# CompactPrompt: A Unified Pipeline for Prompt and Data Compression in LLM Workflows

Joong Ho Choi joongho.choi@bny.com BNY Pittsburgh, PA, USA

Ritvika Sonawane\* rsonawan@andrew.cmu.edu Carnegie Mellon University Pittsburgh, PA, USA Jiayang Zhao jiayang.zhao@bny.com BNY Pittsburgh, PA, USA

Vedant Singh vedant.singh@bny.com BNY Pittsburgh, PA, USA Jeel Shah jeel.shah@bny.com BNY Pittsburgh, PA, USA

Avani Appalla avani.appalla@bny.com BNY Pittsburgh, PA, USA

Will Flanagan william.flanagan@bny.com BNY Pittsburgh, PA, USA

#### **Abstract**

Large Language Models (LLMs) deliver powerful reasoning and generation capabilities but incur substantial run-time costs when operating in agentic workflows that chain together lengthy prompts and process rich data streams. We introduce CompactPrompt, an end-to-end pipeline that merges hard prompt compression with lightweight file-level data compression. CompactPrompt first prunes low-information tokens from prompts using self-information scoring and dependency-based phrase grouping. In parallel, it applies ngram abbreviation to recurrent textual patterns in attached documents and uniform quantization to numerical columns, yielding compact yet semantically faithful representations. Integrated into standard LLM agents, CompactPrompt reduces total token usage and inference cost by up to 60% on benchmark dataset like TAT-QA [29] and FinQA [3], while preserving output quality (Results in less than 5% accuracy drop for Claude-3.5-Sonnet [1], and GPT-4.1-Mini [20]) CompactPrompt helps visualize real-time compression decisions

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ACM ICAIF '25, Singapore
© 2025 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXXXXXXXXXX

Filipe Condessa filipe.condessa@bny.com BNY Pittsburgh, PA, USA

and quantify cost-performance trade-offs, laying the ground-work for leaner generative AI pipelines. Demo is available at https://www.youtube.com/watch?v=xr 7Wbzemzg

#### **Keywords**

Large Language Model, Hard Prompt Compression, Soft Prompt Compression, N-gram Abbreviation, Quantization

#### **ACM Reference Format:**

Joong Ho Choi, Jiayang Zhao, Jeel Shah, Ritvika Sonawane, Vedant Singh, Avani Appalla, Will Flanagan, and Filipe Condessa. 2025. CompactPrompt: A Unified Pipeline for Prompt and Data Compression in LLM Workflows. In *Proceedings of The 2nd Workshop on LLMs and Generative AI for Finance, November 15, Singapore (ACM ICAIF '25)*. ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXXXXXXXXXXXXX

#### 1 Introduction

Global financial institutions were among the first to integrate large-scale generative models into production systems, driven by the need for both operational efficiency and consistent output quality. Prompt compression—the process of algorithmically removing or rewriting superfluous tokens from user inputs—directly reduces computational overhead (fewer tokens  $\rightarrow$  lower inference cost and latency) and can standardize prompts across users with varying levels of expertise. Prior work has shown that verbose or unstructured inputs not only increase model cost but also introduce variability in generated outputs [2, 24]. By applying prompt-level optimization (e.g., token pruning, heuristic rewriting, or LLM-based summarization), institutions can achieve measurable

<sup>\*</sup>Project completed during internship at BNY

cost savings while simultaneously improving response consistency and reliability. This dual impact makes prompt compression a compelling technique for enterprise-scale LLM applications.

Empirical studies on Large language models [8], [10] reveal that beyond a certain length, adding tokens degrades reasoning performance. In [9], a marked decline in various reasoning benchmarks once prompts approach roughly 3,000 tokens, which are well below the context-window limits of contemporary transformer models.

One manifestation of this limitation is the "lost-in-the-middle" effect [14]: as input sequences lengthen, transformer attention mechanisms tend to over-emphasize tokens at the beginning and end of the prompt, allocating disproportion-ately less weight to information appearing midway through. The result is an impaired ability to integrate all relevant context during multi-step reasoning tasks.

Together, these findings highlight a fundamental constraint in current transformer architectures: their capacity to reason coherently over very long sequences is inherently bounded. Although prompt-engineering techniques, such as CoT, can partially mitigate this effect, they cannot fully overcome it. Practitioners must therefore strike a careful balance; they need to provide sufficient context to support complex reasoning without exceeding the model's effective working window, and this is where prompt compression comes in.

To achieve the previously mentioned balance, two principal classes of solutions for prompt compression [12]. Hard prompt compression techniques, including SelectiveContext [11] and LLMLingua [5]. Soft prompt compression methods such as GIST [18] and 500×-Compressor [13]. Crucially, all existing work operates on the prompt in isolation, ignoring another major contributor to context size: external data attachments. Traditional file compression (e.g., gzip, numeric quantizers) reduces storage and transmission costs but is incompatible with token-based LLM APIs. In this paper, we introduce 'CompactPrompt', an integrated framework for simultaneous prompt and file-level compression in LLM pipelines that address these gaps by introducing context compression. It is an end-to-end, training-free pipeline that unifies:

- Hard Prompt Pruning via token-level self-information scoring and dependency-driven phrase grouping.
- Textual n-gram Abbreviation, which constructs a reversible dictionary of high-frequency multi-word expressions in attached documents and replaces them with unique, short, human-readable tokens.
- Numeric Quantization, which maps floating-point columns to fixed-bit integer codes under user-specified error tolerances, preserving analytical fidelity.

#### 2 Related Work

SelectiveContext [11], a pioneering prompt compression method, enables LLMs to process 2x more content while saving 36% memory and 32% inference time using vicuna-13b[16] in summary generation task. It uses self-information to identify and delete less informative parts of prompts and employs Spacy's syntactic parsing to ensure grammatical integrity.

LLMLingua [5] achieves up to 20x prompt compression with minimal performance loss using trained language models. It employs a coarse-to-fine compression method, featuring a budget controller for semantic integrity.

Another method is Nano-Capsulator [4], which transforms an input prompt into a succinct natural-language summary, which is then fed into the target LLM. By pruning irrelevant content and reconstructing fluent sentences, its finetuned Vicuna-7B compressor [15] runs independently of the downstream model. Unlike off-the-shelf summarizers, Nano-Capsulator incorporates a semantic-preservation loss to retain task-critical meaning and a reward function to optimize prompt utility, boosting end-task performance. On the downside, the extra compression network incurs memory overhead, and its two-stage inference pipeline demands more compute than simple encoding.

Beyond these, two other general "hard-prompt" compressors have emerged. LongLLMLingua extends LLMLingua by reordering documents and recovering subsequences to double the effective window [6], while AdaComp adaptively selects context based on query complexity and retrieval quality [26]. LLMLingua-2 further refines this via data-distillation to build a compressed corpus and a classifier that predicts which tokens to keep [21].

Although prior work (*e.g.* [28], [12]) has studied a variety of hard prompt compression and soft prompt rewriting strategies, we found no existing method that directly applies n-gram abbreviation in the context of prompt pruning and data compression. Most known approaches either operate at the token-level removal (filtering [5]), sentence-level selection, or generative rewriting (SCOPE [27]).

#### 3 Design

Our combination of hard prompt pruning, n-gram abbreviation and numeric quantization is novel and addresses a gap in the literature by allowing interpretable, reversible compression that leverages repeated multi-token patterns rather than purely individual token importance.

Hard prompt pruning targets redundant or low-information content in natural language prompts, preserving semantic

meaning while reducing token count; our hard prompt compression strategy leverages both static and dynamic selfinformation measures to identify and prune low-value content from the input prompt, ensuring that the most informative tokens are preserved within a user- specified budget. N-gram abbreviation complements this by focusing on repetitive patterns in attached documents, offering lossless compression for domain-specific terminology or common phrases. Numeric quantization addresses the often-overlooked issue of floating-point precision in data-heavy contexts, allowing for significant reduction in token usage for numerical data while maintaining analytical fidelity within userdefined error bounds. This holistic approach ensures that CompactPrompt can effectively compress diverse input types commonly found in real-world LLM applications, from textheavy prompts to data-rich attachments, without requiring model retraining or sacrificing human readability.

#### 3.1 **Hard Prompt Compression**

Static Self-Information. We begin by constructing a large offline corpus-comprising sources, consisting of Wikipedia, ShareGPT conversations, and arXiv articles. Then we compute unigram frequencies f(t) for every token t as shown below.

$$f(t) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{w_i = t\} = \frac{\#\{i : w_i = t\}}{N}$$
 (1)

From these frequencies, we derive the static self-information score using  $[I(T) = -\log_2 p(T)]$ .

- 3.1.2 Dynamic Self-Information. To capture context-sensitive importance, we query a pretrained LLM or a lightweight scoring agent at runtime. Given a token t and its preceding context (c), we obtain the model probability  $(P_{\text{model}}(t \mid c))$ and compute dynamic self-information. This score reflects the novel information contributed by t in its specific prompt context, complementing the corpus-level view.
- 3.1.3 Combined Scoring and Phrase-Level Pruning. We integrate static and dynamic measures through a weighted combination strategy, employing a simple arithmetic mean when both scores exhibit less than 10% relative difference, as shown below in formula (2) and (3).

$$\Delta = \frac{\left| s_{\rm dyn} - s_{\rm stat} \right|}{s_{\rm stat}} \tag{2}$$

$$\Delta = \frac{\left|s_{\text{dyn}} - s_{\text{stat}}\right|}{s_{\text{stat}}}$$

$$C(s_{\text{stat}}, s_{\text{dyn}}) = \begin{cases} \frac{s_{\text{stat}} + s_{\text{dyn}}}{2}, & \Delta \le 0.1, \\ s_{\text{dyn}}, & \Delta > 0.1. \end{cases}$$
(3)

The reasoning behind 10% threshold is because lower (e.g. 5%) or higher (e.g. 15%) cut-offs tilt too far toward one method, either over-favoring dynamic noise or under-reacting to true context changes. Ten percent offers a pragmatic compromise.

When both methods produce similar scores, their average provides a stable estimate that benefits from both corpuslevel statistics and contextual information. However, when scores differ significantly, the dynamic method's ability to capture context-sensitive importance through querying the pre-trained model at runtime is more reliable than static corpus statistics.

### **Textual n-gram Abbreviation**

N-gram abbreviation uses a method similar to LZW (Lempel-Ziv-Welch) algorithm [25] that compresses text by losslessly replacing common sequences with short, human-readable tokens, so you can search, index or apply Natural Language Processing (NLP) directly on the stream without full decompression. It delivers substantial size reduction with simple table-lookup encoding-more transparent and lightweight than heavy binary compression like ZIP[7] or bzip2[23], and reversible (unlike lossy embeddings) at minimal compute cost. To compress attached text, we employ a userconfigurable n-gram abbreviation pipeline.

- 3.2.1 Extraction and Frequency Analysis. First, the user specifies an n-gram length (n) (commonly between 2 and 5). We extract all n-grams from the document corpus and compute their frequencies. These frequencies are visualized in a histogram, enabling the identification of the top (K) most frequent patterns (typically (K=100 to 150)).
- 3.2.2 Dictionary Construction. We assign each of the top-(K) n-grams a unique placeholder token (e.g., "ABC1", "BD2"). This mapping is stored in a metadata table, ensuring lossless round-trip reconstruction.
- 3.2.3 Contextual Replacement and Reversibility. We replace every occurrence of the most-common n-grams with their placeholders, taking care to resolve overlapping matches and preserve punctuation. At downstream stages or upon user request, placeholders are substituted back to their original n-grams using the metadata table, guaranteeing exact reversibility.

#### 3.3 Numerical Quantization

For numerical columns in structured data attachments, we implement two complementary quantization schemes to reduce token payload while bounding approximation error, namely uniform integer quantization and K-means based quantization.

3.3.1 Uniform Integer Quantization. Given a numeric column (x), we compute its minimum and maximum values  $(\min_x, \max_x)$  and select a bit-width (b), yielding  $(L = 2^b)$  quantization levels. Each value  $(x_i)$  is encoded as shown in equation (4) below and then stored as an integer.

$$q_i = \text{round}\left(\frac{x_i - \min_x}{\max_x - \min_x} (L - 1)\right)$$
 (4)

The tuple  $(\min_x, \max_x, b)$  is retained to reconstruct  $(\hat{x}_i)$  with maximum absolute error as defined in equation (5) below.

$$\hat{x}_i = \min_{x} + \frac{q_i}{L-1} \left( \max_{x} - \min_{x} \right),$$

$$\varepsilon_{\max} = \frac{\max_{x} - \min_{x}}{L-1}$$
(5)

3.3.2 *K-Means-Based Quantization.* Alternatively, we apply k-means clustering to the column values, choosing (k) centroids  $(\mu_1, \ldots, \mu_k)$ . Each  $x_i$  is mapped to the nearest centroid index. Reconstruction uses the stored centroids, which minimizes average squared error.

### 3.4 Representative Example Selection for Few-Shot

To further reduce contextual size when including few-shot exemplars, we select a small set (i.e. 3) of representative data points via clustering and silhouette analysis.

- 3.4.1 Embedding and Normalization. We choose all-mpnet-base-v2 [22] for the embedding model. We embedded textual data into high-dimensional vectors and standardized numeric features (zero mean, unit variance).
- 3.4.2 Clustering with Silhouette Optimization. We run k-means over the embedded or normalized data for  $k \in 5-50$ . For each (k), we compute the average silhouette score—a measure of cluster cohesion versus separation—to identify the optimal cluster count  $(k^*)$ , given the value of k that maximizes this score.
- 3.4.3 Representative Points Selection. Within each of the  $(k^*)$  clusters, we select the point closest to the centroid as a "prototype." These prototypes serve as few-shot examples, capturing the diversity of the dataset in a compact set.

#### 3.5 Semantic Fidelity and Similarity Metrics

To evaluate the semantic integrity of compressed outputs, we employ embedding-based similarity measures.

3.5.1 Full-Dimensional Cosine Similarity. Original and compressed text segments are encoded with all-mpnet-base-v2 [22]. We compute  $cosine(E_{orig}, E_{comp})$  and report both the

mean and 5th percentile scores to ensure worst-case fidelity remains high.

3.5.2 Human Validation. The scale of evaluation ranges from 1 to 5, with 5 being completely identical and thus good. Three evaluators were trained to have same evaluation standard. With 15 examples from each TAT-QA and Fin-QA(overall 30), they were shown randomly sampled examples that represent different scores asked to evaluate other 15 examples from each TAT-QA and Fin-QA(overall 30 again) to ensure their score alignment.

After this training, they rated 90 original & compressed prompt pairs from TAT-QA and Fin-QA for semantic equivalence. We set the score ranging from 1-5, with 1 being completely different and 5 being completely identical. The mean score (4.1/5) correlated with cosine similarities. Fewer than 5 percents of cases showed high vector similarity but lower human ratings, highlighting rare nuance shifts missed by embedding alone.

#### 4 CompactPrompt Tool

### 4.1 CompactPrompt GUI Workflow

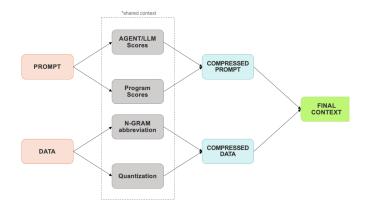


Figure 1: CompactPrompt Workflow

The Tool is divided into two parts: "Prompt Compression" and "Data Compression." Figure 1 shows the workflow of the tool. In the Prompt Compression view shown in Figure 2, users can paste or type their full prompt in the text editor and can choose which LLM calculates self-information scores via a Scorer agent. At the moment, we support GPT-4-Omni [19], GPT-4.1-Mini [20], Claude 3.5 Sonnet [1], and Llama-3.3-70B-instruct [17].

Refer to the Demo video: https://www.youtube.com/watch?v=xr 7Wbzemzg

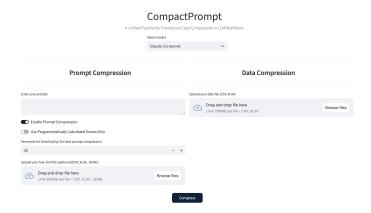


Figure 2: CompactPrompt App Preview

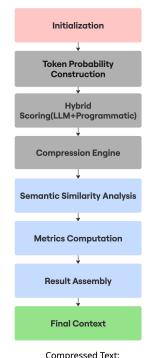
## 4.2 CompactPrompt non-GUI/Library version Workflow

Following the development of CompactPrompt as a GUI application, we recognized the need to extend its utility beyond individual prompt engineering workflows to accommodate enterprise-scale deployment and systematic integration within organizational AI infrastructure. To address this requirement, we architected a library implementation through the UnifiedCompactPromptPipeline (UCPP) class, which encapsulates our comprehensive compression methodology within a programmatically accessible framework. This library instantiation enables seamless integration with existing financial technology stacks, allowing compression capabilities to be incorporated directly into automated trading systems, risk management platforms, and regulatory compliance pipelines. The UCPP implementation maintains the full algorithmic rigor of our original approach while providing enterprise-grade scalability and operational flexibility essential for production financial environments.

The UnifiedCompactPromptPipeline orchestrates an eightstage workflow encompassing initialization, token probability construction, hybrid scoring through combined large language model and programmatic evaluation, our compression engine, semantic similarity analysis, comprehensive metrics computation, result assembly, and utility function execution, as shown in Figure 3. Upon instantiation with model-specific parameters

(e.g., pipeline = UCPP(model='Claude-3.5-Sonnet')), the system processes input prompts through this deterministic pipeline, ultimately delivering both the compressed prompt and detailed compression analytics directly to the terminal interface. This programmatic approach fundamentally transforms CompactPrompt from a standalone tool into a deployable enterprise asset, enabling organizations to systematically reduce LLM operational costs across their entire AI infrastructure while maintaining quantifiable performance

pipeline = UnifiedCompactPromptPipeline(model=claude)
prompt = "You are an AI-financial analysis assistant. Your task is to read the
following excerpt and report the total revenue and net income."



ai-financial analysis assistant task following excerpt report total revenue income

Figure 3: Non-GUI Preview

standards. The library's terminal-based output facilitates direct integration with database query systems and automated trading algorithms, thereby establishing a seamless bridge between prompt optimization and production financial applications where computational efficiency directly translates to competitive advantage and operational cost reduction. The UnifiedCompactPromptPipeline orchestrates a meticulously engineered eight-stage workflow encompassing initialization, token probability construction, hybrid scoring through combined large language model and programmatic evaluation, our compression engine, semantic similarity analysis, comprehensive metrics computation, result assembly, and utility function execution. Upon instantiation with model-specific parameters

(e.g., pipeline = UCPP(model='Claude-3.5-Sonnet')), the system processes input prompts through this deterministic pipeline, ultimately delivering both the optimally compressed prompt and detailed compression analytics directly to the terminal interface. This programmatic approach fundamentally transforms CompactPrompt from a standalone tool into a deployable enterprise asset, enabling organizations to systematically reduce LLM operational costs across their entire AI infrastructure while maintaining quantifiable

performance standards. The library's terminal-based output facilitates direct integration with database query systems and automated trading algorithms, thereby establishing a seamless bridge between prompt optimization and production financial applications where computational efficiency directly translates to competitive advantage and operational cost reduction.

#### 5 Downstream Task Performance

#### 5.1 Task Setup

To evaluate our compression methodology, we conduct experiments on two complementary datasets: TAT-QA [29] and Fin-QA [3]. TAT-QA is a large-scale benchmark for question answering that presents a challenge requiring integration of structured tabular data with unstructured narrative passages, where queries demand cross-referential reasoning between financial tables and their explanatory annotations. The dataset's long-form contexts populated with redundant tabular tokens create natural vulnerabilities to token bloat and repetitive patterns which are precisely the inefficiencies our CompactPrompt methodology addresses through phrase grouping and abbreviation. Fin-QA is derived from financial reports and emphasizes quantitative reasoning operations while maintaining semantic grounding in specialized financial discourse. Its stringent requirements for numerical precision and frequent deployment of recurring financial terminology generate verbose contexts that interweave structured numerical data with dense financial narratives, reflecting high-stakes applications where errors have profound consequences.

The strategic pairing of these datasets creates a comprehensive evaluation framework spanning multiple complexity dimensions. Structurally, TAT-QA's hybrid table-narrative format contrasts with Fin-QA's domain-dense financial text, collectively covering heterogeneous input modalities from structured tabular data to numerically intensive content. While TAT-QA emphasizes cross-referential information synthesis, Fin-QA prioritizes precision arithmetic and financial logic, ensuring our compression methodology is evaluated on both semantic preservation and quantitative accuracy retention. The datasets exhibit complementary error profiles; while TAT-QA failures typically manifest as information retrieval breakdowns, Fin-QA errors arise from numerical miscalculations or semantic misinterpretation of financial terminology, stress-testing our approach across information compression fidelity and numerical compression safety. This combination provides rigorous validation across the full spectrum of contemporary QA challenges.

In addition to dataset-level evaluation, we conduct a controlled sensitivity study on the hyper-parameters governing our abbreviation-based data compression, specifically the

choice of n in n-gram extraction and the number of top-n frequent tokenized sequences retained. By systematically varying these parameters, we assess how compression aggressiveness influences downstream accuracy across both TAT-QA and Fin-QA. The smaller configurations such as bigrams with Top-3 yield modest reductions (1.12×) compared with baseline prompt, while larger n-grams combined with broader Top-n lists produce progressively stronger effects, with 4-grams at Top-5 achieving the highest compression ratio of 1.44×. This trend illustrates the inherent trade-off between efficiency and fidelity-higher n and Top-n settings maximize token savings but risk discarding subtle semantic cues. To ensure generalizability, the evaluation spans a diverse set of state-of-the-art LLMs, including GPT-4-Omni [19], GPT-4.1-Mini [20], Claude 3.5 Sonnet [1], and Llama-3.3-70B-instruct [17], thereby capturing model-specific sensitivities to context abbreviation. The results provide actionable insights into hyper-parameter tuning, guiding practitioners to balance efficiency and accuracy in real-world deployments.

#### 5.2 Results

We evaluated CompactPrompt across TAT-QA and Fin-QA, comparing multiple compression strategies and ablations over n-gram length and top-n frequency thresholds, as shown in the tables in the appendix. Experiments span four leading GPT-4-Omni [19], GPT-4.1-Mini [20], Claude 3.5 Sonnet [1], and Llama-3.3-70B-instruct [17]. And we found diverse sensitivities on the models to compression and abbreviation.

#### 5.2.1 Ablation on N-gram and Top-N

. **Top-N** (T) controls the breadth of the abbreviation dictionary. Small T focuses on highly redundant patterns (maximizing token savings per substitution) and minimizes disruption; large T quickly enters diminishing returns and elevates semantic risk.

N-gram size (G) governs the semantic granularity of each substitution. In practice, we target short, highly recurrent bi-grams (e.g. "per share," "interest expense") because they yield the biggest compression gains with zero ambiguity. Very long n-grams often encode one-off, document-specific phrasing—but our scheme remains lossless by design: every replaced n-gram is recorded in a reversible lookup table, so the exact original text can be reconstructed. In other words, even if we substitute long, instance-specific sequences, no information is ever discarded—only temporarily aliased and fully recoverable.

**Best configuration:** The largest improvement occurs at T = 3, G = 2 ( $\Delta = +5.0$ ; annotated "Top 1" in the figure). Targeting only the three most frequent bi-grams concentrates compression on truly repetitive phrasing while avoiding substitutions that would disrupt local semantics. This setting delivers a strong cost–performance trade-off across models.

| Top N/<br>N-gram | Method     | Claude-3.5-<br>Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B<br>-instruct |
|------------------|------------|-----------------------|------------|--------------|----------------------------|
| 3/2              | Baseline   | 66                    | 54         | 60           | 54                         |
| 2/2              | Compressed | 72                    | 42         | 56           | 46                         |
| 2/3              | Compressed | 70                    | 46         | 64           | 46                         |
| 2/4              | Compressed | 70                    | 44         | 58           | 42                         |
| 3/2              | Compressed | 68                    | 58         | 54           | 38                         |
| 3/3              | Compressed | 74                    | 46         | 58           | 42                         |
| 3/4              | Compressed | 74                    | 50         | 56           | 38                         |
| 4/2              | Compressed | 64                    | 44         | 54           | 44                         |
| 4/3              | Compressed | 72                    | 40         | 58           | 42                         |
| 4/4              | Compressed | 66                    | 54         | 54           | 40                         |
| 5/2              | Compressed | 68                    | 44         | 56           | 38                         |
| 5/3              | Compressed | 70                    | 22         | 60           | 38                         |
| 5/4              | Compressed | 66                    | 60         | 58           | 42                         |

Figure 4: Table showing performance of different models across varying Top N and N-gram configurations

**Next-best configurations:** Moderate gains are observed for T=2, G=4 ( $\Delta\approx +1.0$ ; "Top 2") and T=3, G=4 ( $\Delta\approx +0.5$ ; "Top 3"). These combinations expand coverage to slightly longer, domain-specific expressions (*e.g.*, recurring financial terms) without overwhelming the prompt with substitutions.

**Failure region:** Performance drops cluster in the aggressive quadrant  $T \ge 4$  (across most G), where crosses dominate. Abbreviating too many patterns increases the chance of (i) replacing context-bearing phrases, (ii) creating dense placeholder text that is harder for models to resolve, and (iii) introducing overlap/collision effects between placeholders—each of which degrades reasoning fidelity.

**Model sensitivity:** Claude 3.5 Sonnet and GPT-4.1-Mini show consistent gains in the conservative regime  $T \leq 3$  with  $G \in \{2,4\}$ . GPT-4-Omni exhibits mixed responses (both sizable gains and losses), suggesting tighter guardrails on T. Llama-3.3-70B is comparatively fragile: it benefits at G = 2,  $T \leq 3$  but degrades rapidly as T increases.

## 5.2.2 CompactPrompt Performance with best performing N-gram and TopN values

. After applying CompactPrompt with n-gram = 2 and TopN = 3, the size of the prompts for both TAT-QA and Fin-QA is cut roughly in half. TAT-QA's token count achieves about a  $2.35\times$  reduction, or 58% fewer tokens, and Fin-QA's about a  $2.12\times$  reduction, or 53% fewer tokens. This token reduction is achieved while generally maintaining similar or slight improvement for the models.

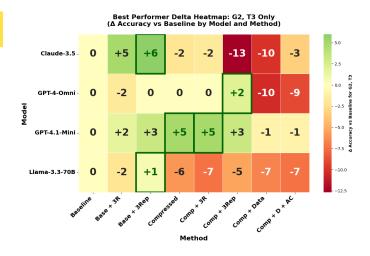


Figure 5: Accuracy deltas ( $\Delta$ ) relative to baseline on TAT-QA with n-gram size 2 and Top-3 abbreviation. Claude-3.5 and GPT-4.1-Mini show improvements up to +6 and +5 points, while GPT-4-Omni and Llama-3.3-70B remain mostly stable or decline under compression.

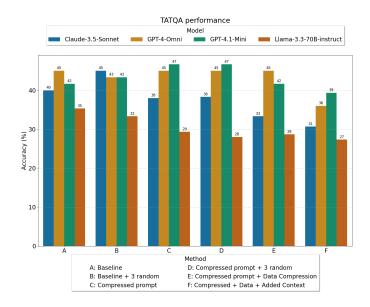


Figure 6: Accuracy on TAT-QA with baseline and compression variants using three random exemplars. GPT-4.1-Mini benefits from compression (+5%), while Claude-3.5 and GPT-4-Omni maintain stable performance, and Llama-3.3-70B shows declines under most compressed settings.

We evaluate CompactPrompt on the TAT-QA benchmark, which requires cross-referencing structured tables with accompanying narrative passages. Figure 6 shows absolute

model accuracies across prompting methods, while Figure 5 presents accuracy deltas relative to baseline under the best-performing hyperparameter setting (T = 3, G = 2).

Baseline vs. Compression: As shown in Figure 6, uncompressed prompts (Method A) yield accuracies between 33–45% depending on the model. Applying CompactPrompt's compression (Method C) maintains this baseline performance and, in several cases, improves it. The heatmap in Figure 5 highlights that Claude 3.5 Sonnet (+6) and GPT-4.1-Mini (+5) achieve the largest gains, GPT-4-Omni remains stable, and Llama-3.3-70B shows a modest improvement (+1). Together, the two plots confirm that CompactPrompt reduces prompt size without sacrificing accuracy, while often delivering a measurable boost.

#### **Effect of Exemplars**

When compression is combined with exemplar-based prompting, Figures 5 show effects on adding the random exampels from the TATQA dataset. Adding three representative examples (Method C and E) sustains the gains for Claude 3.5 Sonnet and GPT-4.1-Mini (both +5) while keeping GPT-4-Omni near baseline. In contrast, adding random exemplars (Method B or D-random) and also shown in 6 produces less consistent improvements, reinforcing the importance of careful example selection.

#### **Data Compression and Added Context**

Figures 6 further illustrate that integrating document-level data compression (Method E) or reintroducing the abbreviation dictionary as added context (Method F) yields mixed outcomes. For GPT-4.1-Mini, performance remains same level baseline, but other models exhibit small declines. This suggests that while textual abbreviation provides a reliable efficiency–fidelity trade-off, tabular data compression must be applied selectively to avoid disrupting cross-referential reasoning.

#### **Cross-model insights:**

- Claude 3.5 Sonnet is the most robust across settings, consistently outperforming its baseline under compression in both Figures 5 and 6.
- GPT-4.1-Mini also shows dependable improvements, particularly with representative exemplars.
- GPT-4-Omni maintains stable accuracy across all settings, showing that CompactPrompt introduces no regressions.
- Llama-3.3-70B improves slightly under light compression but is more sensitive to aggressive configurations, as reflected in the heatmap's negative deltas.

#### Summary of findings in TATQA:

Across both Figures 5 and 6, CompactPrompt achieves up to +6 accuracy points while cutting token usage by more than half. The key outcome is that model performance is

preserved under compression, fulfilling the primary goal of efficiency without loss. The consistent gains for Claude 3.5 Sonnet and GPT-4.1-Mini demonstrate that compression can even enhance reasoning quality—a welcome bonus beyond cost reduction.

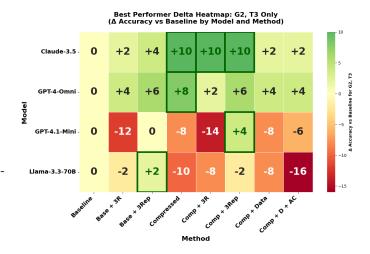


Figure 7: Accuracy deltas ( $\Delta$ ) relative to baseline on FinQA with n-gram size 2 and Top-3 abbreviation. Claude-3.5 and GPT-4-Omni achieve gains up to +10 and +8 points, while GPT-4.1-Mini and Llama-3.3-70B experience notable drops.

We further evaluate CompactPrompt on Fin-QA, a benchmark emphasizing numerical reasoning over financial text. Figures 8 and 9 report absolute accuracies with random versus representative exemplars, while Figure 7 shows deltas relative to baseline under the best-performing hyperparameter setting (G = 2, T = 3).

**Baseline vs. Compression** Figure 8 shows baseline accuracies ranging from 54–66% across models. Compressed prompts (Method C) maintain these levels for most models and even surpass them in some cases: Claude 3.5 Sonnet improves from 66% to 76% (+10), while GPT-4-Omni also achieves notable gains (+6 to +8 as highlighted in Figure 7). These improvements demonstrate that CompactPrompt can preserve numerical reasoning accuracy while reducing token counts, and in certain cases enhance performance.

Random vs. Representative Exemplars A clear distinction emerges when comparing Figures 8 and 9. With random exemplars, results are less consistent: GPT-4.1-Mini and Llama-3.3-70B occasionally show declines, indicating sensitivity to exemplar choice. However, with representative exemplars (Figure 9), performance stabilizes across models. Claude 3.5 Sonnet sustains its +10 point gain, GPT-4-Omni improves by up to +8, and even GPT-4.1-Mini recovers from

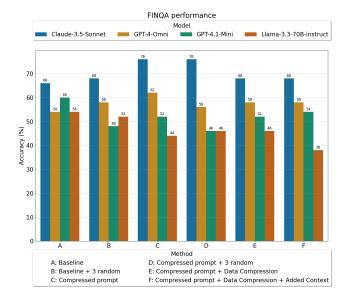


Figure 8: Accuracy on FinQA with baseline and compression variants using three random exemplars. Claude-3.5 achieves the highest performance (76%) under compressed prompt settings, while GPT-4.1-Mini and Llama-3.3-70B show reduced stability under compression.

its earlier declines to exceed baseline in selected configurations. This highlights the importance of exemplar quality: representative examples yield reliable improvements under compression, while random examples can introduce noise.

**Data Compression and Added Context** As with TAT-QA, integrating data-level compression (Method E) or appending abbreviation dictionaries (Method F) yields mixed results. In Figure 7, Claude 3.5 Sonnet maintains +10 across compressed settings, and GPT-4-Omni shows stable gains. By contrast, GPT-4.1-Mini and Llama-3.3-70B are more sensitive, sometimes registering modest declines. These outcomes suggest that while CompactPrompt consistently preserves baseline accuracy, aggressive data compression should be tuned carefully depending on the model.

#### Cross-model insights

- Claude 3.5 Sonnet benefits most, achieving sustained improvements of +10 points in compressed conditions (Figures 8 and 9).
- GPT-4-Omni also gains consistently (+6 to +8), showing strong robustness.
- GPT-4.1-Mini is more sensitive to exemplar choice: it declines with random examples but recovers when representative ones are used.
- Llama-3.3-70B shows smaller gains (+2 in some settings) and is more affected by aggressive compression.

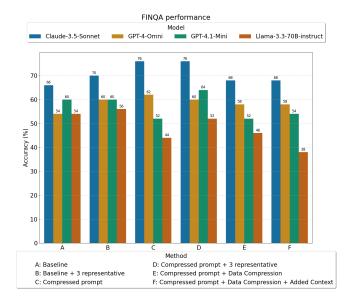


Figure 9: Accuracy on FinQA with baseline and compression variants using three representative exemplars. Representative sampling yields stronger gains for GPT-4.1-Mini (+5% over baseline) and more consistent improvements across models compared to random exemplars.

**Summary of findings in Fin-QA:** Across Figures 7 – 9, FinQA experiments confirm that CompactPrompt achieves substantial token savings without accuracy loss, fulfilling the primary goal of compression. Moreover, for Claude 3.5 Sonnet and GPT-4-Omni, compression paired with representative examples yields clear accuracy improvements (+6 to +10 points), showing that efficiency and reasoning quality can be advanced together when hyperparameters and exemplars are tuned appropriately.

**Overall Insights:** Across both TAT-QA and Fin-QA, the results demonstrate that CompactPrompt reliably reduces token usage by more than half while preserving baseline accuracy—a crucial outcome for practical deployment. The ablation study highlights that careful tuning of abbreviation hyperparameters (G=2,T=3) is key to striking the right balance between efficiency and fidelity. Representative exemplars further amplify these benefits, stabilizing performance and enabling consistent gains across models. While Claude 3.5 Sonnet and GPT-4-1-Mini show the strongest improvements, GPT-4-Omni remains robust under all settings, and even Llama-3.3-70B benefits under lighter compression. Taken together, these findings confirm that CompactPrompt can be deployed with confidence: the primary goal of efficiency is achieved without loss, and in many cases, accuracy

is enhanced—a positive signal for integrating compression into high-stakes financial workflows.

#### 6 Conclusion

CompactPrompt demonstrates that a training-free, hardcompression approach can substantially reduce prompt length, achieving up to 60% token savings on TAT-QA and FinQA while maintaining, and in some cases improving, QA accuracy. Representative exemplar selection further enhances performance by capturing the underlying characteristics of the dataset more faithfully than random exemplars. We also show that high embedding-based semantic similarity (≥ 0.92) generally indicates safe compression, though moderate similarity drops do not necessarily harm downstream performance. Interestingly, prompts compressed with Claude 3.5 Sonnet exhibited slightly lower semantic embedding similarity (0.941) than those from GPT-4.1-Mini (0.952), yet performed better on TAT-QA—underscoring that semantic metrics alone cannot guarantee task-level fidelity. We therefore advocate a multi-pronged evaluation strategy combining embedding-based thresholds with human review, LLM-as-ajudge assessments, and direct task accuracy checks to ensure that efficiency gains do not come at the expense of quality.

Global financial institutions have proven to be among the earliest adopters of generative AI. As banks pursue democratization of LLM-based systems, shareholders rightly expect both return on investment and strict adherence to regulatory standards around privacy. Prompt compression directly supports both goals: lowering costs by reducing token usage and simultaneously offering a pathway to limit inadvertent disclosure of sensitive client or firm data.

#### 7 Future Work

Looking ahead, several promising directions can extend the utility of CompactPrompt:

- Adaptive compression. Future versions could dynamically adjust the compression budget based on task complexity, context length, or real-time performance signals.
- **Privacy-aware compression.** Assigning higher weights to sensitive fields (*e.g.*, identifiers, account numbers, social security numbers) would allow compression to function as a lightweight privacy filter, removing highrisk tokens before data reaches an LLM.
- Integration with enterprise pipelines. Embedding CompactPrompt into financial workflows such as regulatory reporting, automated compliance, and portfolio monitoring could help standardize cost-efficient deployment at scale.

- **Beyond text and tables.** Extending compression techniques to multimodal contexts (*e.g.*, charts, PDF documents, or time-series feeds) would broaden applicability to real-world financial analysis tasks.
- Reduced reliance on RAG. While retrieval-augmented generation (RAG) remains powerful, compression could enable larger proprietary documents to be directly incorporated into prompts, offering faster inference and simpler deployment when RAG infrastructure is costly to maintain.

CompactPrompt is proprietary software owned by The Bank of New York (BNY). In accordance with company policies and intellectual property regulations, we are unable to make the tool publicly available. Access to the tool is restricted to internal use and authorized personnel only hence external distribution or open-source release is not permitted.

#### References

- Anthropic. 2024. Claude 3.5 Sonnet. Model announcement on Anthropic website. Available at https://www.anthropic.com/news/claude-3-5-sonnet.
- [2] Tom B. Brown, Ben Mann, Nick Ryder, Mira Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In Advances in Neural Information Processing Systems, Vol. 33. 1877–1901.
- [3] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A Dataset of Numerical Reasoning over Financial Data. *Proceedings of EMNLP 2021* (2021).
- [4] Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. 2024. Learning to Compress Prompt in Natural Language Formats. arXiv:2402.18700 [cs.CL] https://arxiv.org/abs/2402.18700
- [5] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models. arXiv:2310.05736 https://arxiv.org/abs/2310. 05736
- [6] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression. arXiv:2310.06839 [cs.CL] https://arxiv.org/abs/2310.06839
- [7] Phil Katz. 1989. PKZIP Application Note. https://pkware.cachefly.net/ webdocs/casestudies/APPNOTE.TXT
- [8] Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same Task, More Tokens: the Impact of Input Length on the Reasoning Performance of Large Language Models. arXiv:2402.14848 [cs.CL] https://arxiv.org/abs/2402.14848
- [9] Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same Task, More Tokens: the Impact of Input Length on the Reasoning Performance of Large Language Models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024). Association for Computational Linguistics, Bangkok, Thailand, 15339–15353. https://aclanthology.org/2024.acl-long.818.pdf
- [10] Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context LLMs Struggle with Long In-context Learning. arXiv:2404.02060 [cs.CL] https://arxiv.org/abs/2404.02060

- [11] Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. Compressing Context to Enhance Inference Efficiency of Large Language Models. arXiv:2310.06201 https://arxiv.org/abs/2310.06201
- [12] Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. 2024. Prompt Compression for Large Language Models: A Survey. arXiv:2410.12388 [cs.CL] https://arxiv.org/abs/2410.12388
- [13] Zongqian Li, Yixuan Su, and Nigel Collier. 2024. 500xCompressor: Generalized Prompt Compression for Large Language Models. arXiv:2408.03094 https://arxiv.org/abs/2408.03094
- [14] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. arXiv:2307.03172 [cs.CL] https://arxiv.org/abs/2307.03172
- [15] LMSys. 2023. vicuna-7b-v1.5. https://huggingface.co/lmsys/vicuna-7b-v1.5. Version 1.5.
- [16] LMSys. 2024. vicuna-13b-v1.5. https://huggingface.co/lmsys/vicuna-13b-v1.5. Version 1.5.
- [17] Meta. 2025. Llama 3.3. https://www.llama.com/docs/model-cards-and-prompt-formats/llama3\_3/. Accessed: 2025-09-29.
- [18] Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2024. Learning to Compress Prompts with Gist Tokens. arXiv:2304.08467 https://arxiv.org/abs/2304.08467
- [19] OpenAI. 2024. GPT-4o. https://platform.openai.com/docs/models/gpt-4o. Accessed: 2025-09-29.
- [20] OpenAI. 2025. GPT-4.1-mini. https://platform.openai.com/docs/models/gpt-4.1-mini. Accessed: 2025-09-29.
- [21] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data Distillation for Efficient and Faithful Task-Agnostic Prompt Compression. arXiv:2403.12968 [cs.CL] https://arxiv.org/abs/2403.12968
- [22] Nils Reimers and Iryna Gurevych. 2021. all-mpnet-base-v2. https://huggingface.co/sentence-transformers/all-mpnet-base-v2.
- [23] Julian Seward. 1996. bzip2 and libbzip2 A Block-Sorting File Compressor. https://sourceware.org/bzip2/ Version 0.2.0.
- [24] X. Shi, C. Zhao, and Y. Liu. 2023. Optimizing Prompt Engineering: A Comprehensive Survey. Journal of Artificial Intelligence Research 78 (2023), 105–134.
- [25] Terry A. Welch. 1984. A Technique for High-Performance Data Compression. *Computer* 17, 6 (June 1984), 8–19. doi:10.1109/MC.1984. 1659158
- [26] Qianchi Zhang, Hainan Zhang, Liang Pang, Hongwei Zheng, and Zhiming Zheng. 2024. AdaComp: Extractive Context Compression with Adaptive Predictor for Retrieval-Augmented Large Language Models. arXiv:2409.01579 [cs.CL] https://arxiv.org/abs/2409.01579
- [27] Tinghui Zhang, Yifan Wang, and Daisy Zhe Wang. 2025. SCOPE: A Generative Approach for LLM Prompt Compression. arXiv:2508.15813 [cs.CL] https://arxiv.org/abs/2508.15813
- [28] Zheng Zhang, Jinyi Li, Yihuai Lan, Xiang Wang, and Hao Wang. 2025. An Empirical Study on Prompt Compression for Large Language Models. arXiv:2505.00019 [cs.CL] https://arxiv.org/abs/2505.00019
- [29] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance. 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). Association for Computational Linguistics, Online, 3277–3287. doi:10.18653/v1/2021.acl-long.254

### A Appendix

Table 1: TAT-QA Accuracy (%) for different models, methods, Top-N and n-gram under Baseline setting

| Top N/N-gram | Method                      | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
|--------------|-----------------------------|-------------------|------------|--------------|------------------------|
| 3/2          | Baseline                    | 40                | 45         | 42           | 35                     |
| 3/2          | Baseline + 3 random         | 45                | 43         | 43           | 33                     |
| 3/2          | Baseline + 3 representative | 46                | 45         | 45           | 36                     |

Table 2: TAT-QA Accuracy (%) for different models, methods, Top-N and n-gram under compressed setting

| Top N/N-gram | Method                        | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
|--------------|-------------------------------|-------------------|------------|--------------|------------------------|
| 3/2          | Compressed Prompt             | 38                | 45         | 47           | 29                     |
| 3/2          | Compressed + 3 random         | 38                | 45         | 47           | 28                     |
| 3/2          | Compressed + 3 representative | 27                | 47         | 45           | 30                     |

Table 3: TAT-QA Accuracy (%) for different models, methods, Top-N and n-gram under Compressed + Data

| Top N/N-gram | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
|--------------|-------------------|------------|--------------|------------------------|
| 2/2          | 31                | 34         | 39           | 29                     |
| 2/3          | 33                | 45         | 42           | 29                     |
| 2/4          | 31                | 35         | 43           | 31                     |
| 3/2          | 30                | 35         | 41           | 29                     |
| 3/3          | 34                | 35         | 39           | 30                     |
| 3/4          | 31                | 37         | 39           | 32                     |
| 4/2          | 31                | 35         | 42           | 31                     |
| 4/3          | 35                | 35         | 43           | 29                     |
| 4/4          | 33                | 36         | 42           | 30                     |

Table 4: TAT-QA Accuracy (%) for different models, methods, Top-N and n-gram under Compressed + Data + Added Context

| Top N/N-gram | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
|--------------|-------------------|------------|--------------|------------------------|
| 2/2          | 38                | 36         | 42           | 27                     |
| 2/3          | 31                | 36         | 39           | 27                     |
| 2/4          | 29                | 35         | 41           | 28                     |
| 3/2          | 37                | 36         | 41           | 28                     |
| 3/3          | 33                | 37         | 41           | 27                     |
| 3/4          | 29                | 37         | 40           | 29                     |
| 4/2          | 37                | 35         | 41           | 30                     |
| 4/3          | 32                | 33         | 42           | 29                     |
| 4/4          | 30                | 33         | 41           | 29                     |

Table 5: FinQA Accuracy (%) for different models, methods, Top-N and n-gram for Baseline

| Top N/N-gram | Method                      | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
|--------------|-----------------------------|-------------------|------------|--------------|------------------------|
| 3/2          | Baseline                    | 66                | 54         | 60           | 54                     |
| 3/2          | Baseline + 3 random         | 68                | 58         | 48           | 52                     |
| 3/2          | Baseline + 3 representative | 70                | 60         | 60           | 56                     |

Table 6: FinQA Accuracy (%) for different models, methods, Top-N and n-gram under compressed setting

| Top N/N-gram | Method                        | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
|--------------|-------------------------------|-------------------|------------|--------------|------------------------|
| 3/2          | Compressed Prompt             | 76                | 62         | 52           | 44                     |
| 3/2          | Compressed + 3 random         | 76                | 56         | 46           | 46                     |
| 3/2          | Compressed + 3 representative | 76                | 60         | 64           | 52                     |

Table 7: FinQA Accuracy (%) for different models, methods, Top-N and n-gram for Compressed + Data

| Top N/N-gram | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
|--------------|-------------------|------------|--------------|------------------------|
|              |                   |            |              |                        |
| 2/2          | 74                | 54         | 56           | 44                     |
| 2/3          | 74                | 38         | 60           | 42                     |
| 2/4          | 70                | 36         | 62           | 42                     |
| 3/2          | 68                | 58         | 52           | 46                     |
| 3/3          | 74                | 56         | 60           | 42                     |
| 3/4          | 74                | 44         | 58           | 44                     |
| 4/2          | 72                | 48         | 58           | 44                     |
| 4/3          | 74                | 48         | 56           | 42                     |
| 4/4          | 78                | 54         | 58           | 48                     |
| 5/2          | 74                | 44         | 62           | 46                     |
| 5/3          | 70                | 48         | 58           | 44                     |
| 5/4          | 76                | 58         | 56           | 42                     |

Table 8: FinQA Accuracy (%) for different models, methods, Top-N and n-gram for Compressed + Data + Added Context

| Top N/N-gram | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
|--------------|-------------------|------------|--------------|------------------------|
|              |                   |            |              |                        |
| 2/2          | 72                | 42         | 56           | 46                     |
| 2/3          | 70                | 46         | 64           | 46                     |
| 2/4          | 70                | 44         | 58           | 42                     |
| 3/2          | 68                | 58         | 54           | 38                     |
| 3/3          | 74                | 46         | 58           | 42                     |
| 3/4          | 74                | 50         | 56           | 38                     |
| 4/2          | 64                | 44         | 54           | 44                     |
| 4/3          | 72                | 40         | 58           | 42                     |
| 4/4          | 66                | 54         | 54           | 40                     |
| 5/2          | 68                | 44         | 56           | 38                     |
| 5/3          | 70                | 22         | 60           | 38                     |
|              |                   |            |              | Continued on next page |

| Table 8 (continu | ed)               |            |              |                        |
|------------------|-------------------|------------|--------------|------------------------|
| Top N/N-gram     | Claude-3.5-Sonnet | GPT-4-Omni | GPT-4.1-Mini | Llama-3.3-70B-instruct |
| 5/4              | 66                | 60         | 58           | 42                     |