760	0.0320
29	0.1232	0.8385	0.7191	0.0163	0.0748	0.0236

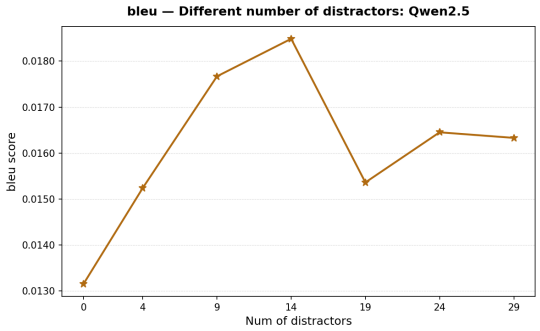
Table 4.5: **Exp1a Result -** Performance metrics across varying distractor document counts in Document-Grounded scenario. PropScore is reported at penalty strictness $\gamma = 7$ and SentenceScore at $\gamma = 9$.

Distractors	ROUGE-L	BERTScore	SBERT	BLEU	PropScore-7	SentenceScore-9
0	0.1403	0.8473	0.7627	0.0409	0.2146	0.1721
4	0.1437	0.8474	0.7183	0.0328	0.1179	0.0852
9	0.1476	0.8482	0.7197	0.0342	0.1196	0.0741
14	0.1520	0.8489	0.7181	0.0383	0.1286	0.0855
19	0.1475	0.8465	0.7159	0.0354	0.1160	0.0798
24	0.1398	0.8439	0.7180	0.0343	0.1292	0.0943
29	0.1456	0.8453	0.7205	0.0377	0.1417	0.0956

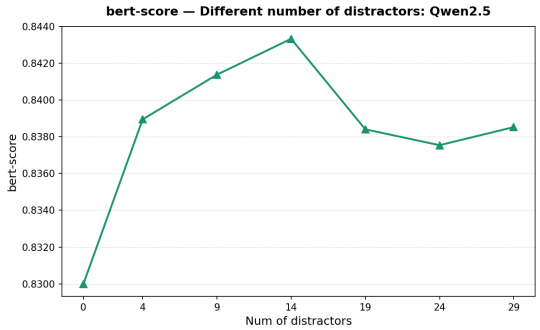
Table 4.6: **Exp1b Result -** Performance metrics across varying distractor document counts in Answer-Grounded scenario. PropScore is reported at penalty strictness $\gamma = 7$ and SentenceScore at $\gamma = 9$.



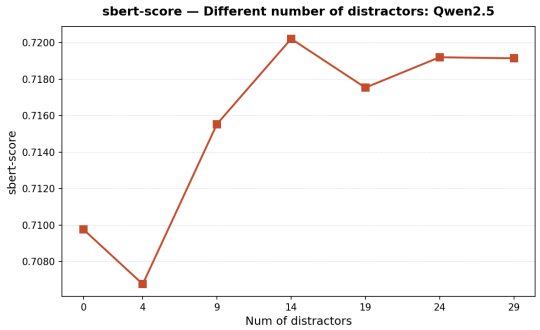
(a) ROUGE-L



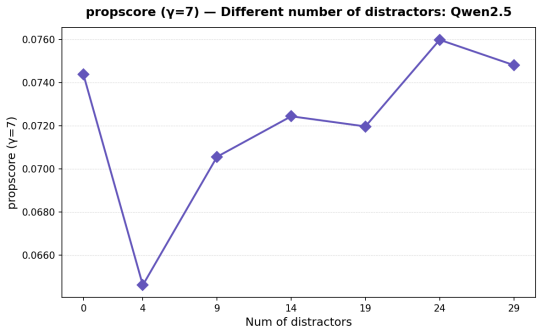
(b) BLEU



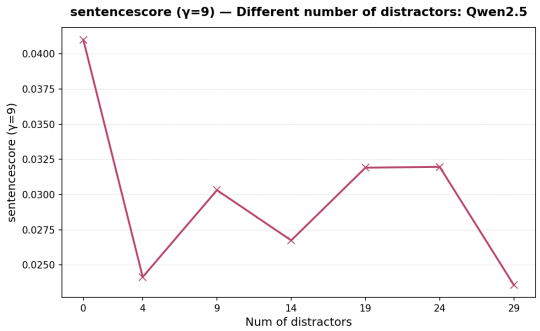
(c) BERTScore



(d) Sentence-BERT (SBERT)

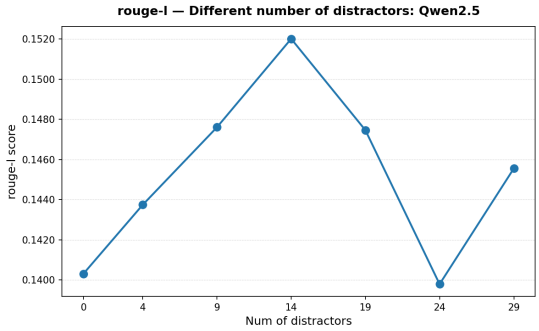


(e) PropScore ($\gamma = 7$)

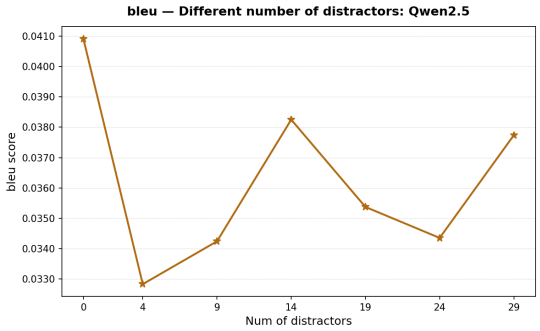


(f) SentenceScore ($\gamma = 9$)

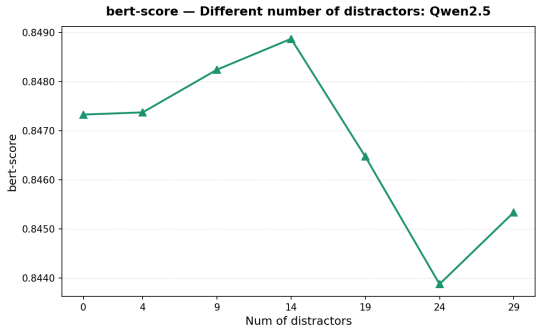
Figure 4.5: Category A (Document Grounded): Comprehensive metric comparison as distractor count increases. ROUGE-L, BLEU and BERTScore show artificial improvement, Sentence-BERTScore and PropScore drops initially and then increases again, while sentence score accurately capture the context degradation.



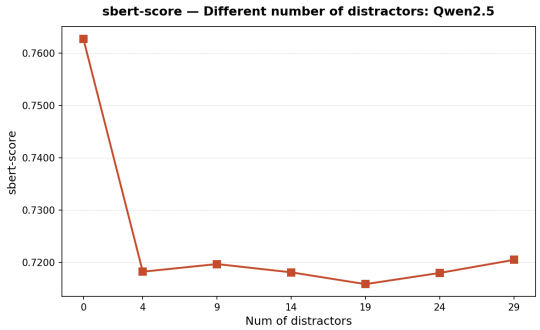
(a) ROUGE-L



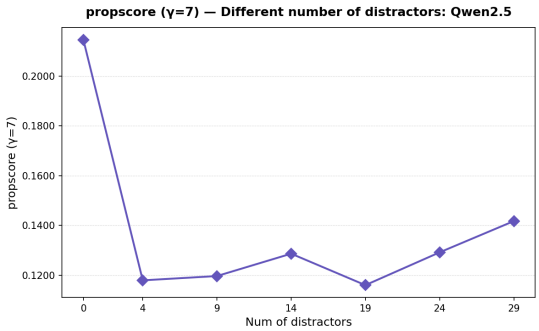
(b) BLEU



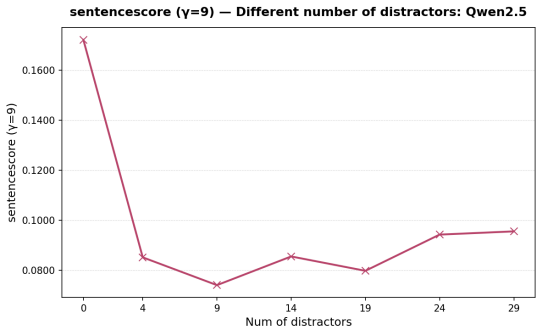
(c) BERTScore



(d) Sentence-BERT (SBERT)



(e) PropScore ($\gamma = 7$)



(f) SentenceScore ($\gamma = 9$)

Figure 4.6: Category A (Document Grounded): Comprehensive metric comparison as distractor count increases. ROUGE-L, BLEU and BERTScore show artificial improvement. Sentence-BERTScore and PropScore and sentence score accurately capture the context degradation.

4.2 exp2a and exp2b Results

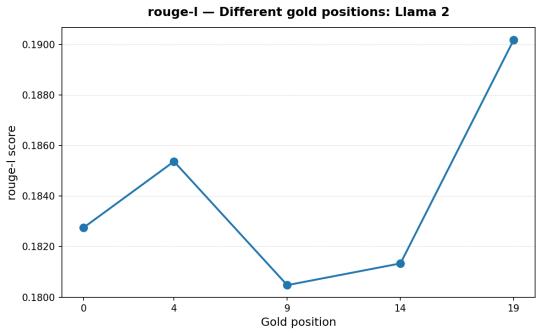
4.2.1 Llama-2-7b-chat-hf

Dist	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1813	0.8573	0.7101	0.0293	0.0523	0.0173
	4	0.1754	0.8561	0.7039	0.0277	0.0498	0.0206
	9	0.1788	0.8577	0.7093	0.0302	0.0510	0.0195
19	0	0.1827	0.8610	0.7249	0.0308	0.0495	0.0192
	4	0.1854	0.8645	0.7279	0.0315	0.0556	0.0205
	9	0.1805	0.8631	0.7258	0.0294	0.0572	0.0214
	14	0.1813	0.8646	0.7283	0.0311	0.0588	0.0192
	19	0.1902	0.8659	0.7320	0.0332	0.0604	0.0209

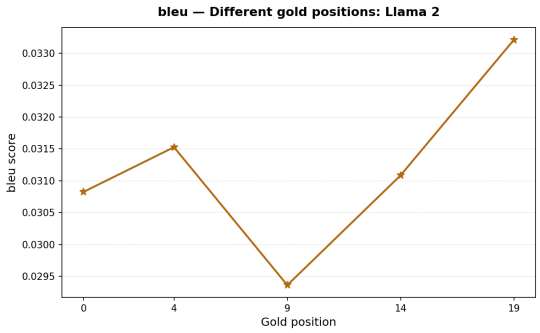
Table 4.7: **Exp2a Results -** Performance metrics across varying gold document positions within 9-distractor and 19-distractor context windows.

Dist	Gold Position	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.4030	0.8960	0.7626	0.2481	0.2684	0.2334
	4	0.3466	0.8870	0.7279	0.1995	0.2036	0.1821
	9	0.3877	0.9015	0.7519	0.2200	0.2114	0.1922
19	0	0.3890	0.8984	0.7639	0.2434	0.2505	0.2245
	4	0.3500	0.8912	0.7445	0.2148	0.2270	0.2031
	9	0.3625	0.8939	0.7474	0.2266	0.2309	0.2139
	14	0.3472	0.8903	0.7460	0.2081	0.2123	0.1858
	19	0.4094	0.9058	0.7668	0.2567	0.2494	0.2352

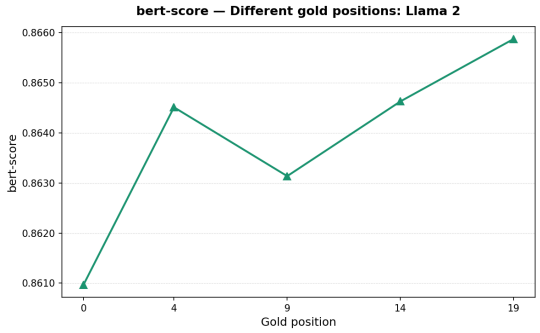
Table 4.8: **Exp2b Results -** Performance metrics across varying gold document positions within 9-distractor and 19-distractor context windows.



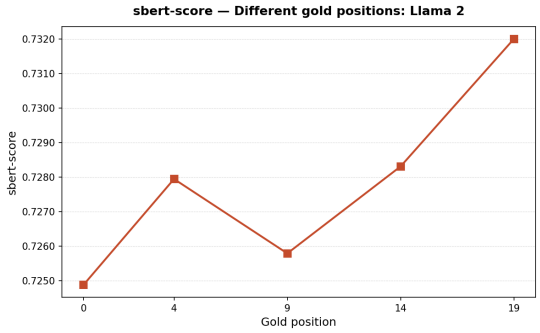
(a) ROUGE-L



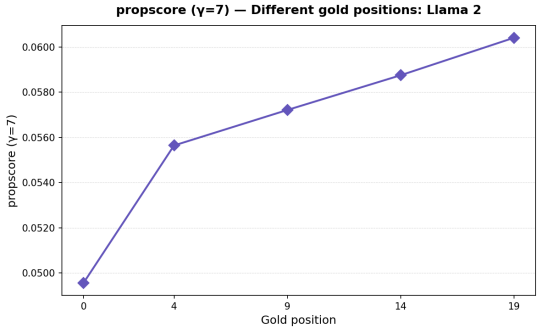
(b) BLEU



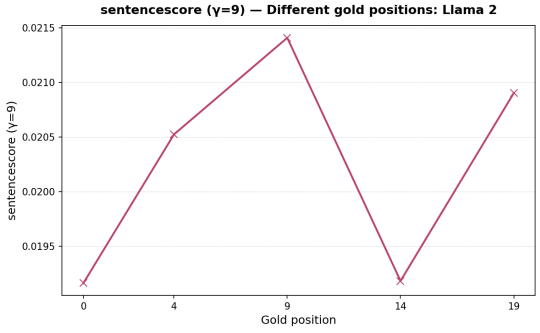
(c) BERTScore



(d) Sentence-BERT (SBERT)

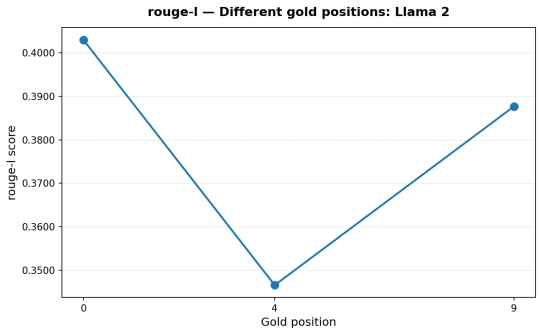


(e) PropScore ($\gamma = 7$)

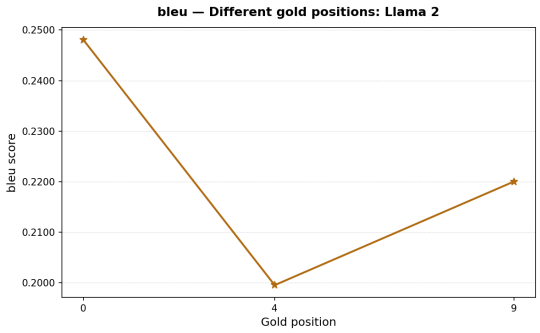


(f) SentenceScore ($\gamma = 9$)

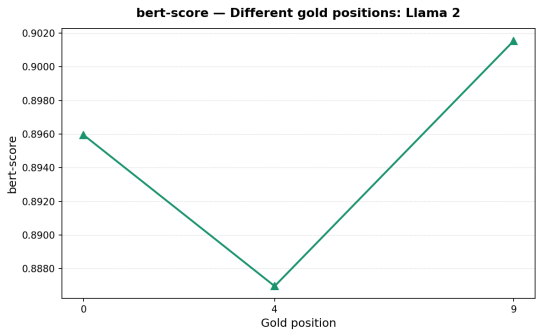
Figure 4.7: Category A (Document Grounded, 9 distractors): Comprehensive metric comparison as gold position increases. ROUGE-L, BLEU show mild "lost in the middle effect with high recency bias; BERTScore and SBERTScore starts with lowest value, then dips followed by an increase, then the metric gradually increases and peaks at last position; PropScore increases linearly, while SentenceScore peaks at the middle.



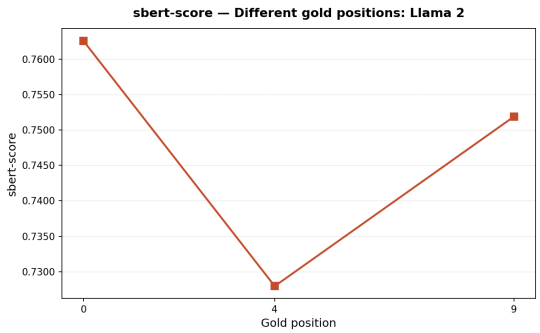
(a) ROUGE-L



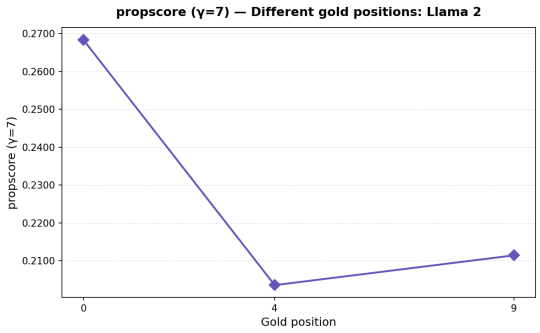
(b) BLEU



(c) BERTScore



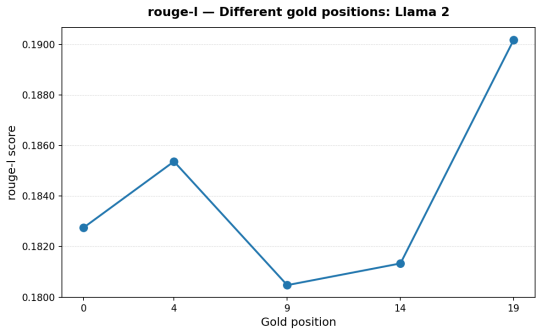
(d) Sentence-BERT (SBERT)



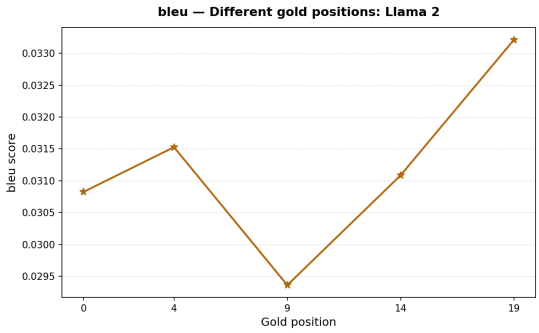
(e) PropScore ($\gamma = 7$)

(f) SentenceScore ($\gamma = 9$)

Figure 4.8: Category B (Answer Grounded, 9 distractors): Comprehensive metric comparison as distractor count increases. All the metrics indicate a strong "lost in the middle" effect with high primacy bias.



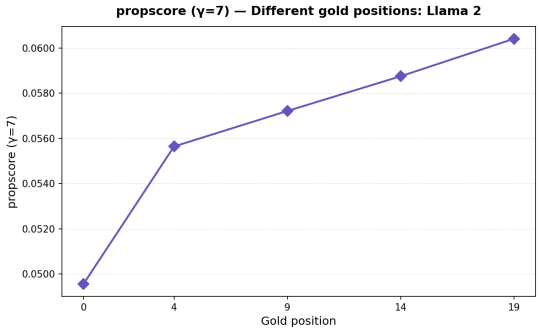
(a) ROUGE-L



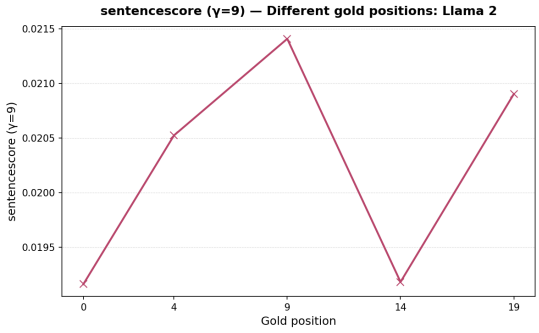
(b) BLEU

(c) BERTScore

(d) Sentence-BERT (SBERT)

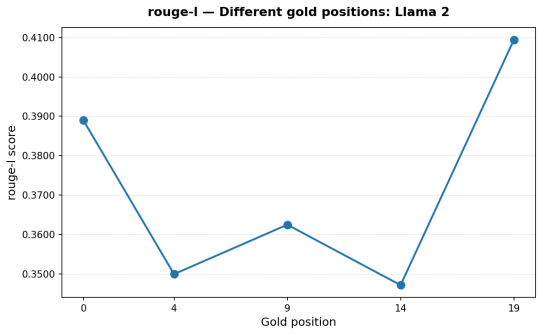


(e) PropScore ($\gamma = 7$)

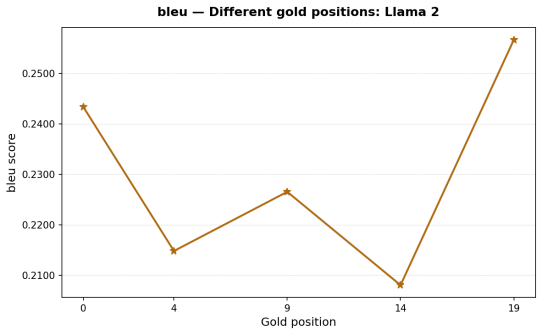


(f) SentenceScore ($\gamma = 9$)

Figure 4.9: Category A (Document Grounded, 19 distractors): Comprehensive metric comparison as distractor count increases. ROUGE-L, BLEU and BERTScore show artificial improvement, Sentence-BERTScore and PropScore drops initially and then increases again, while sentence score accurately capture the context degradation.



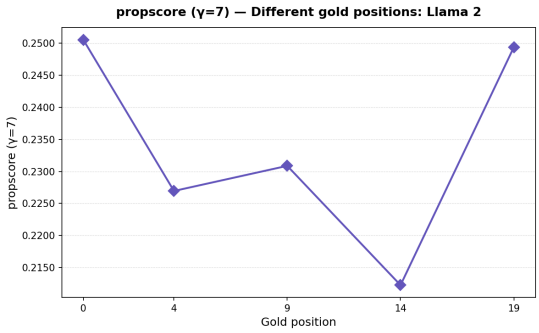
(a) ROUGE-L



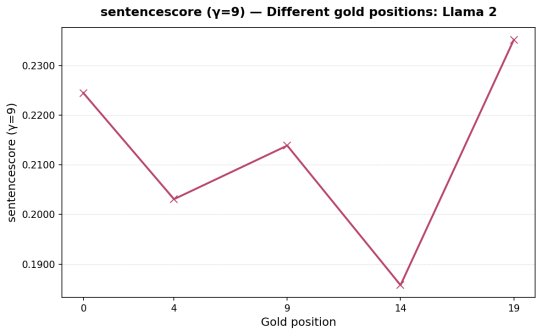
(b) BLEU

(c) BERTScore

(d) Sentence-BERT (SBERT)



(e) PropScore ($\gamma = 7$)



(f) SentenceScore ($\gamma = 9$)

Figure 4.10: Category B (Answer Grounded, 19 distractors): Comprehensive metric comparison as distractor count increases. All the metrics indicate a strong "lost in the middle" effect with high primacy and recency bias.

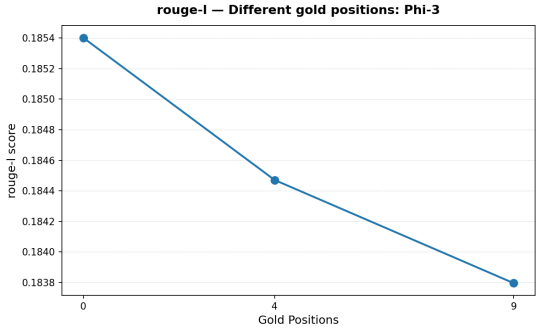
4.2.2 Model: Phi-3-mini

Dist	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore-7	SentenceScore-9
9	0	0.2416	0.8779	0.7054	0.0691	0.1101	0.0747
	4	0.2904	0.8861	0.7019	0.1092	0.1293	0.0998
	9	0.3088	0.8927	0.6916	0.1134	0.1221	0.0885
9	0	0.3193	0.8937	0.7220	0.1100	0.1542	0.0904
	4	0.3893	0.9049	0.7229	0.1791	0.1985	0.1537
	9	0.3371	0.8961	0.6834	0.1271	0.1367	0.1021
29	0	0.2500	0.8797	0.6952	0.0781	0.1028	0.0663
	3	0.2833	0.8823	0.6922	0.1135	0.1197	0.0854
	7	0.2901	0.8834	0.6921	0.1231	0.1237	0.1026
	10	0.2848	0.8830	0.6930	0.1165	0.1273	0.0994
	14	0.2996	0.8842	0.6915	0.1326	0.1416	0.1154
	18	0.2976	0.8826	0.6954	0.1293	0.1373	0.1094
	22	0.3077	0.8863	0.7008	0.1389	0.1432	0.1228
	25	0.3134	0.8868	0.6949	0.1442	0.1350	0.1204
	29	0.3451	0.8960	0.6969	0.1611	0.1563	0.1329

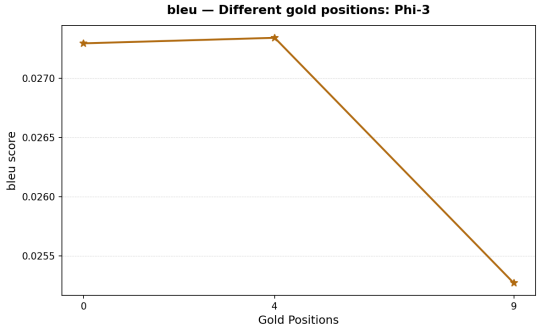
Table 4.9: **Exp2a Results -** Performance metrics across varying gold document positions within 9-distractor, 9-noise-document, and 29-distractor context windows.

Dist	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.2416	0.8779	0.7054	0.0691	0.1101	0.0747
	4	0.2904	0.8861	0.7019	0.1092	0.1293	0.0998
	9	0.3088	0.8927	0.6916	0.1134	0.1221	0.0885
19	0	0.2440	0.8772	0.6904	0.0760	0.1003	0.0715
	4	0.2935	0.8844	0.6913	0.1200	0.1351	0.1088
	9	0.2908	0.8836	0.6929	0.1200	0.1194	0.1001
	14	0.3055	0.8855	0.6977	0.1341	0.1420	0.1173
	19	0.3214	0.8927	0.6872	0.1351	0.1434	0.1135
29	0	0.2500	0.8797	0.6952	0.0781	0.1028	0.0663
	3	0.2833	0.8823	0.6922	0.1135	0.1197	0.0854
	7	0.2901	0.8834	0.6921	0.1231	0.1237	0.1026
	10	0.2848	0.8830	0.6930	0.1165	0.1273	0.0994
	14	0.2996	0.8842	0.6915	0.1326	0.1416	0.1154
	18	0.2976	0.8826	0.6954	0.1293	0.1373	0.1094
	22	0.3077	0.8863	0.7008	0.1389	0.1432	0.1228
	25	0.3134	0.8868	0.6949	0.1442	0.1350	0.1204
	29	0.3451	0.8960	0.6969	0.1611	0.1563	0.1322

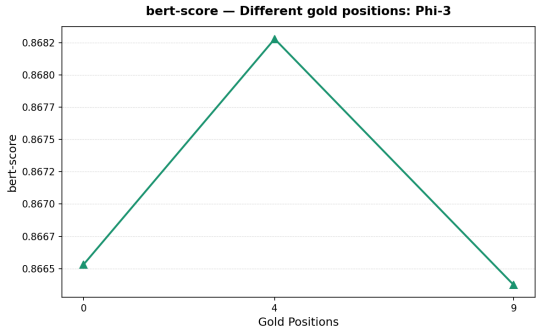
Table 4.10: **Exp2b Results -** Performance metrics across varying gold document positions within 9-distractor, 19-distractor, and 29-distractor context windows.



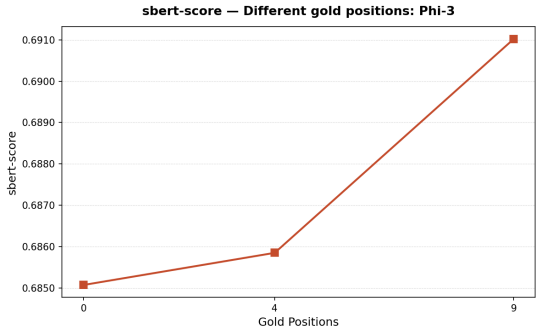
(a) ROUGE-L



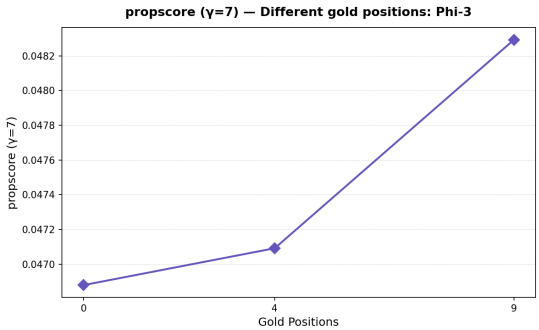
(b) BLEU



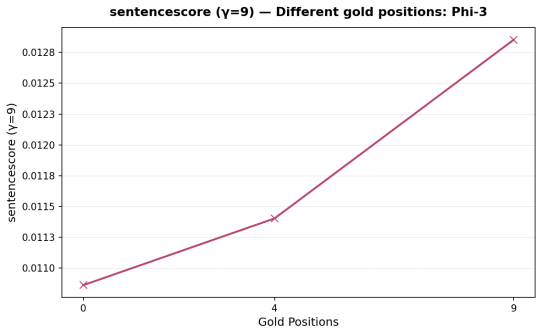
(c) BERTScore



(d) Sentence-BERT (SBERT)

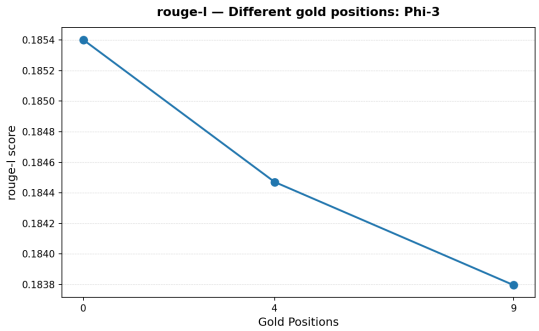


(e) PropScore ($\gamma = 7$)

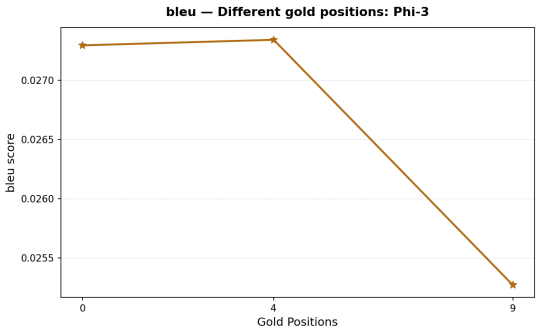


(f) SentenceScore ($\gamma = 9$)

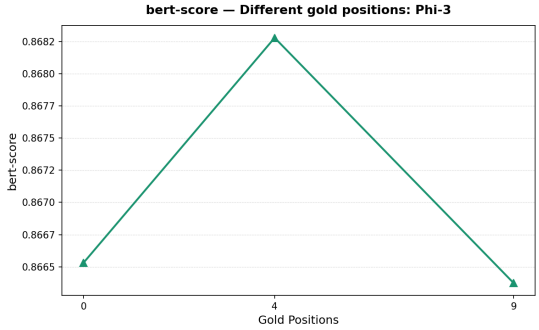
Figure 4.11: Category A (Document Grounded, 9 distractors): Comprehensive metric comparison as gold position increases. ROUGE-L, BLEU show mild "lost in the middle effect with high recency bias; BERTScore and SBERTScore starts with lowest value, then dips followed by an increase, then the metric gradually increases and peaks at last position; PropScore increases linearly, while SentenceScore peaks at the middle.



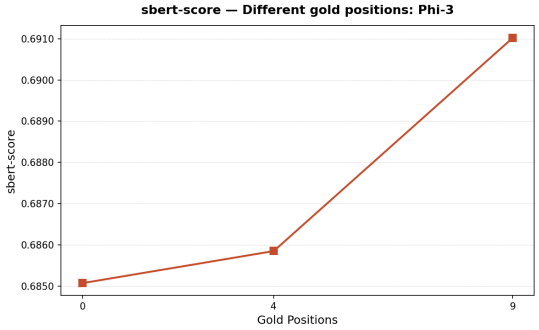
(a) ROUGE-L



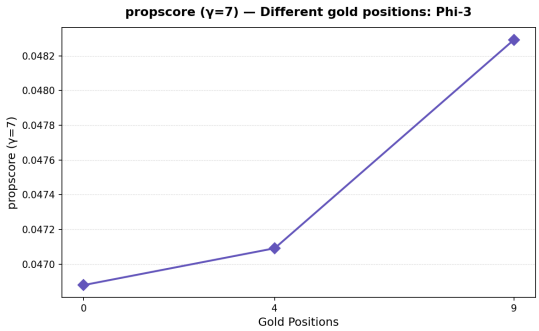
(b) BLEU



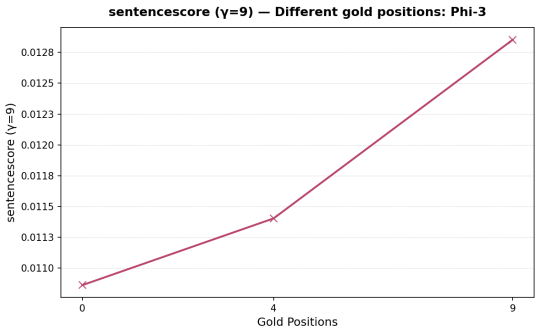
(c) BERTScore



(d) Sentence-BERT (SBERT)

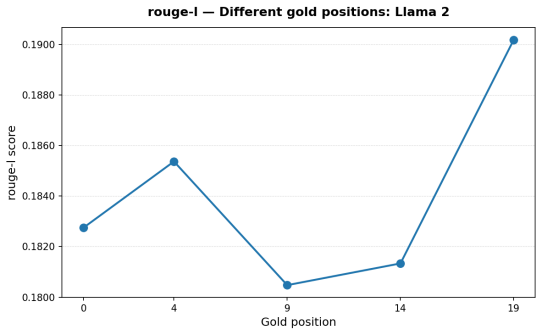


(e) PropScore ($\gamma = 7$)

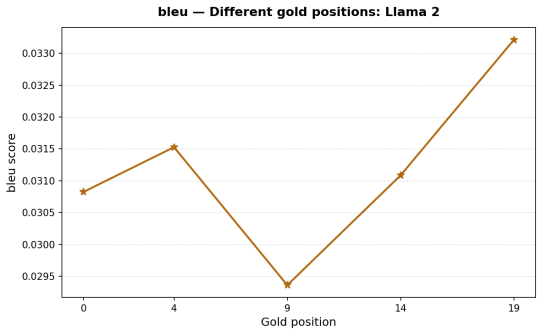


(f) SentenceScore ($\gamma = 9$)

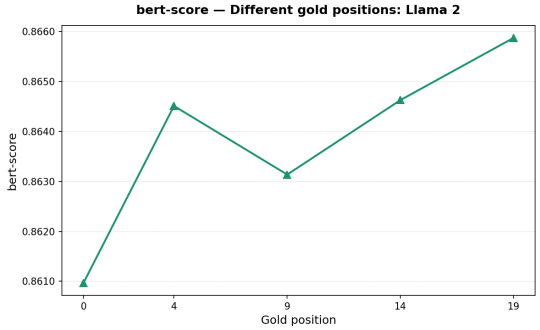
Figure 4.12: Category B (Answer Grounded, 9 distractors): Comprehensive metric comparison as distractor count increases. All the metrics indicate a strong "lost in the middle" effect with high primacy bias.



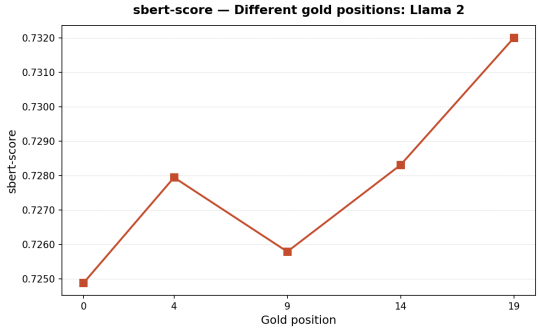
(a) ROUGE-L



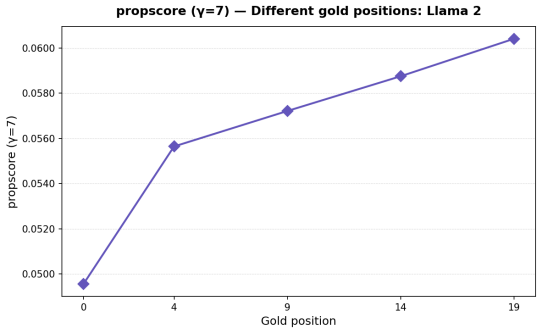
(b) BLEU



(c) BERTScore



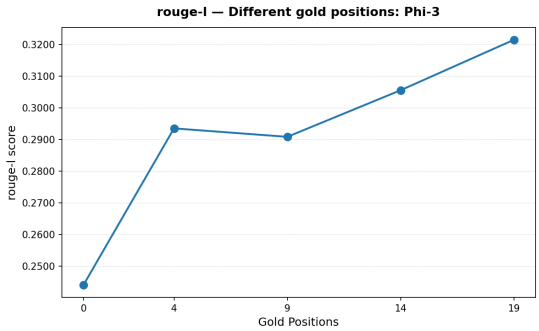
(d) Sentence-BERT (SBERT)



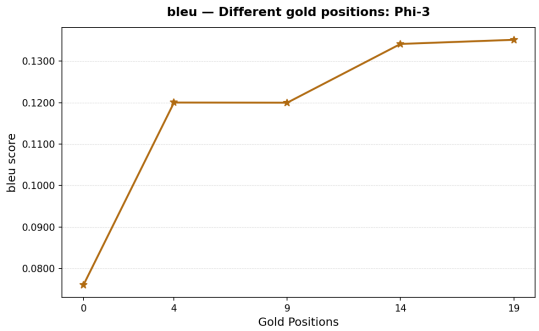
(e) PropScore ($\gamma = 7$)

(f) SentenceScore ($\gamma = 9$)

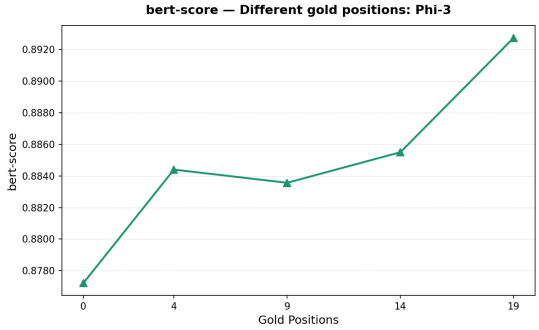
Figure 4.13: Category A (Document Grounded, 19 distractors): Comprehensive metric comparison as distractor count increases. ROUGE-L, BLEU and BERTScore show artificial improvement, Sentence-BERTScore and PropScore drops initially and then increases again, while sentence score accurately capture the context degradation.



(a) ROUGE-L

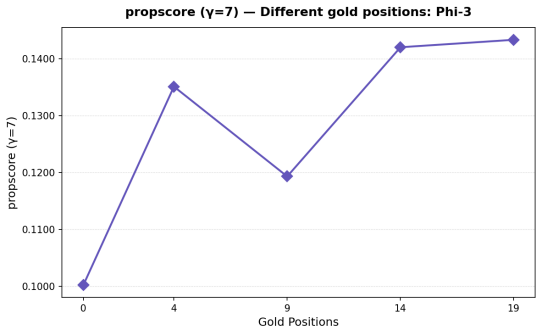


(b) BLEU

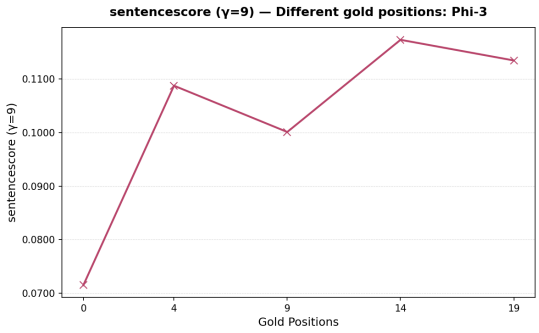


(c) BERTScore

(d) Sentence-BERT (SBERT)

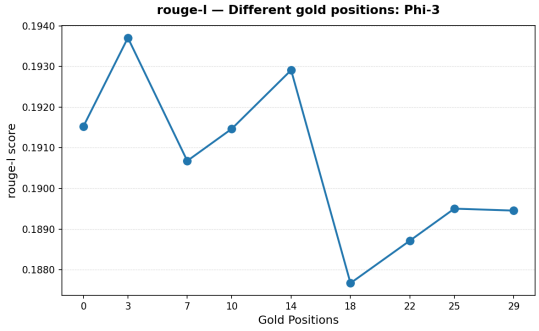


(e) PropScore ($\gamma = 7$)

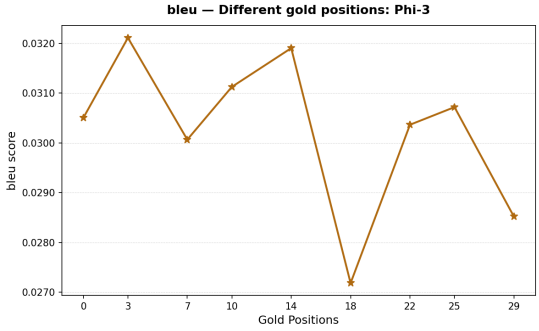


(f) SentenceScore ($\gamma = 9$)

Figure 4.14: Category B (Answer Grounded, 19 distractors): Comprehensive metric comparison as distractor count increases. All the metrics indicate a strong "lost in the middle" effect with high primacy and recency bias.



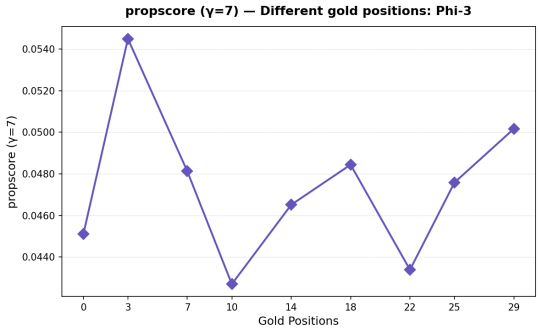
(a) ROUGE-L



(b) BLEU

(c) BERTScore

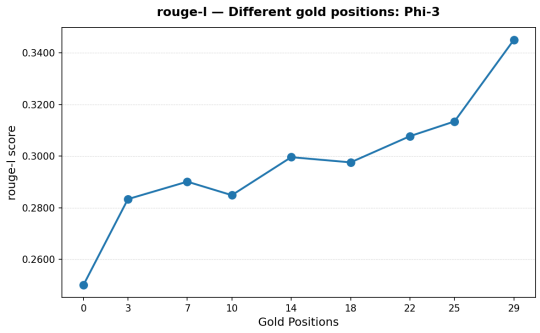
(d) Sentence-BERT (SBERT)



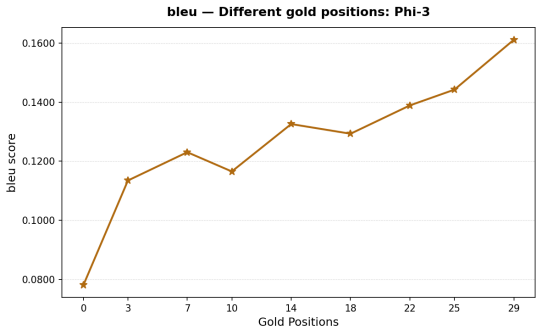
(e) PropScore ($\gamma = 7$)

(f) SentenceScore ($\gamma = 9$)

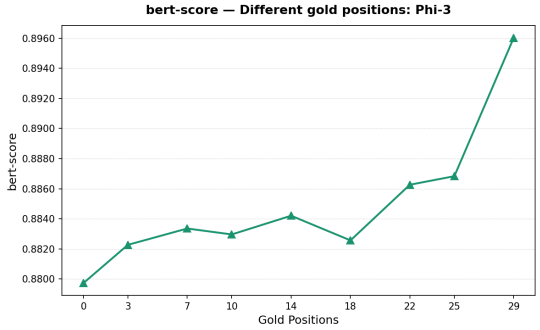
Figure 4.15: Category A (Document Grounded, 29 distractors): Comprehensive metric comparison as distractor count increases. ROUGE-L, BLEU and BERTScore show artificial improvement, Sentence-BERTScore and PropScore drops initially and then increases again, while sentence score accurately capture the context degradation.



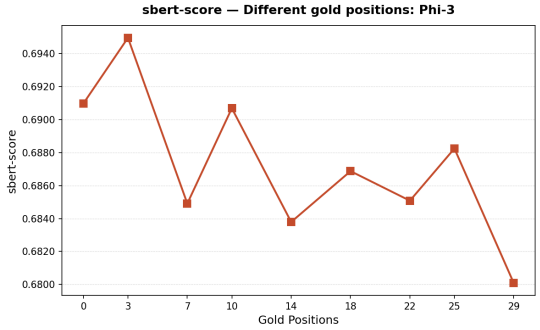
(a) ROUGE-L



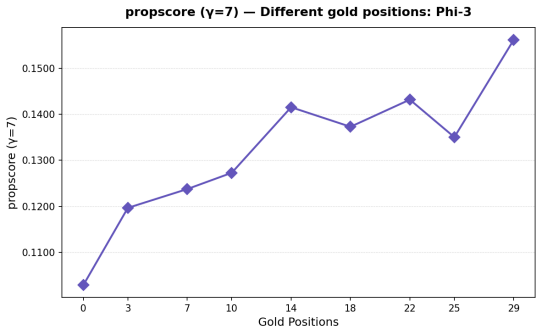
(b) BLEU



(c) BERTScore



(d) Sentence-BERT (SBERT)



(e) PropScore ($\gamma = 7$)

(f) SentenceScore ($\gamma = 9$)

Figure 4.16: Category B (Answer Grounded, 29 distractors): Comprehensive metric comparison as distractor count increases. All the metrics indicate a strong "lost in the middle" effect with high primacy and recency bias.

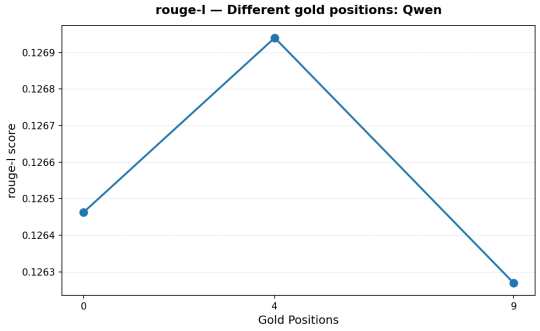
4.2.3 Model: Qwen2.5-7b-instruct

Dist	Gold Position	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1265	0.8418	0.7237	0.0175	0.0660	0.0285
	4	0.1269	0.8421	0.7181	0.0176	0.0689	0.0225
	9	0.1263	0.8414	0.7208	0.0168	0.0724	0.0278
19	0	0.1264	0.8422	0.7165	0.0173	0.0771	0.0262
	4	0.1254	0.8396	0.7172	0.0172	0.0745	0.0213
	9	0.1234	0.8402	0.7187	0.0164	0.0720	0.0246
	14	0.1246	0.8399	0.7197	0.0169	0.0651	0.0211
	19	0.1242	0.8403	0.7216	0.0168	0.0678	0.0251
29	0	0.1211	0.8384	0.7199	0.0164	0.0683	0.0266
	3	0.1221	0.8397	0.7178	0.0171	0.0731	0.0294
	7	0.1220	0.8379	0.7166	0.0165	0.0740	0.0320
	10	0.1226	0.8389	0.7184	0.0171	0.0711	0.0321
	14	0.1226	0.8394	0.7206	0.0168	0.0750	0.0365
	18	0.1246	0.8401	0.7165	0.0168	0.0692	0.0333
	22	0.1240	0.8389	0.7174	0.0183	0.0825	0.0373
	25	0.1238	0.8399	0.7207	0.0170	0.0687	0.0300
	29	0.1220	0.8390	0.7176	0.0164	0.0662	0.0272

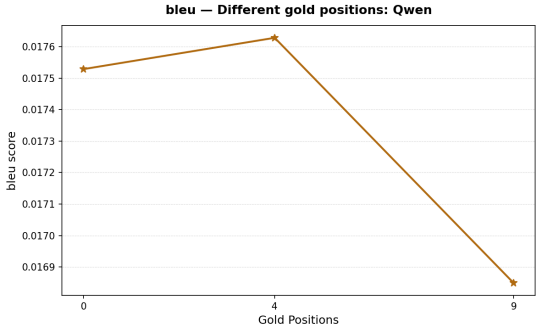
Table 4.11: **Exp2a Results -** Performance metrics for Llama-2 across varying gold document positions within 9-distractor, 19-distractor, and 29-distractor context windows.

Dist	Gold Position	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1544	0.8522	0.7199	0.0373	0.1290	0.0723
	4	0.1528	0.8508	0.7130	0.0375	0.1390	0.0931
	9	0.1477	0.8472	0.7198	0.0357	0.1264	0.0755
19	0	0.1502	0.8494	0.7158	0.0339	0.1153	0.0610
	4	0.1442	0.8462	0.7085	0.0347	0.1170	0.0671
	9	0.1480	0.8479	0.7129	0.0372	0.1248	0.0820
	14	0.1464	0.8471	0.7083	0.0371	0.1237	0.0795
	19	0.1441	0.8463	0.7162	0.0346	0.1266	0.0750
29	0	0.1533	0.8493	0.7171	0.0385	0.1265	0.0747
	3	0.1413	0.8432	0.7063	0.0336	0.1201	0.0750
	7	0.1440	0.8458	0.7082	0.0356	0.1174	0.0808
	10	0.1462	0.8461	0.7088	0.0383	0.1275	0.0838
	14	0.1498	0.8472	0.7124	0.0402	0.1285	0.0972
	18	0.1510	0.8489	0.7152	0.0397	0.1309	0.1006
	22	0.1504	0.8466	0.7109	0.0432	0.1336	0.0905
	25	0.1545	0.8484	0.7163	0.0427	0.1277	0.0834
	29	0.1467	0.8453	0.7180	0.0382	0.1382	0.0874

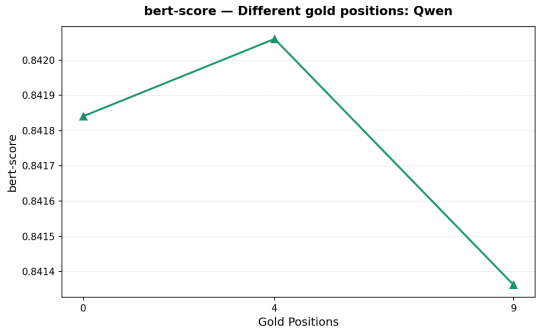
Table 4.12: **Exp2b Results -** Performance metrics across varying gold document positions within 9-distractor, 19-distractor, and 29-distractor context windows.



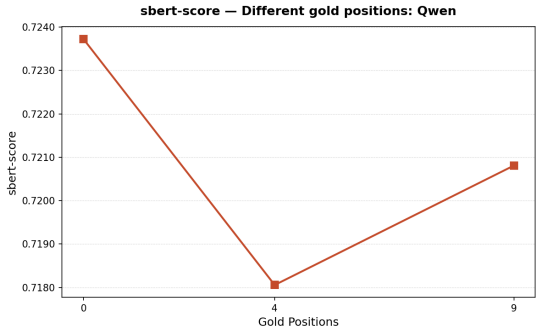
(a) ROUGE-L



(b) BLEU



(c) BERTScore

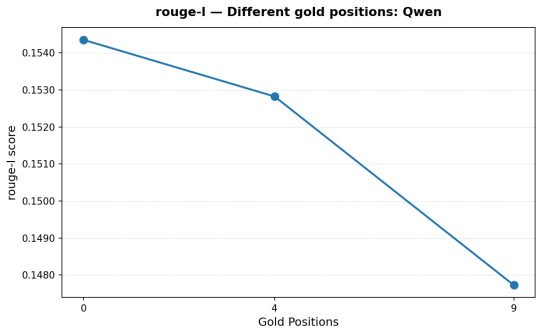


(d) Sentence-BERT (SBERT)

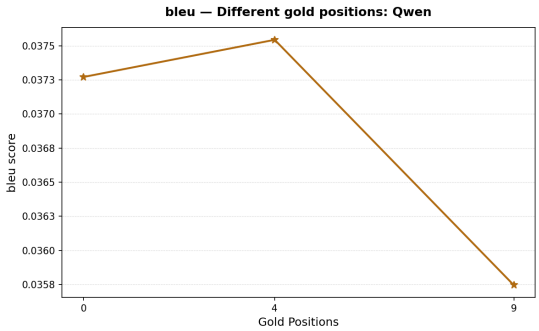
(e) PropScore ($\gamma = 7$)

(f) SentenceScore ($\gamma = 9$)

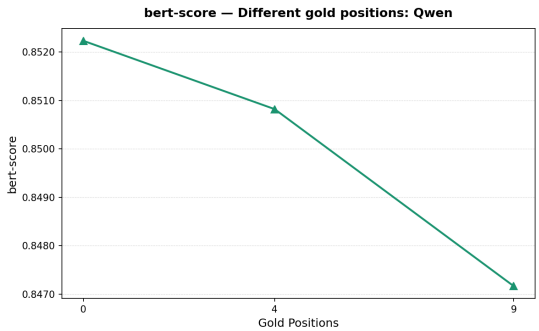
Figure 4.17: Category A (Document Grounded, 9 distractors): Comprehensive metric comparison as gold position increases. ROUGE-L, BLEU show mild "lost in the middle effect with high recency bias; BERTScore and SBERTScore starts with lowest value, then dips followed by an increase, then the metric gradually increases and peaks at last position; PropScore increases linearly, while SentenceScore peaks at the middle.



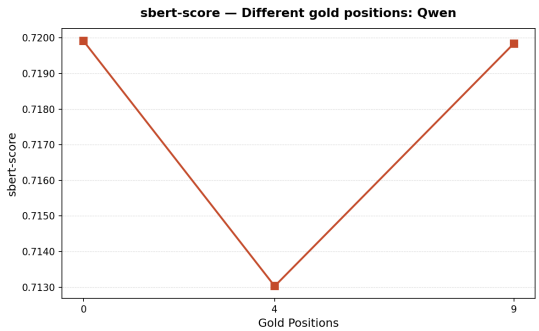
(a) ROUGE-L



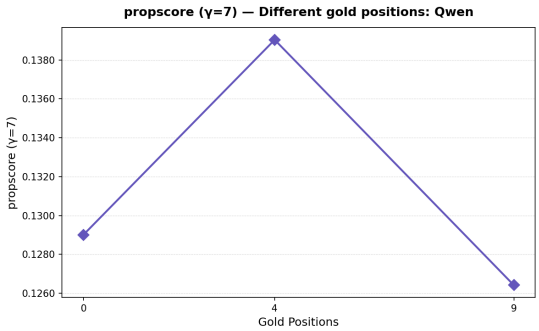
(b) BLEU



(c) BERTScore



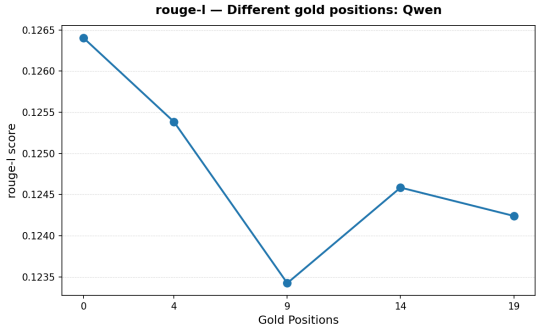
(d) Sentence-BERT (SBERT)



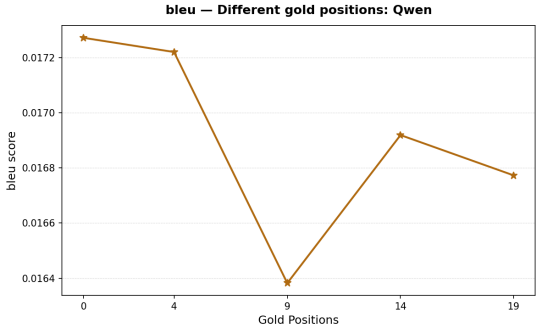
(e) PropScore ($\gamma = 7$)

(f) SentenceScore ($\gamma = 9$)

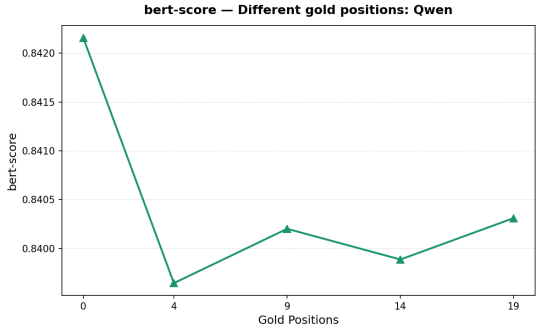
Figure 4.18: Category B (Answer Grounded, 9 distractors): Comprehensive metric comparison as distractor count increases. All the metrics indicate a strong "lost in the middle" effect with high primacy bias.



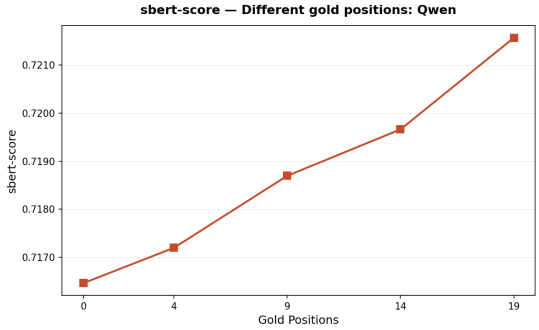
(a) ROUGE-L



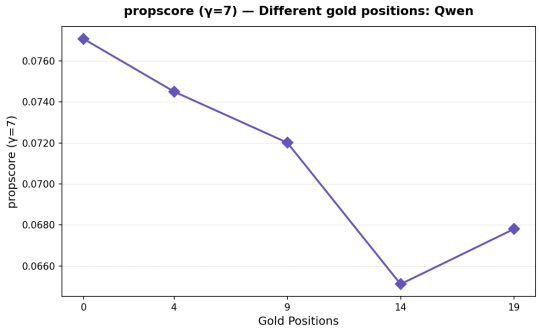
(b) BLEU



(c) BERTScore



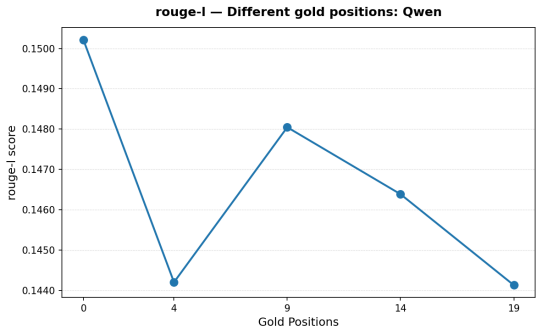
(d) Sentence-BERT (SBERT)



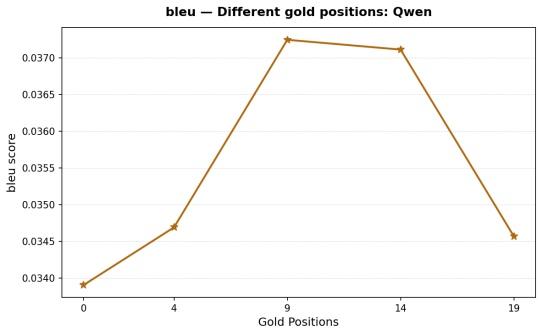
(e) PropScore ($\gamma = 7$)

(f) SentenceScore ($\gamma = 9$)

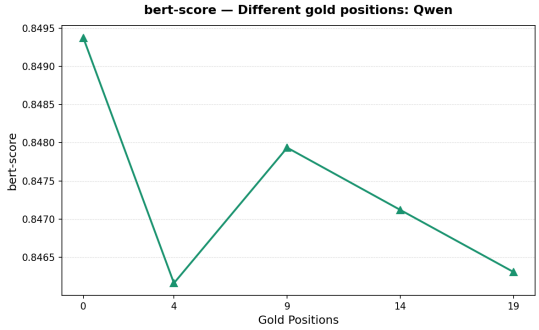
Figure 4.19: Category A (Document Grounded, 19 distractors): Comprehensive metric comparison as distractor count increases. ROUGE-L, BLEU and BERTScore show artificial improvement, Sentence-BERTScore and PropScore drops initially and then increases again, while sentence score accurately capture the context degradation.



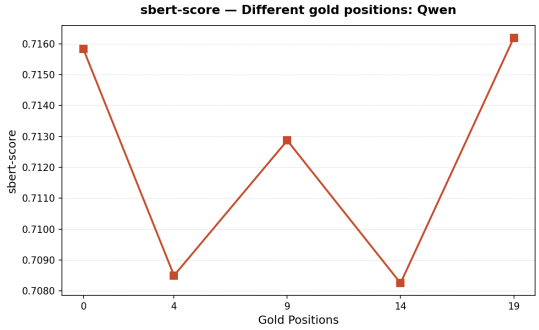
(a) ROUGE-L



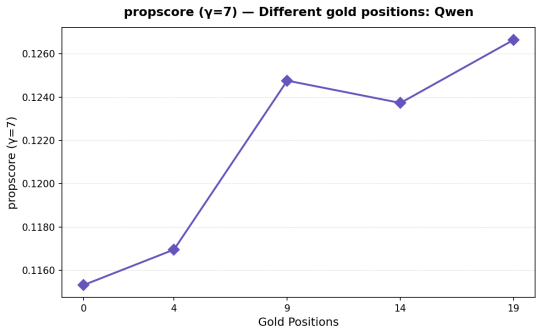
(b) BLEU



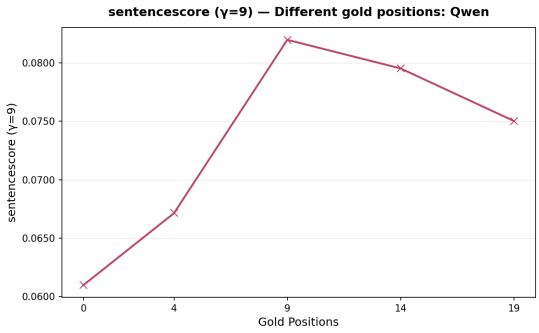
(c) BERTScore



(d) Sentence-BERT (SBERT)

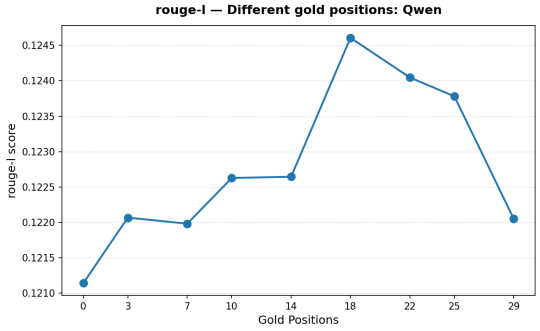


(e) PropScore ($\gamma = 7$)

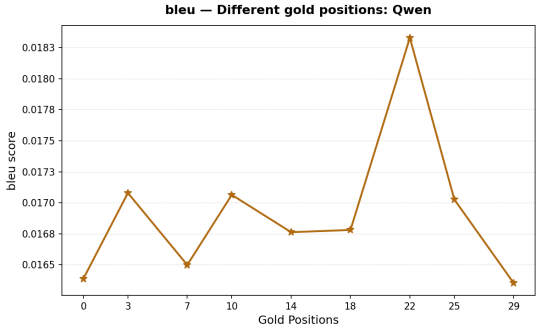


(f) SentenceScore ($\gamma = 9$)

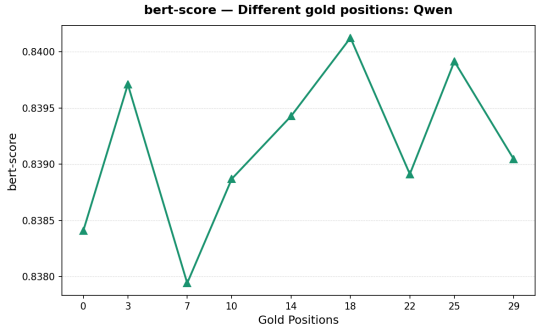
Figure 4.20: Category B (Answer Grounded, 19 distractors): Comprehensive metric comparison as distractor count increases. All the metrics indicate a strong "lost in the middle" effect with high primacy and recency bias.



(a) ROUGE-L

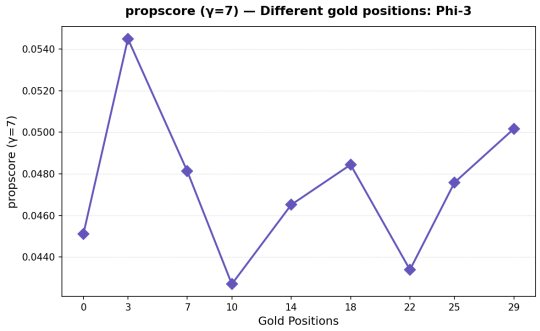


(b) BLEU



(c) BERTScore

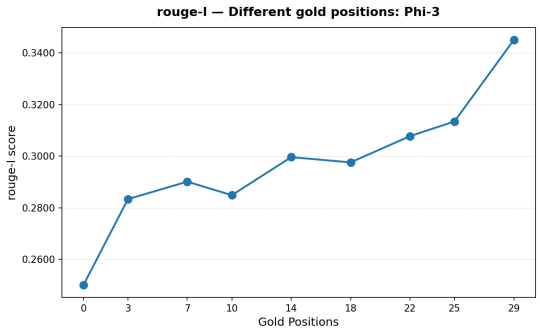
(d) Sentence-BERT (SBERT)



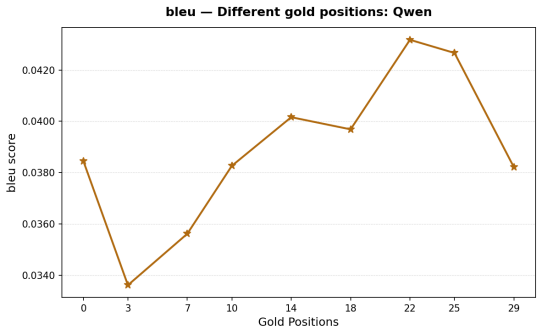
(e) PropScore ($\gamma = 7$)

(f) SentenceScore ($\gamma = 9$)

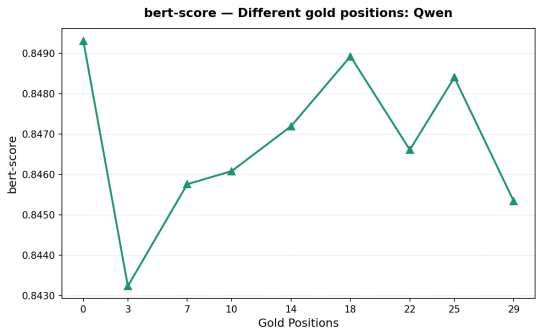
Figure 4.21: Category A (Document Grounded, 29 distractors): Comprehensive metric comparison as distractor count increases. ROUGE-L, BLEU and BERTScore show artificial improvement, Sentence-BERTScore and PropScore drops initially and then increases again, while sentence score accurately capture the context degradation.



(a) ROUGE-L



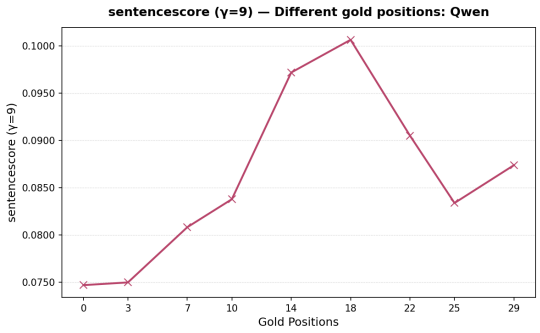
(b) BLEU



(c) BERTScore

(d) Sentence-BERT (SBERT)

(e) PropScore ($\gamma = 7$)



(f) SentenceScore ($\gamma = 9$)

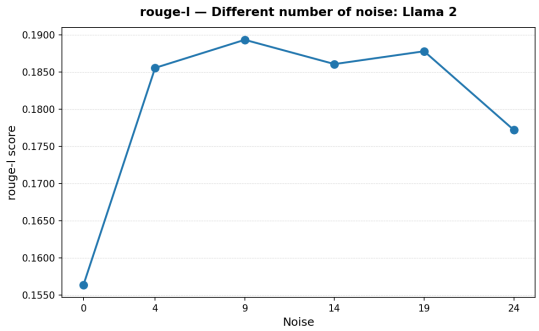
Figure 4.22: Category B (Answer Grounded, 29 distractors): Comprehensive metric comparison as distractor count increases. All the metrics indicate a strong "lost in the middle" effect with high primacy and recency bias.

4.3 exp3a and 3b Results

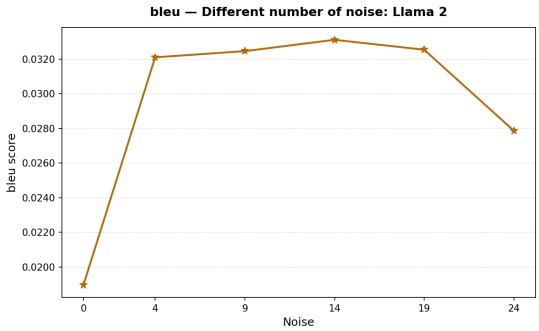
4.3.1 Llama-2-7b-chat-hf

Experiment	Noise	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
exp3a	0	0.1563	0.8494	0.6628	0.0190	0.0430	0.0133
	4	0.1856	0.8611	0.7011	0.0321	0.0545	0.0185
	9	0.1894	0.8638	0.7042	0.0325	0.0589	0.0230
	14	0.1861	0.8635	0.7151	0.0331	0.0653	0.0199
	19	0.1878	0.8672	0.7309	0.0325	0.0575	0.0245
	24	0.1772	0.8637	0.7645	0.0279	0.0211	0.0295
exp3b	0	0.2117	0.8681	0.6461	0.0463	0.0693	0.0474
	4	0.3561	0.8988	0.7439	0.1744	0.1778	0.1536
	9	0.4238	0.9085	0.7755	0.2570	0.2759	0.2317
	14	0.4594	0.9143	0.7974	0.3053	0.3170	0.2828
	19	0.4411	0.9116	0.7942	0.2965	0.3081	0.2835
	24	0.4922	0.9176	0.8117	0.3511	0.3189	0.3171

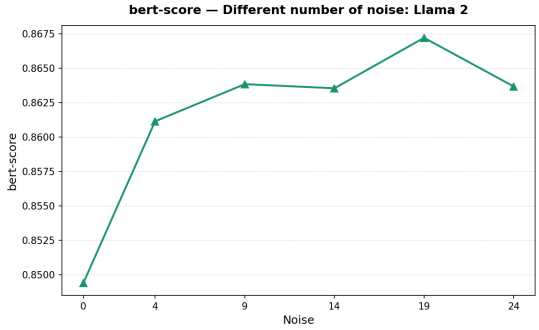
Table 4.13: Comparative performance metrics across varying noise document loads for exp3a and exp3b setups. PropScore values reflect strictness threshold $\gamma = 7$ and SentenceScore uses $\gamma = 9$.



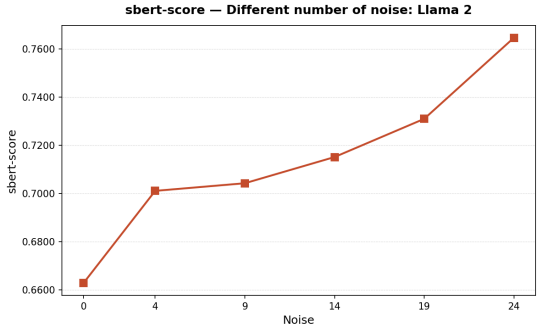
(a) ROUGE-L



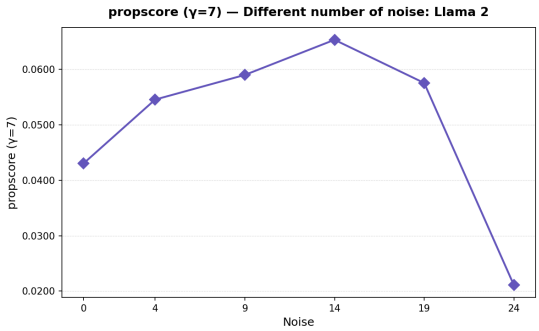
(b) BLEU



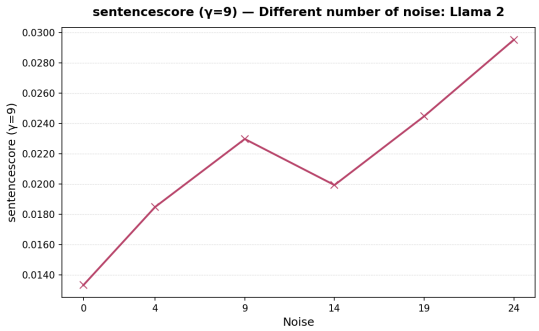
(c) BERTScore



(d) Sentence-BERT (SBERT)

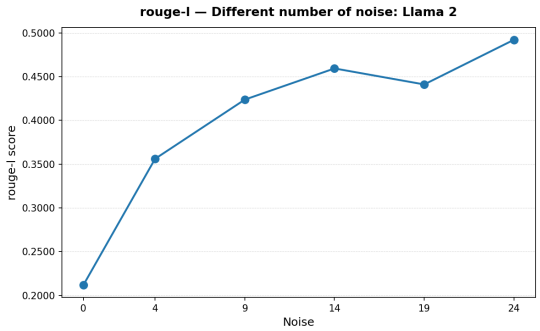


(e) PropScore ($\gamma = 7$)

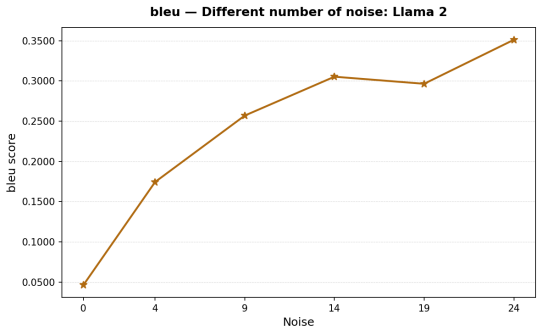


(f) SentenceScore ($\gamma = 9$)

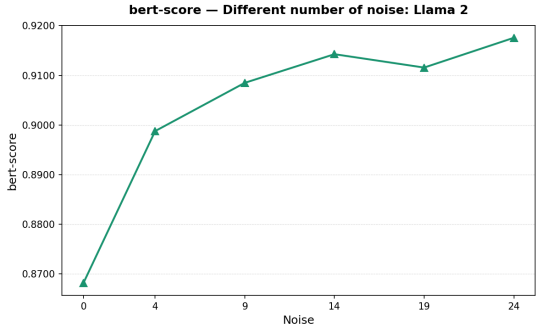
Figure 4.23: Category A (Document Grounded): Comprehensive metric comparison as distractor count increases. All metrics increase as distractor count increases, most peaks at 14 or 19, then flattens (except sentence score).



(a) ROUGE-L

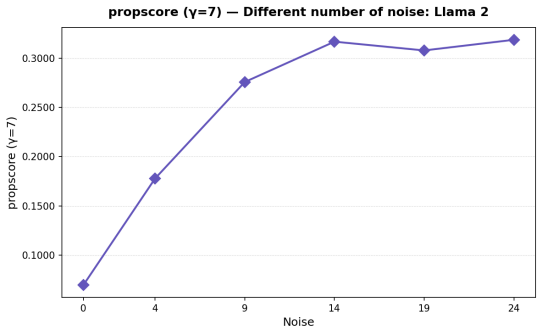


(b) BLEU

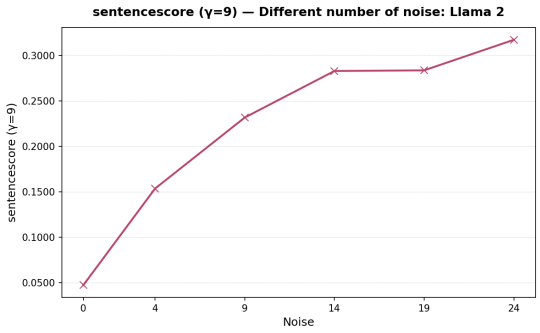


(c) BERTScore

(d) Sentence-BERT (SBERT)



(e) PropScore ($\gamma = 7$)



(f) SentenceScore ($\gamma = 9$)

Figure 4.24: Category B (Answer Grounded): Comprehensive metric comparison as distractor count increases. All metrics increase as distractor count increases, then flattens.

4.3.2 Phi-3-mini

Exp	Noise	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
exp3a	0	0.1342	0.8418	0.6812	0.0178	0.0577	0.0423
	4	0.1919	0.8736	0.6637	0.0256	0.0516	0.0168
	9	0.1916	0.8729	0.6566	0.0276	0.0494	0.0107
	14	0.1931	0.8738	0.6637	0.0267	0.0401	0.0142
	19	0.1938	0.8731	0.6605	0.0286	0.0483	0.0152
	24	0.1937	0.8695	0.6611	0.0300	0.0485	0.0115
	29	0.1904	0.8714	0.6644	0.0277	0.0488	0.0140
exp3b	0	0.1557	0.8488	0.7604	0.0484	0.2506	0.3484
	4	0.3161	0.8964	0.6680	0.1054	0.1221	0.0847
	9	0.3406	0.8936	0.6755	0.1292	0.1357	0.1034
	14	0.3588	0.8994	0.6813	0.1526	0.1546	0.1250
	19	0.3700	0.9009	0.6860	0.1591	0.1580	0.1250
	24	0.3744	0.9010	0.6954	0.1708	0.1706	0.1399
	29	0.3820	0.9024	0.6990	0.1832	0.1775	0.1602

Table 4.14: Comparative performance metrics across varying noise document loads under the exp3a and exp3b frameworks. PropScore tracks strictness threshold $\gamma = 7$ and SentenceScore utilizes $\gamma = 9$.

4.3.3 Qwen2.5-7b-instruct

Exp	Noise	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
exp3a	0	0.1044	0.8294	0.7123	0.0128	0.0798	0.0446
	4	0.1225	0.8395	0.7089	0.0163	0.0691	0.0261
	9	0.1279	0.8418	0.7168	0.0181	0.0786	0.0393
	14	0.1327	0.8450	0.7222	0.0190	0.0790	0.0355
	19	0.1309	0.8438	0.7213	0.0190	0.0806	0.0348
	24	0.1307	0.8438	0.7282	0.0190	0.0834	0.0465
	29	0.1313	0.8447	0.7264	0.0189	0.0828	0.0357
exp3b	0	0.1398	0.8473	0.7639	0.0411	0.2192	0.1566
	4	0.1636	0.8559	0.7491	0.0489	0.1864	0.1350
	9	0.1832	0.8609	0.7528	0.0621	0.2083	0.1462
	14	0.2086	0.8677	0.7690	0.0809	0.2230	0.1650
	19	0.2047	0.8655	0.7597	0.0771	0.2215	0.1532
	24	0.2116	0.8679	0.7660	0.0836	0.2367	0.1766
	29	0.2126	0.8684	0.7641	0.0834	0.2297	0.1662

Table 4.15: Comparative evaluation metrics across expanding context loads (0 to 29 noise documents) under exp3a and exp3b frameworks. Custom constraints utilize PropScore strictness threshold $\gamma = 7$ and SentenceScore strictness $\gamma = 9$.

4.4 exp4a and exp4b Results

4.4.1 Llama-2-7b-chat-hf

Num Noise	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1704	0.8525	0.6866	0.0256	0.0605	0.0194
	4	0.1831	0.8592	0.7020	0.0315	0.0613	0.0190
	9	0.1864	0.8625	0.6968	0.0319	0.0613	0.0226
19	0	0.1825	0.8645	0.7159	0.0265	0.0570	0.0198
	4	0.1856	0.8661	0.7290	0.0299	0.0572	0.0166
	9	0.1852	0.8653	0.7355	0.0329	0.0618	0.0240
	14	0.1852	0.8651	0.7331	0.0328	0.0535	0.0220
	19	0.1916	0.8660	0.7309	0.0329	0.0607	0.0242

Table 4.16: **Exp4a Results -** Performance metrics for Llama-2 across varying gold document positions within 9-noise-document and 19-noise-document context windows.

Noise	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1704	0.8525	0.6866	0.0256	0.0605	0.0194
	4	0.1831	0.8592	0.7020	0.0315	0.0613	0.0190
	9	0.1864	0.8625	0.6968	0.0319	0.0613	0.0226
19	0	0.1825	0.8645	0.7159	0.0265	0.0570	0.0198
	4	0.1856	0.8661	0.7290	0.0299	0.0572	0.0166
	9	0.1852	0.8653	0.7355	0.0329	0.0618	0.0240
	14	0.1852	0.8651	0.7331	0.0328	0.0535	0.0220
	19	0.1916	0.8660	0.7309	0.0329	0.0607	0.0242

Table 4.17: **Exp4b Results -** Performance metrics for Llama-2 across varying gold document positions within 9-noise-document and 19-noise-document context windows.

4.4.2 Phi-3-mini

Noise	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1544	0.8522	0.7199	0.0373	0.1290	0.0723
	4	0.1528	0.8508	0.7130	0.0375	0.1390	0.0931
	9	0.1477	0.8472	0.7198	0.0357	0.1264	0.0755
19	0	0.1502	0.8494	0.7158	0.0339	0.1153	0.0610
	4	0.1442	0.8462	0.7085	0.0347	0.1170	0.0671
	9	0.1480	0.8479	0.7129	0.0372	0.1248	0.0820
	14	0.1464	0.8471	0.7083	0.0371	0.1237	0.0795
	19	0.1441	0.8463	0.7162	0.0346	0.1266	0.0750
29	0	0.1533	0.8493	0.7171	0.0385	0.1265	0.0747
	3	0.1413	0.8432	0.7063	0.0336	0.1201	0.0750
	7	0.1440	0.8458	0.7082	0.0356	0.1174	0.0808
	10	0.1462	0.8461	0.7088	0.0383	0.1275	0.0838
	14	0.1498	0.8472	0.7124	0.0402	0.1285	0.0972
	18	0.1510	0.8489	0.7152	0.0397	0.1309	0.1006
	22	0.1504	0.8466	0.7109	0.0432	0.1336	0.0905
	25	0.1545	0.8484	0.7163	0.0427	0.1277	0.0834
	29	0.1467	0.8453	0.7180	0.0382	0.1382	0.0874

Table 4.18: **Exp4a Results -** Performance metrics across expanding context loads (9, 19, and 29 distractors) as a function of the gold document position for Phi-3.

Noise	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1544	0.8522	0.7199	0.0373	0.1290	0.0723
	4	0.1528	0.8508	0.7130	0.0375	0.1390	0.0931
	9	0.1477	0.8472	0.7198	0.0357	0.1264	0.0755
19	0	0.1502	0.8494	0.7158	0.0339	0.1153	0.0610
	4	0.1442	0.8462	0.7085	0.0347	0.1170	0.0671
	9	0.1480	0.8479	0.7129	0.0372	0.1248	0.0820
	14	0.1464	0.8471	0.7083	0.0371	0.1237	0.0795
	19	0.1441	0.8463	0.7162	0.0346	0.1266	0.0750
29	0	0.1533	0.8493	0.7171	0.0385	0.1265	0.0747
	3	0.1413	0.8432	0.7063	0.0336	0.1201	0.0750
	7	0.1440	0.8458	0.7082	0.0356	0.1174	0.0808
	10	0.1462	0.8461	0.7088	0.0383	0.1275	0.0838
	14	0.1498	0.8472	0.7124	0.0402	0.1285	0.0972
	18	0.1510	0.8489	0.7152	0.0397	0.1309	0.1006
	22	0.1504	0.8466	0.7109	0.0432	0.1336	0.0905
	25	0.1545	0.8484	0.7163	0.0427	0.1277	0.0834
	29	0.1467	0.8453	0.7180	0.0382	0.1382	0.0874

Table 4.19: **Exp4b Results -** Performance summary across context capacities (9, 19, and 29 distractor counts) tracked against varying gold position for Phi-3.

4.4.3 Qwen2.5-7b-instruct

Noise	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1274	0.8422	0.7160	0.0174	0.0742	0.0336
	4	0.1245	0.8410	0.7173	0.0174	0.0857	0.0417
	9	0.1287	0.8415	0.7171	0.0184	0.0665	0.0290
19	0	0.1243	0.8391	0.7092	0.0178	0.0750	0.0318
	4	0.1249	0.8411	0.7144	0.0173	0.0794	0.0415
	9	0.1211	0.8404	0.7215	0.0163	0.0880	0.0366
	14	0.1216	0.8405	0.7219	0.0164	0.0822	0.0342
	19	0.1295	0.8435	0.7226	0.0181	0.0757	0.0350
29	0	0.1254	0.8406	0.7202	0.0178	0.0877	0.0300
	3	0.1216	0.8400	0.7208	0.0169	0.0730	0.0298
	7	0.1214	0.8389	0.7166	0.0163	0.0753	0.0278
	10	0.1206	0.8387	0.7224	0.0168	0.0686	0.0296
	14	0.1200	0.8397	0.7237	0.0155	0.0841	0.0355
	18	0.1223	0.8398	0.7240	0.0169	0.0855	0.0419
	22	0.1224	0.8414	0.7219	0.0167	0.0774	0.0314
	25	0.1225	0.8411	0.7233	0.0161	0.0726	0.0309
	29	0.1306	0.8435	0.7210	0.0193	0.0746	0.0345

Table 4.20: **Exp4a Results -** Performance metrics across context capacities (9, 19, and 29 distractor counts) tracked against different gold positions for Qwen2.5.

Noise	Gold Pos	ROUGE-L	BERTScore	SBERT	BLEU	PropScore7	SentenceScore9
9	0	0.1544	0.8522	0.7199	0.0373	0.1290	0.0723
	4	0.1528	0.8508	0.7130	0.0375	0.1390	0.0931
	9	0.1477	0.8472	0.7198	0.0357	0.1264	0.0755
19	0	0.1502	0.8494	0.7158	0.0339	0.1153	0.0610
	4	0.1442	0.8462	0.7085	0.0347	0.1170	0.0671
	9	0.1480	0.8479	0.7129	0.0372	0.1248	0.0820
	14	0.1464	0.8471	0.7083	0.0371	0.1237	0.0795
	19	0.1441	0.8463	0.7162	0.0346	0.1266	0.0750
29	0	0.1533	0.8493	0.7171	0.0385	0.1265	0.0747
	3	0.1413	0.8432	0.7063	0.0336	0.1201	0.0750
	7	0.1440	0.8458	0.7082	0.0356	0.1174	0.0808
	10	0.1462	0.8461	0.7088	0.0383	0.1275	0.0838
	14	0.1498	0.8472	0.7124	0.0402	0.1285	0.0972
	18	0.1510	0.8489	0.7152	0.0397	0.1309	0.1006
	22	0.1504	0.8466	0.7109	0.0432	0.1336	0.0905
	25	0.1545	0.8484	0.7163	0.0427	0.1277	0.0834
	29	0.1467	0.8453	0.7180	0.0382	0.1382	0.0874

Table 4.21: **Exp4b Results -** Performance metrics across context capacities (9, 19, and 29 distractor counts) tracked against varying gold position for Qwen2.5

4.5 exp5a and exp5b Results

4.5.1 Llama-2-7b-chat-hf

Table 4.22: Evaluation metrics across distractor–noise compositions for 9-filler and 19-filler settings. PropScore is reported at γ=7 and SentenceScore at γ=9. Llama 2

Fillers	R/S	ROUGE-L	BERT	SBERT	BLEU	PropScore7	SentScore9
9 Fillers	0 / 9	0.1785	0.8579	0.7037	0.0302	0.0557	0.0218
	2 / 7	0.1807	0.8592	0.7067	0.0314	0.0688	0.0192
	5 / 4	0.1848	0.8616	0.7146	0.0322	0.0534	0.0230
	7 / 2	0.1899	0.8643	0.7169	0.0336	0.0557	0.0168
	9 / 0	0.1888	0.8641	0.7076	0.0328	0.0624	0.0238
19 Fillers	0 / 19	0.1842	0.8654	0.7287	0.0321	0.0593	0.0204
	2 / 17	0.1862	0.8654	0.7367	0.0340	0.0591	0.0264
	4 / 15	0.1836	0.8638	0.7356	0.0330	0.0582	0.0196
	7 / 12	0.1863	0.8644	0.7395	0.0347	0.0592	0.0221
	10 / 9	0.1872	0.8653	0.7412	0.0340	0.0596	0.0229
	14 / 5	0.1884	0.8668	0.7397	0.0335	0.0603	0.0184
	17 / 2	0.1892	0.8637	0.7267	0.0319	0.0573	0.0234
	19 / 0	0.1876	0.8669	0.7356	0.0319	0.0584	0.0258

Table 4.23: Metrics across distractor–noise compositions for 9-filler and 19-filler settings. Llama 2

Fillers	R/S	ROUGE-L	BERT-Score	SBERT	BLEU	PropScore7	SentScore9
9 Fillers	0 / 9	0.3853	0.9016	0.7449	0.2203	0.2151	0.1994
	2 / 7	0.4172	0.9068	0.7599	0.2476	0.2365	0.2178
	5 / 4	0.4577	0.9136	0.7776	0.2956	0.2896	0.2739
	7 / 2	0.4734	0.9165	0.7894	0.3116	0.3112	0.2897
	9 / 0	0.4286	0.9094	0.7785	0.2624	0.2677	0.2345
19 Fillers	0 / 19	0.4105	0.9055	0.7720	0.2598	0.2575	0.2331
	2 / 17	0.4301	0.9081	0.7764	0.2819	0.2860	0.2638
	4 / 15	0.4184	0.9065	0.7762	0.2709	0.2751	0.2484
	7 / 12	0.4228	0.9070	0.7796	0.2688	0.2746	0.2533
	9 / 10	0.4332	0.9091	0.7890	0.2863	0.2852	0.2673
	11 / 8	0.4493	0.9116	0.7936	0.3019	0.3051	0.2854
	14 / 5	0.4480	0.9114	0.7905	0.3039	0.3091	0.2970
	17 / 2	0.4497	0.9124	0.7964	0.3070	0.3088	0.2866
	19 / 0	0.4507	0.9130	0.7971	0.3068	0.3185	0.2983

4.5.2 Phi-3-mini

Table 4.24: Metrics across distractor–noise compositions for 9-filler, 19-filler, and 29-filler settings. Phi-3

Fillers	R/S	ROUGE-L	BERT	SBERT	BLEU	PropScore7	SentScore9
9 Fillers	0 / 9	0.1897	0.8695	0.6915	0.0296	0.0500	0.0120
	2 / 7	0.1942	0.8699	0.6847	0.0296	0.0454	0.0143
	5 / 4	0.1927	0.8721	0.6694	0.0279	0.0464	0.0199
	7 / 2	0.1915	0.8725	0.6700	0.0278	0.0483	0.0115
	9 / 0	0.1909	0.8724	0.6543	0.0260	0.0469	0.0128
19 Fillers	0 / 19	0.1864	0.8665	0.6889	0.0285	0.0454	0.0135
	2 / 17	0.1956	0.8699	0.6841	0.0286	0.0483	0.0153
	4 / 15	0.1912	0.8713	0.6851	0.0284	0.0402	0.0166
	7 / 12	0.1936	0.8723	0.6813	0.0293	0.0456	0.0136
	9 / 10	0.1948	0.8705	0.6838	0.0291	0.0485	0.0161
	11 / 8	0.1958	0.8709	0.6764	0.0312	0.0476	0.0137
	14 / 5	0.1924	0.8711	0.6795	0.0272	0.0488	0.0149
	17 / 2	0.1957	0.8693	0.6699	0.0310	0.0513	0.0129
	19 / 0	0.1967	0.8734	0.6640	0.0296	0.0471	0.0147
29 Fillers	0 / 29	0.1916	0.8703	0.6875	0.0309	0.0479	0.0118
	2 / 27	0.1976	0.8724	0.6941	0.0308	0.0487	0.0112
	4 / 25	0.1932	0.8708	0.6823	0.0290	0.0461	0.0095
	7 / 22	0.1947	0.8700	0.6864	0.0308	0.0537	0.0129
	9 / 20	0.1949	0.8719	0.6929	0.0298	0.0537	0.0120
	12 / 17	0.1988	0.8723	0.6877	0.0297	0.0452	0.0138
	14 / 15	0.1932	0.8705	0.6804	0.0295	0.0553	0.0148
	17 / 12	0.1972	0.8715	0.6873	0.0307	0.0505	0.0122
	19 / 10	0.1958	0.8695	0.6866	0.0303	0.0509	0.0101
	22 / 7	0.1955	0.8710	0.6745	0.0299	0.0468	0.0103
	24 / 5	0.1995	0.8702	0.6820	0.0320	0.0496	0.0160
	27 / 2	0.1907	0.8690	0.6681	0.0281	0.0456	0.0129
	29 / 0	0.1942	0.8715	0.6617	0.0303	0.0469	0.0111

4.5.3 Qwen2.5-7b-instruct

Table 4.25: Metrics across distractor–noise compositions for 9-filler, 19-filler, and 29-filler settings. Phi-3

Fillers	R/S	ROUGE-L	BERT	SBERT	BLEU	PropScore7	SentScore9
9 Fillers	0 / 9	0.3138	0.8938	0.6874	0.1181	0.1248	0.0873
	2 / 7	0.3328	0.8946	0.6932	0.1340	0.1339	0.0989
	5 / 4	0.3303	0.8936	0.6788	0.1300	0.1297	0.1025
	7 / 2	0.3468	0.8942	0.6835	0.1469	0.1483	0.1146
	9 / 0	0.3403	0.8932	0.6789	0.1308	0.1414	0.1033
19 Fillers	0 / 19	0.3318	0.8946	0.6994	0.1440	0.1478	0.1092
	2 / 17	0.3330	0.8938	0.6947	0.1470	0.1522	0.1202
	4 / 15	0.3485	0.8962	0.6984	0.1555	0.1591	0.1273
	7 / 12	0.3507	0.8968	0.7006	0.1597	0.1608	0.1254
	9 / 10	0.3633	0.9003	0.7008	0.1761	0.1685	0.1452
	11 / 8	0.3718	0.8986	0.6997	0.1834	0.1789	0.1471
	14 / 5	0.3739	0.9016	0.7018	0.1804	0.1830	0.1569
	17 / 2	0.3837	0.9008	0.6963	0.1899	0.1893	0.1657
	19 / 0	0.3646	0.8999	0.6843	0.1628	0.1645	0.1322
29 Fillers	0 / 29	0.3374	0.8949	0.6985	0.1512	0.1527	0.1248
	2 / 27	0.3498	0.8952	0.6983	0.1666	0.1632	0.1353
	4 / 25	0.3639	0.8979	0.7092	0.1838	0.1793	0.1496
	7 / 22	0.3604	0.8972	0.7000	0.1746	0.1741	0.1526
	9 / 20	0.3700	0.9001	0.7076	0.1876	0.1814	0.1592
	12 / 17	0.3718	0.9004	0.7010	0.1893	0.1813	0.1624
	14 / 15	0.3712	0.9003	0.7125	0.1897	0.1846	0.1596
	17 / 12	0.3826	0.8995	0.7103	0.1999	0.1867	0.1635
	19 / 10	0.3918	0.9038	0.7124	0.2086	0.1967	0.1739
	22 / 7	0.3829	0.9020	0.7054	0.1955	0.1897	0.1663
	24 / 5	0.3956	0.9060	0.7105	0.2101	0.1962	0.1788
	27 / 2	0.3906	0.9015	0.6981	0.2022	0.1911	0.1781
	29 / 0	0.3857	0.9047	0.7025	0.1866	0.1885	0.1526

4.6 Ablation Study for PropScore and SentenceScore

We sampled 200 queries, reference answers, and generated answers from our 1k ELI5 queries once the models had generated output. Then we read the two answers and scored them on a scale of 1 to based on how similar two answers are. We discuss the findings of our ablation study here.

Table 4.26: Metrics across distractor–noise compositions for 9-filler, 19-filler, and 29-filler settings. Qwen2.5

Fillers	R/S	ROUGE-L	BERT	SBERT	BLEU	PropScore7	SentScore9
9 Fillers	0 / 9	0.1266	0.8425	0.7168	0.0174	0.0678	0.0342
	2 / 7	0.1226	0.8394	0.7155	0.0162	0.0711	0.0310
	5 / 4	0.1243	0.8408	0.7173	0.0165	0.0763	0.0269
	7 / 2	0.1261	0.8408	0.7205	0.0173	0.0732	0.0299
	9 / 0	0.1292	0.8425	0.7155	0.0180	0.0817	0.0359
19 Fillers	0 / 19	0.1245	0.8402	0.7173	0.0170	0.0677	0.0203
	2 / 17	0.1202	0.8374	0.7213	0.0157	0.0641	0.0258
	4 / 15	0.1197	0.8389	0.7252	0.0157	0.0624	0.0219
	7 / 12	0.1218	0.8388	0.7199	0.0168	0.0682	0.0284
	9 / 10	0.1200	0.8386	0.7182	0.0159	0.0750	0.0313
	11 / 8	0.1237	0.8402	0.7266	0.0172	0.0702	0.0323
	14 / 5	0.1268	0.8411	0.7237	0.0173	0.0788	0.0374
	17 / 2	0.1253	0.8412	0.7217	0.0167	0.0745	0.0391
	19 / 0	0.1280	0.8417	0.7266	0.0184	0.0835	0.0331
29 Fillers	0 / 29	0.1218	0.8387	0.7179	0.0160	0.0718	0.0273
	2 / 27	0.1183	0.8366	0.7205	0.0163	0.0733	0.0229
	4 / 25	0.1184	0.8362	0.7226	0.0170	0.0654	0.0246
	7 / 22	0.1184	0.8365	0.7258	0.0169	0.0776	0.0314
	9 / 20	0.1195	0.8382	0.7249	0.0161	0.0746	0.0278
	12 / 17	0.1187	0.8377	0.7221	0.0162	0.0744	0.0285
	14 / 15	0.1193	0.8376	0.7245	0.0161	0.0750	0.0248
	17 / 12	0.1176	0.8377	0.7287	0.0158	0.0733	0.0278
	19 / 10	0.1175	0.8380	0.7276	0.0166	0.0799	0.0259
	22 / 7	0.1212	0.8383	0.7239	0.0164	0.0739	0.0297
	24 / 5	0.1183	0.8389	0.7252	0.0153	0.0796	0.0329
	27 / 2	0.1257	0.8420	0.7244	0.0181	0.0827	0.0326
	29 / 0	0.1286	0.8431	0.7226	0.0184	0.0867	0.0336

4.6.1 Kendall τ and Pearson Correlation ρ

In the table 4.28, we see that the sentence score improved the Kendall τ by a margin of 0.1, while PropScore fell way behind. However, since the sentence-level version of PropScore outperforms all the baselines in τ, it suggests that PropScore's poor performance is caused by its reliance on LLM-extracted propositions. So, we believe that PropScore would have performed even better than SentenceScore if the proposition extraction were of better quality.

Now, both metrics performed significantly badly in ρ. Pearson Correlation measures the linear relationship between the human-written answer and the generated answer. However, human-written answers are highly non-linear in nature. Our

Table 4.27: Metrics across distractor–noise compositions for 9-filler, 19-filler, and 29-filler settings. Qwen2.5

Fillers	R/S	ROUGE-L	BERT	SBERT	BLEU	PropScore7	SentScore9
9 Fillers	0 / 9	0.1465	0.8465	0.7162	0.0345	0.1161	0.0805
	2 / 7	0.1519	0.8490	0.7208	0.0408	0.1404	0.0940
	5 / 4	0.1558	0.8512	0.7289	0.0432	0.1536	0.1134
	7 / 2	0.1634	0.8539	0.7331	0.0469	0.1762	0.1207
	9 / 0	0.1814	0.8604	0.7543	0.0615	0.2044	0.1468
19 Fillers	0 / 19	0.1468	0.8467	0.7145	0.0359	0.1339	0.0855
	2 / 17	0.1462	0.8480	0.7186	0.0358	0.1409	0.0925
	4 / 15	0.1505	0.8492	0.7232	0.0412	0.1516	0.0998
	7 / 12	0.1549	0.8494	0.7243	0.0453	0.1598	0.1097
	9 / 10	0.1609	0.8518	0.7276	0.0482	0.1603	0.1134
	11 / 8	0.1647	0.8532	0.7340	0.0500	0.1670	0.1200
	14 / 5	0.1675	0.8543	0.7352	0.0514	0.1718	0.1164
	17 / 2	0.1919	0.8609	0.7466	0.0694	0.1930	0.1355
	19 / 0	0.2014	0.8651	0.7617	0.0753	0.2231	0.1544
29 Fillers	0 / 29	0.1449	0.8459	0.7197	0.0366	0.1352	0.0918
	2 / 27	0.1504	0.8468	0.7237	0.0410	0.1434	0.0954
	4 / 25	0.1529	0.8476	0.7215	0.0444	0.1516	0.1038
	7 / 22	0.1542	0.8476	0.7252	0.0463	0.1519	0.1148
	9 / 20	0.1560	0.8499	0.7273	0.0453	0.1610	0.1587
	12 / 17	0.1616	0.8510	0.7296	0.0516	0.1629	0.1039
	14 / 15	0.1614	0.8518	0.7304	0.0511	0.1545	0.1079
	17 / 12	0.1644	0.8520	0.7299	0.0520	0.1691	0.1148
	19 / 10	0.1617	0.8517	0.7306	0.0490	0.1664	0.1156
	22 / 7	0.1746	0.8553	0.7334	0.0563	0.1763	0.1211
	24 / 5	0.1759	0.8569	0.7351	0.0561	0.1731	0.1204
	27 / 2	0.1884	0.8607	0.7467	0.0660	0.1945	0.1312
	29 / 0	0.2138	0.8685	0.7654	0.0863	0.2301	0.1720

metrics, by design, can capture these non-linearities. However, being a non-linear metric causes it to lose Pearson Correlation score.

4.6.2 Statistical Significance Test

In Table 4.29, we also show the results of the significance test for Kendall τ, showing that SentenceScore's performance is indeed significant. Also, PropScore's underperformance is significant, too, in this case.
In the test, 1000 samples were generated via bootstrapping from our 200 human-annotated dataset.

Table 4.28: Correlation coefficients vs. Human Annotation for selected configurations and baselines.

Metric	Pearson	Kendall Tau
Sentence Gamma Variants		
sentence_gamma_8	0.2990	0.4462
sentence_gamma_9	0.2982	**0.4465**
sentence_gamma_10	0.2978	0.4455
Baseline Metrics		
sentence-bert	**0.4909**	0.3472
bert-score	0.4616	0.3389
bleu-4	0.4101	0.3198
rouge-l	0.4077	0.3146
Atomic Gamma Variants		
prop_gamma_6	0.2346	0.2624
prop_gamma_7	0.2328	0.2644
prop_gamma_8	0.2317	0.2637

Table 4.29: Significance Test Results for τ

Proposed Metric	Prop Tau	Baseline	Base Tau	p-value	Significant?
sentence_gamma_9	0.4465	bert-score	0.3389	0.038	Yes ($p < 0.05$)
sentence_gamma_9	0.4465	sentence-bert	0.3472	0.040	Yes ($p < 0.05$)
sentence_gamma_9	0.4465	blue-4	0.3198	0.019	Yes ($p < 0.05$)
sentence_gamma_9	0.4465	rouge-l	0.3146	0.015	Yes ($p < 0.05$)
prop_gamma_7	0.2644	bert-score	0.3389	0.863	No
prop_gamma_7	0.2644	sentence-bert	0.3472	0.919	No
prop_gamma_7	0.2644	blue-4	0.3146	0.822	No
prop_gamma_7	0.2644	rouge-l	0.3146	0.784	No

4.7 Discussion

4.7.1 Exp1: Varying Distractor, Gold at Fixed Position

What we observe. In all three models, adding more distractor documents in **exp1a** causes ROUGE-L, BLEU, and BERTScore to increase as context load increases. It suggests an improvement in generation quality, while the opposite should happen.
PropScore and SentenceScore show us a completely different picture for Phi-3-mini and Qwen2.5: PropScore either drops or remains constant as distractor count increases in the Category A setting, suggesting a drop in generation quality as distractor count increases. It fits with our intuition of what should happen, as more distractors mean more confusion. In Category B, a sharp jump in scores is observed for all models when even a small number of distractors are introduced — for instance, Llama-2's ROUGE-L rises from 0.2122 (0 distractors) to 0.3907 (9 distractors), while PropScore and SentenceScore values drop sharply.

Why it happens. When more distractors are added, the model's generated output begins adding vocabulary and context from those distractor documents. Now, distractors are topically related to the query; this increases n-gram and embedding-based similarity against the reference answer, even though the model may be generating misleading or irrelevant information. This is a serious limitation of lexical and contextual embedding metrics.

In Category B, the rise upon adding distractors is also worth examining. When there is no distractor, the model only has the reference answer and doesn't have any context around it. Instruction-tuned models in this kind of setting often paraphrase the answer rather than reproduce the answer closely, which causes low overlap. Adding even a few distractors appears to the model's generation to be closer to the provided answer.

4.7.2 Exp2: Varying Gold Position among Distractors

What we observe. This experiment explicitly examines the "lost in the middle" phenomenon. In Category B, all three models and all evaluation metrics show a **U-shaped performance curve**. We see that scores peak when the gold document is placed at position 0 or at the last position, and drop significantly when it is placed in the middle of the context. For Llama-2 with 9 distractors in Category B, ROUGE-L is 0.4030 at position 0, drops to 0.3466 at the middle position, and

increases to 0.3877 at the last position. This is a reproduction of the paper [1]'s claim, being present even in the long-form QA setup. This pattern is observed in 19 and 29 distractor cases as well and is replicated in Phi-3-mini and Qwen2.5 for Category B. This suggests that "lost in the middle" persists even in Long-Form QA to a huge extent.

For the Category A setting, the positional effect is weaker and less consistent across metrics. Traditional metrics show negligible variation across gold positions, and even PropScore and SentenceScore exhibit only mild trends. The U-shaped curve, while noticeable to some degree, is considerably straightened compared to Category B.

Why it happens. Decoder-only models such as Llama-2 and Qwen2.5 process context causally, i.e., each token attends only to preceding tokens, and information early in the context is repeatedly attended to during the generation of subsequent tokens. Tokens situated in the middle of a long context get a lesser amount of attention. The recency bias arises because the most recently processed tokens remain better represented in the model's internal state at generation time. The primacy bias likely happens as early context shapes the model's initial interpretation of the entire input, and this persists during the generation.

The difference between Category A and Category B results is also something to look at. When the answer itself is used as a gold document (category B), the position directly influences whether the model will reproduce it or not. The gold document contains information relevant to generating an answer (Category A). So, the model can generate good-enough content from background knowledge or distractor documents, bypassing the positional retrieval failure. This suggests that the "lost in the middle" phenomenon affects retrieval of information more, not the generation.

Qwen2.5 shows quite flat positional curves compared to Llama-2 and Phi-3-mini, especially in Category A. This is likely caused by Qwen's large pretraining and more developed positional encoding and attention mechanism. Besides, Qwen is developed specifically to handle long context prompts, which can be an attributing factor as well.

4.7.3 Exp3: Varying Noise Count with Gold at Fixed Position

What we observe. Replacing distractors with random noise documents shows a very different picture. In Category A, traditional metrics still increase with noise count, same as seen in exp1a. But for Phi-3-mini and Qwen2.5 in Category A, PropScore and SentenceScore are more stable than in the corresponding distractor experiment. In Category B, the effect is even more persistent. For Llama-2, ROUGE-L rises from 0.2117 at zero noise documents to 0.4922 at 24 noise documents, a bigger jump than observed with distractors (reaching 0.4280 at 24 distractors in exp1b). PropScore and SentenceScore in Category B show a monotonic increase as noise is added across all three models.

An anomaly appears in the Category A results for Phi-3-mini: at zero noise documents, PropScore (0.0577) and SentenceScore (0.0423) are higher than at 4 noise documents (0.0516 and 0.0168, respectively), before stabilizing at lower values.

Why it happens. The consistent quality improvement under noise injection is the "power of noise" effect [2] reproduced in the Long-Form QA setting. Noise documents are topically unrelated to the query, unlike distractors. They do not lead to misleading or contradictory claims about the topic. So, the gold document stands out more distinctly against a background of unrelated content than against a cluster of topically similar but potentially adverse distractors. The model's attention is better focused toward the relevant information.

The initial drop observed for Phi-3-mini in Category A, where a small number of noise documents actually lowers quality before stabilizing, may reflect that adding even a few noise documents disturbed the model's focused generation from a single gold document. The context might shift a bit topically.

4.7.4 Exp4: Varying Gold Position among Noise Documents

What we observe. When the surrounding context consists entirely of noise documents rather than distractors, the "lost in the middle" positional effect is weakened or absent. For Llama-2 with 19 noise documents in Category A, ROUGE-L ranges only from 0.1825 to 0.1916 across all gold positions, which is way smaller spread compared to what we observed with distractors in exp2b under the same context load. Phi-3-mini shows a similarly weakened positional effect with noise documents across all context loads (9, 19, 29). Qwen2.5 shows negligible positional variation in both categories.

Why it happens. The "lost in the middle" degradation appears to depend not just on context length, but on the coherence of the context around the gold document. When surrounding documents are topically related distractors, the model's attention is pulled toward multiple competing, thematically consistent signals. Thus, locating and attending to the gold document when it is buried in the middle becomes way harder for the model. When the surrounding documents are random noise, there is no thematic competition. The noise documents do not interfere with the gold signal in the same way, and the model can more reliably locate the relevant information regardless of its position.

This finding provides an explanation for the Power of Noise effect observed in exp3. Noise does not merely act as neutral padding that increases context length; it reduces positional retrieval interference. The positional bias documented in [1] may therefore be, in part, an attention competition phenomenon rather than a pure positional encoding artifact.

4.7.5 Exp5: Mixed Distractor–Noise Compositions

What we observe. As the proportion of distractors increases (and noise decreases) at any fixed total context load, performance *consistently improves* according to PropScore and SentenceScore in the Category B setting across all three models. For Phi-3-mini with 19 fillers in Category B, PropScore increases from 0.1478 (all noise, 0 distractors) to 0.1893 (all distractors, 0 noise). For Qwen2.5 with 19 fillers in Category B, the pattern is clearer still, with PropScore rising from 0.1339 to 0.2231 as the distractor proportion increases from 0 to 19. The same monotonic trend holds across 9-filler and 29-filler settings. In Category A, however, the trend is weaker and less consistent, with metrics fluctuating without a clear directional pattern as the distractor–noise ratio changes.

Why it happens. This result may appear to contradict the "Power of Noise" findings from exp3 and exp4, but is in fact consistent with them under a more careful interpretation. In exp5, the gold document is placed at a *fixed position* (the beginning of the context) across all compositions. The positional retrieval problem is therefore minimized by design. Under this condition, what matters most is the quality of the surrounding context: distractors, being topically related to the query, provide useful contextual cues — domain-relevant vocabulary, related concepts, and thematic framing — that help the model generate more fluent, detailed, and on-topic long-form answers. Noise documents, being entirely unrelated, provide no such scaffolding and contribute nothing to generation quality beyond padding.

The "Power of Noise" benefit observed in exp3 and exp4 thus operates through a *different mechanism* than what is tested in exp5. In exp3 and exp4, noise helps by reducing attention competition from misleading distractors when gold position is varied. In exp5, where gold position is fixed at the front and retrieval interference is eliminated, distractors become beneficial rather than harmful because they improve generation quality. This points to an important practical implication: **the optimal RAG context construction strategy depends on the specific failure mode being addressed**. If positional retrieval is the bottleneck, noise injection is beneficial. If generation quality given correct retrieval is the bottleneck, topically relevant retrieved documents — even imperfect distractors — may be preferable to noise.

Chapter 5

Conclusion

This dissertation set out to rigorously evaluate the structural vulnerabilities of Large Language Models (LLMs) within Retrieval-Augmented Generation (RAG) pipelines, specifically focusing on Long-Form Question Answering. While RAG effectively mitigates knowledge freezing and reduces hallucination, recent literature has exposed significant positional biases, namely, the "Lost in the Middle" phenomenon, and the counterintuitive "Power of Noise" paradigm. By shifting the evaluation framework from traditional extractive QA to the complex, generative demands of the ELI5 dataset, this work establishes the boundaries and mechanisms of these phenomena in tasks requiring deep synthesis and reasoning.

5.1 Summary of Experimental Findings

Through a comprehensive series of experiments across three distinct models (Llama-2-7b-chat, Phi-3-mini, and Qwen2.5-7b-instruct), this study directly addressed the three primary research questions outlined in Chapter 1:

- **Manifestation of "Lost in the Middle" in Long-Form QA (RQ1):** The experiments confirm that the "lost in the middle" phenomenon persists to some degree, even in Long-Form QA. In two models, we see a distinct U-shaped performance degradation curve, while the third model seems to be robust to positional effects. This effect was more noticeable in the Category B scenarios, where primacy and recency biases dominated. Qwen2.5 seems to be resilient to this bias.

- **The Influence of Random Noise (RQ2):** The reproduction of the "power of noise" paradigm shows the counterintuitive phenomenon in how LLMs handle long contexts. Substituting topically similar distractors with entirely ran-

dom noise mitigated the performance degradation. Instead of confusing the model, controlled noise acted as a regularizer, allowing the LLM to successfully isolate the gold signal from the context and significantly boosting generation quality compared to prompt contexts loaded with adversarial distractors.

5.2 Regarding (RQ3)

This reproduction effort shows a vital nuance to the understanding of the "lost in the middle" phenomenon. **Positional degradation is not strictly a function of sequence length, but of context coherence and attention competition.** When the prompt context is filled with topically related adverse distractors, the LLM's self-attention mechanism is forced to attend competing, thematically coherent signals. Finding the gold document in the middle of this structure causes severe attentional interference. Conversely, when the surrounding context consists of random noise, this thematic competition becomes negligible. The LLM can easily differentiate the relevant signal regardless of its physical position in the prompt.

Furthermore, the Mixed Composition experiments (Exp 5) demonstrated that if positional interference is bypassed entirely (by placing the gold document first), distractors actually become beneficial, providing generative scaffolding and domain vocabulary that noise cannot provide. This creates a fundamental tension in RAG design: highly relevant retrievals help generate fluent, well-scaffolded answers, but if they score higher than the absolute ground truth and push the gold document to the middle of the context, they actively destroy the model's factual accuracy.

Bibliography

[1] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. *Lost in the Middle: How Language Models Use Long Contexts*. arXiv preprint arXiv:2307.03172, 2023.

[2] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoan Schindeler, Marcello Pelillo, and Alessandro Moschitti. *The Power of Noise: Redefining Retrieval for RAG Systems*. arXiv preprint arXiv:2401.14887, 2024.

[3] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. *Retrieval-augmented generation for knowledge-intensive NLP tasks*. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020.

[4] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. *ELI5: Long form question answering*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3558–3567. Association for Computational Linguistics, 2019.

[5] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv preprint arXiv:2307.09288, 2023.

[6] Marah Abdin, Samer Awadallah, Hany Awadalla, Ahmed Awadallah, et al. *Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone*. arXiv preprint arXiv:2404.14219, 2024.

[7] Qwen Team. *Qwen2.5: A Party of Foundation Models*. `https://qwenlm.github.io/blog/qwen2.5/`, 2024.

[8] Chin-Yew Lin. *ROUGE: A Package for Automatic Evaluation of Summaries.* In *Text Summarization Branches Out*, pages 74–81, 2004.

[9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. *BLEU: a Method for Automatic Evaluation of Machine Translation.* In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[10] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. *BERTScore: Evaluating Text Generation with BERT.* In *International Conference on Learning Representations (ICLR)*, 2020.

[11] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.* In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3982–3992, 2019.

[12] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. *FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation.* arXiv preprint arXiv:2305.14251, 2023.

[13] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Meng Jiang, and Mo Yu. *Dense X Retrieval: What Retrieval Granularity Should We Use?* arXiv preprint arXiv:2312.06648, 2023.

[14] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. *Ragas: Automated Evaluation of Retrieval Augmented Generation.* arXiv preprint arXiv:2309.15217, 2023.

[15] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment.* arXiv preprint arXiv:2303.16634, 2023.

[16] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. *ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems.* arXiv preprint arXiv:2311.09476, 2023.

[17] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena.* arXiv preprint arXiv:2306.05685, 2023.

[18] MohamamdJavad Ardestani, Ehsan Kamalloo, and Davood Rafiei. *LongRecall: A Structured Approach for Robust Recall Evaluation in Long-Form Text*. arXiv preprint arXiv:2508.15085, 2025.

[19] Maojia Song, Shang Hong Sim, Rishabh Bhardwaj, Hai Leong Chieu, Navonil Majumder, and Soujanya Poria. *Measuring and Enhancing Trustworthiness of LLMs in RAG through Grounded Attributions and Learning to Refuse*. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[20] Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, Ian Soboroff, Hoa Trang Dang, and Jimmy Lin. *The Great Nugget Recall: Automating Fact Extraction and RAG Evaluation with Large Language Models*. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25)*, 2025.

[21] João Alberto de Oliveira Lima. *Improving RAG Retrieval via Propositional Content Extraction: A Speech Act Theory Approach*. arXiv preprint arXiv:2503.10654, 2025.

[22] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. *SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking*. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 2288–2292, 2021.

[23] Pengcheng He, Jianfeng Gao, and Weizhu Chen. *DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing*. arXiv preprint arXiv:2111.09543, 2021.

[24] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. *BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation*. arXiv preprint arXiv:2402.03216, 2024.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. arXiv preprint arXiv:1706.03762, 2017.

[26] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. arXiv preprint arXiv:2004.05150, 2020.

[27] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. *Big Bird: Transformers for Longer Sequences*. Advances in Neural Information Processing Systems, 33:17271–17284, 2020.

[28] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2978–2988, 2019.

[29] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. Advances in Neural Information Processing Systems, 35:16344–16359, 2022.

[30] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. *Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context*. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 284–294, 2018.

[31] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. *An Investigation into Neural Net Optimization via Hessian Eigenvalue Density*. In Proceedings of the 36th International Conference on Machine Learning, pages 2232–2241, 2019.

[32] Jian Guan, Zhuoer Feng, Yamei Chen, Ruilin He, Xiaoxi Mao, Changjie Fan, and Minlie Huang. *LOT: A Benchmark for Evaluating Chinese Long Text Understanding and Generation*. Transactions of the Association for Computational Linguistics, 10:434–451, 2022.

[33] Vatsal Sharan, Sham Kakade, Percy Liang, and Gregory Valiant. *Prediction with a Short Memory*. In Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pages 1012–1024, 2018.

[34] Chinnadhurai Sankar, Zhouhan Lin, Sandeep Subramanian, Jingzi He, Sarath Chandar, and Yoshua Bengio. *Do Neural Dialog Systems Remember the Conversation History?*. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4237–4247, 2019.

[35] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Hannaneh Hajishirzi, Luke Zettlemoyer, and Wen-tau Yih. *REPLUG: Retrieval-Augmented Black-Box Language Models*. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational

Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4117–4129, 2024.

[36] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. *In-Context Retrieval-Augmented Language Models*. Transactions of the Association for Computational Linguistics, 11:1316–1331, 2023.

[37] Guanghui Qin, Yukun Feng, and Benjamin Van Durme. *The NLP Task Effectiveness of Long-Range Transformers*. In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 3774–3790, 2023.

[38] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. *How Context Affects Language Models' Factual Predictions*. In Proceedings of the 2020 Conference on Automated Knowledge Base Construction, 2020.

[39] Gautier Izacard and Edouard Grave. *Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering*. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 874–880, 2021.

[40] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. *Dense Passage Retrieval for Open-Domain Question Answering*. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781, 2020.

[41] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. *Unsupervised Dense Information Retrieval with Contrastive Learning*. arXiv preprint arXiv:2112.09118, 2021.

[42] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Hannaneh Hajishirzi, Luke Zettlemoyer, and Wen-tau Yih. *REPLUG: Retrieval-Augmented Black-Box Language Models*. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4117–4129, 2024.

[43] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. *In-Context Retrieval-Augmented Language Models*. Transactions of the Association for Computational Linguistics, 11:1316–1331, 2023.

[44] Ofir Press, Noah A. Smith, and Mike Lewis. *Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation*. In International Conference on Learning Representations, 2022.

[45] Ofir Press, Noah A. Smith, and Mike Lewis. *Shortformer: Better Language Modeling with Shorter Inputs*. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 5493–5505, 2021.

[46] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. *Okapi at TREC-3*. In Proceedings of the Third Text REtrieval Conference (TREC-3), NIST Special Publication 500-225, pages 109–126, 1994.