Integrating Domain Knowledge for Financial QA: A Multi-Retriever RAG Approach with LLMs

Stanford CS224N Custom Project

Yukun Zhang
Department of Computer Science
Stanford University
yukzhang@stanford.edu

Stefan Elbl Droguett
Graduate School of Business
Stanford University
selbl@stanford.edu

Samyak Jain
Graduate School of Business
Stanford University
samyakj@stanford.edu

Abstract

This project addresses the errors of financial numerical reasoning Question Answering (QA) tasks due to the lack of domain knowledge in finance. Despite recent advances in Large Language Models (LLMs), financial numerical questions remain challenging because they require specific domain knowledge in finance and complex multi-step numeric reasoning. We implement a multi-retriever Retrieval Augmented Generators (RAG) system to retrieve both external domain knowledge and internal question contexts, and utilize the latest LLM to tackle these tasks. Through comprehensive ablation experiments and error analysis, we find that domain-specific training with the SecBERT encoder significantly contributes to our best neural symbolic model surpassing the FinQA paper's top model, which serves as our baseline. This suggests the potential superior performance of domain-specific training. Furthermore, our best prompt-based LLM generator achieves the state-of-the-art (SOTA) performance with significant improvement (>7%), yet it is still below the human expert performance. This study highlights the trade-off between hallucinations loss and external knowledge gains in smaller models and few-shot examples. For larger models, the gains from external facts typically outweigh the hallucination loss. Finally, our findings confirm the enhanced numerical reasoning capabilities of the latest LLM, optimized for few-shot learning.

1 Key Information to include

- Mentor: Anna Goldie
- Acknowledgements: we would like to express our gratitude to Dr. Chris Manning for his advice and support to this project.
- Team Contributions: Yukun deployed and implemented the code for the multi-retriever system with two options for the generator and helped with the training process for fine-tuning. Stefan performed the fine-tuning and training for the system on various models and plotted the structure diagram for our system. Samyak conducted pre-processing and exploratory data analysis for the datasets, and helped with the prompt-based T5 models.
- External Collaborators (if you have any): None
- Sharing project: No

2 Introduction

Financial numerical reasoning QA tasks present a challenge for current LLMs as they require an understanding of dense financial terminology and complex computation to reach the correct answer. We attempt to mitigate this issue by leveraging additional external context information and the capabilities of LLMs. For example, a model may not understand financial terminologies such as 'options' and 'fair value,' which are not explained in the query contexts (see Figure 1). We propose that access to external financial dictionaries (like Investopedia) to look up financial definitions as non-parametric memory can mitigate this issue. Inspired by RAG [1], we aim to build a powerful multi-retriever RAG system with two options for the generator (neural symbolic and LLM) to better retrieve key facts from both internal query contexts and external terminology explanations to generate final answers with higher accuracy.

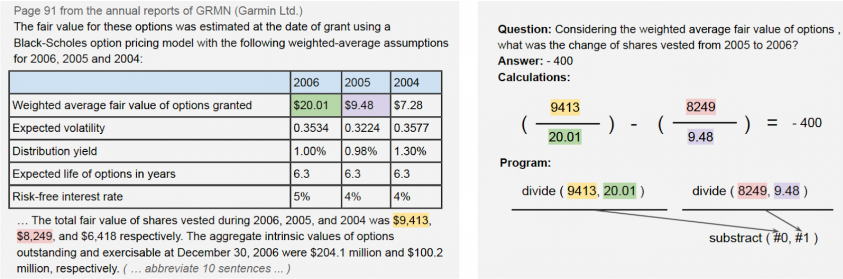

Figure 1: Sample question and answer from the FinQA dataset

3 Related Work

Existing models for financial numerical reasoning QA tasks still fall short of human expert performance. A recent survey [2] on finance LLMs finds that the current SOTA model for financial QA tasks is the zero-shot GPT-4 with 75% exact match accuracy, significantly below the 91% accuracy of human experts. According to the FinQA paper [3], over 85% of the errors stem from: (i) lack of domain knowledge in finance; or (ii) numerical reasoning errors. To address the latter, the ConvFinQA paper [4] adopted a method similar to Chain of Thought [5] on a financial QA dialogue dataset and showed a significant improvement for the outputs.

Inspired by the RAG model, which accesses external knowledge without additional training and outperforms SOTA results in many NLP tasks, we propose incorporating finance-specific non-parametric memory to help the model understand domain-specific concepts and improve performance. We follow the approach in the DPR (Dense Passage Retrieval) - FAISS paper [6] to design our external retriever, which combines DPR with efficient similarity search for superior performance.

4 Approach

Figure 2 depicts our model's architecture. Given that the query context can be long (over 2600 tokens) and contains irrelevant information, we focus on retrieving only the useful facts to improve system efficiency. The architecture includes two retrievers (internal and external) and a generator with two options for generating the final answers: a prompt-based LLM generator and a symbolic neural generator. We describe each part in detail below:

Internal Retriever: We fine-tune BERT-family models [7] to train a binary classifier for extracting the most relevant in-context supporting facts. We follow Hugging Face's steps to fine-tune pretrained models. Adopting the MathQAExample class from FinQA example code, we create question-context sentence pairs with appropriate padding and tokenization. Each sentence is labeled as positive or negative based on pre-labeled golden fact sentences. We use the Adam optimizer and CrossEntropyLoss for binary classification, and we add gradient clipping and a ReduceLROnPlateau learning rate scheduler to ensure consistent training loss reduction. For each query, we select the top 5 and top 3 sentences from the model outputs based on logit score rankings.

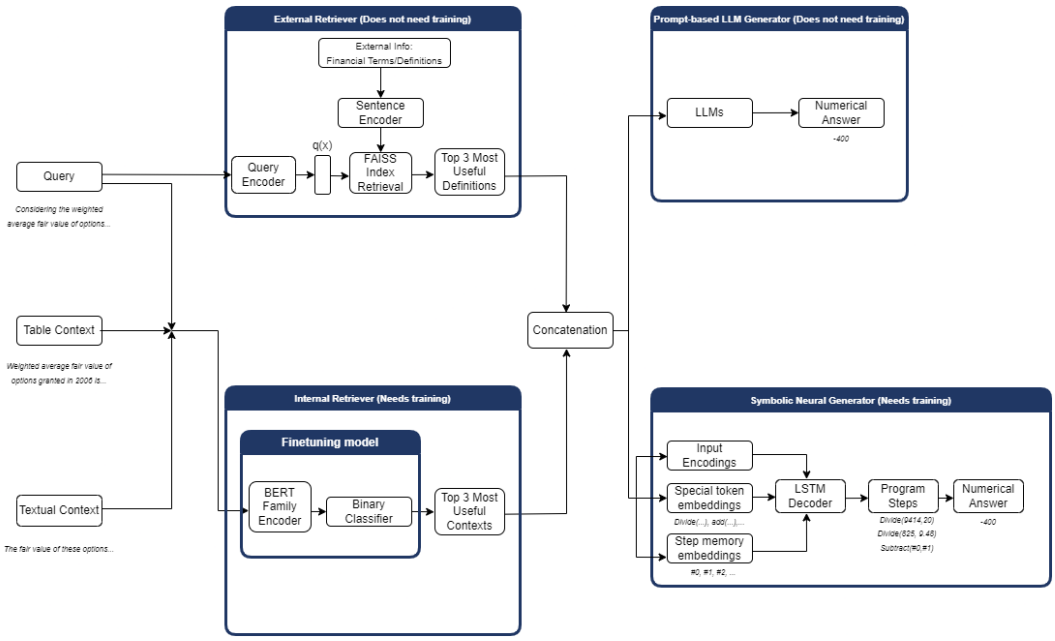

Figure 2: Model's architecture

External Retriever We use the DPR-FAISS structure similar to RAG to extract external supporting facts most relevant to the query from outsourced data without additional training. We begin by using an encoder to generate embeddings for each one-sentence definition summary in the financial terminology dictionary. Since FAISS is based on inner product similarity, we apply L2 normalization to the embeddings. We then use FAISS for fast retrieval, retrieving the top 3 related definitions for each query and storing them as external domain knowledge with associated similarity scores.

Prompt-based Generator: To fit in our system, the prompt-based LLM generator needs to be an encoder-decoder model. Using a paid API in Google Colab, we leverage Gemini Pro models [8] as our main LLM generator and feed the outputs from the retrievers as inputs to the LLM. Given an explicitly specified instruction prompt (see Appendix A for details) concatenated after the query, the model generates the final answer as a number or yes/no, whichever is applicable. Zero-shot and few-shot examples (see Appendix A for prompting details) are also tested on the dev set.

Symbolic Neural Generator: This generator requires training and consists of an encoder and decoder. We set up the same operation tokens (such as "add("), constant tokens (such as "const_100") and step memory tokens (such as #0, #1) defined in the FinQA paper to formulate the program steps. The operation tokens are the 10 most common types of math operations used to answer our queries; the constant tokens represent common constant numbers used in financial calculations and the step memory tokens denote the result from the n-th step (see Appendix B for details). We combine operation tokens and constant tokens, referred to as "special tokens", in the model diagram.

Each program step token can come from either the numbers in the retrieved contexts or from the three types of tokens described above. Our generator aims to generate each program step token one at a time based on its index (all numbers in the retrieved contexts and the three types of tokens are stored in the "program_ids" variable). For example, the single program step token "subtract(" has the index "5," and if the output of our model is "5," it will be converted to the string "subtract(".

To begin with, we randomly initialize the embeddings of these three token indexes as h_i^o, h_i^c, h_i^s. Then we use pretrained BERT family models to encode the retriever output as h_i^i. Our LSTM decoder receives the entire token embeddings as:

$$H = [h_i^i, h_i^o, h_i^c, h_i^s]$$

We use the decoder output h_T to calculate two attention vectors, att_i and att_h, for input sequence attention and decoder history attention, respectively. We define a context vector c_T to combine all the

contextual information from input sequences and the decoder history, where:

$$c_T = W_c[att_i; att_h; h_T]$$

A third attention vector att_r for input sequence is also added for the reasoning path of the program, so the entire reasoning results are:

$$H_T = W_h[H; H \circ att_r]$$

The final prediction output of the model is generated by choosing the top program index from the softmax results:

$$w_T = \text{softmax}(H_T \cdot c_T)$$

We use the Adam optimizer and CrossEntropyLoss for training. Additionally, we incorporate gradient clipping and a ReduceLROnPlateau learning rate scheduler to ensure the training loss consistently decreases over time. During inference, we use masks at each decoding step to ensure the structural correctness of the generated programs. The direct outputs of the program step tokens from the generator are concatenated to form the complete program step string. For instance, the golden generated program step for the example query in Figure 1 is the string "divide(9413, 20.01), divide(8249, 9.48), subtract(#0, #1)". We adopt the functions from the FinQA paper to evaluate program steps and compute the final numerical answers from the given program steps. The final answer is the number -400.

Baseline: using our new system, we re-implement the best-performing FinQANet structure from the original FinQA paper, training for 20 epochs. This single retriever-generator architecture employs a BERT-base-encoded retriever for internal context and a RoBERTa-large-encoded [9], LSTM-decoded [10] generator. The results are shown in the later experiment section.

5 Experiments

5.1 Data

For our internal QA data, we use the original FinQA dataset. Based on the earnings reports of S&P 500 companies, FinQA is an expert-annotated dataset containing 8,281 financial QA pairs, along with labeled numerical reasoning processes and highlighted supporting sentences. It is released as training (6,251), dev (883), and test (1,147) sets. Figure 1 shows an example question. The data stores query question texts, the textual contexts, and tabular contexts in JSON format. The average number of tokens per question is 687, and the maximum is 2,679. We improved the pre-processing methods from the FinQA analysis to transform each row of the table into long sentences based on various table formats. For example, in Figure 1, the first entry of the last table row is rewritten as "Risk-free interest rate of 2006 is 5%." This ensures that all inputs are in sentence textual formats.

For our external domain knowledge data, we use the FinRAD dataset [11], which includes explanations and definitions for over 13,000 financial terms and words. We pre-processed the data by replacing and removing unusual symbols and correcting typos. Although most definitions are one or two sentences long with an average of 46 tokens, we find many long definitions with examples and formulas, with a maximum token length of 667. To address this, we leveraged Gemini-1.0-pro to create one-sentence summaries for each term definition, and the generated outputs are very precise and consistent based on random human assessments.

5.2 Evaluation method

For the supervised internal retriever with fine-tuning, given that our labeled dataset is largely imbalanced (most sentences in the context are irrelevant to the query and are labeled as negative), we use recall to measure the performance for the top 3 and top 5 retrieved facts. For the unsupervised external retriever, we store the cosine similarity scores for the top 3 retrieved definitions.

We evaluate the performance of our generator by measuring execution accuracy (whether the answer is correct) and program accuracy (whether the steps taken by the model to solve the problem match those used by experts when labeling the data). These metrics introduce a trade-off: false positives affect execution accuracy (the model can guess the correct answer), while false negatives affect

program accuracy (if the steps taken are not entirely equivalent to the standard solution in the dataset). For the symbolic neural model, we directly calculate the numerical answers using the given program steps to measure execution accuracy. For the prompt-based LLM method, since we do not directly generate the program steps, we add a small ϵ to allow for some rounding errors in the final outcome. For example, if the model outputs "94.17%" and the golden answer is 0.942, we consider it correct.

5.3 Experimental details

We successfully implement a system with two retrievers (internal and external) and a generator with two options. Due to computational limitations, we could only train each for 20 epochs. Training time takes over 13 hours for each type of pretrained model in the internal retriever on one RTX 3080Ti GPU and over 4 hours for each type of pretrained model in the symbolic neural generator on one Colab A100 GPU. We set the learning rate to 2e-5, the batch size to 16, and the dropout rate to 0.1.

Internal Retriever: Using the annotated labels from the FinQA dataset, we fine-tune our classifier with five different pretrained models (BERT Base, SecBERT Base [12], FinBERT Base [13], RoBERTa Base and SpanBERT [14]).

External Retriever: Because we do not have labeled data, we utilize five different pretrained models (DPR, BERT Base, SecBERT Base, FinBERT Base and RoBERTa Base) to generate high-quality embeddings for similarity search. We use the FAISS index for fast retrieval to select the top 3 most relevant definitions for each question.

Prompt-based LLM Generator: we leverage the capabilities of current LLMs (Gemini-1.0-pro and Gemini-1.5-pro-latest) and conduct ablation experiments to assess the usefulness of the two retrievers. First, we run zero-shot and few-shot experiments by bundling the contextual texts, table texts, and query questions as inputs to the model. Then, we use the LLM as the single generator with only an internal retriever (BERT-base and SecBERT) and without any external knowledge. Finally, we test the whole system using the LLM as the single generator with both the internal retriever (BERT-base and SecBERT) and the DPR-FAISS external retriever. Additionally, without any fine-tuning, we test both the T5-base [15] and GPT-2 decoder [16] with BERT encoders as a single generator with no retrievers (zero shot), internal retriever only, and both internal and external retriever.

Neural Symbolic Generator: We examine six different pretrained models (BERT Base, BERT Large, SecBERT Base, FinBERT Base, RoBERTa Large, and RoBERTa Base) as the encoder and conduct ablation experiments to assess the effects of the two retrievers, with or without external knowledge. Moreover, we evaluate our model's performance across data subsets for various types of questions (single reasoning step vs. multiple reasoning steps; long context questions vs. short context questions; table-only questions vs. text-only vs. table-text mixed). We use our best-performing neural symbolic model and prompt-based LLMs for the performance breakdown.

5.4 Results

Fine-tuned Models	Top 3 Recall %	Top 5 Recall %
BERT Base	88.96	92.75
SecBERT Base	91.27	95.16
FinBERT Base	88.94	92.83
RoBERTa Base	87.76	92.67
SpanBERT	90.68	93.92

Table 1: Internal Retriever on Dev Set

Table 1 shows that for the internal retriever, SecBERT Base gives us much higher recall in both top 3 and top 5 retrieved facts compared to the BERT Base model used in our baseline. Since SecBERT is a BERT model entirely trained on 260,773 publicly available 10-K financial documents, it incorporates much more domain knowledge in finance. This domain-specific training can allow SecBERT to retrieve relevant documents more effectively and reduce the likelihood of missing important information.

Table 2 shows the similarity scores for our external retriever. Due to domain-specific training in the finance domain, we observe the highest similarity scores for the SecBERT encoder.

Sentence Encoders	Median Similarity Score	Mean Similarity Score
DPR	0.8583	0.8556
BERT Base	0.8538	0.8527
SecBERT Base	0.9287	0.9201
FinBERT Base	0.8759	0.8735
RoBERTa Base	0.8744	0.8721

Table 2: External Retriever on Dev Set for the Top 3 Facts

Architecture (20 Epochs of Training)	Program Accuracy %	Execution Accuracy %
BERT Base Internal Retriever + RoBERTa Large Encoder Generator (Baseline)	57.87	60.02
BERT Base Internal Retriever + DPR-FAISS External Retriever + RoBERTa Large Encoder Generator	58.49	60.96
SecBERT Internal Retriever + RoBERTa Large Encoder Generator	59.70	62.11
SecBERT Internal Retriever + DPR-FAISS External Retriever + RoBERTa Large Encoder Generator	**60.54**	**63.48**
SecBERT Internal Retriever + RoBERTa Base Encoder Generator	56.49	58.33
SecBERT Internal Retriever + DPR-FAISS External Retriever + RoBERTa Base Encoder Generator	56.75	58.81
SecBERT Internal Retriever + BERT Large Encoder Generator	53.95	55.76
SecBERT Internal Retriever + DPR-FAISS External Retriever + BERT Large Encoder Generator	54.64	56.97
SecBERT Internal Retriever + FinBERT Encoder Generator	49.53	52.08
SecBERT Internal Retriever + DPR-FAISS External Retriever + FinBERT Encoder Generator	48.77	50.90

Table 3: Selected neural-symbolic models' performance

Architecture	Execution Accuracy %
Gemini-1.0-pro Zero-shot	34.76
Gemini-1.5-pro-latest Zero-shot	36.27
SecBERT Internal Retriever + Gemini-1.0-pro Generator	39.30
SecBERT Internal Retriever + Gemini-1.5-pro-latest Generator	41.84
SecBERT Internal Retriever + Gemini-1.0-pro Generator + Few Shot Prompt	22.03
SecBERT Internal Retriever + Gemini-1.5-pro-latest Generator + Few Shot Prompt	66.02
SecBERT Internal Retriever + DPR-FAISS External Retriever + Gemini-1.0-pro Generator	41.75
SecBERT Internal Retriever + DPR-FAISS External Retriever + Gemini-1.5-pro-latest Generator	45.33
SecBERT Internal Retriever + DPR-FAISS External Retriever + Gemini-1.0-pro Generator + Few Shot Prompt	24.69
SecBERT Internal Retriever + DPR-FAISS External Retriever + Gemini-1.5-pro-latest Generator + Few Shot Prompt	**69.37**

Table 4: Selected prompt-based models' performance

Table 3 summarizes the results for neural symbolic models (see Appendix D for full results). We observe over a 3% performance improvement in both program accuracy and execution accuracy when we scale the generator encoder from RoBERTa Base to Large. This result, with extra model parameters, agrees with the scaling laws for Neural Language Models [17]. Moreover, increasing the pre-training data size seems to have a larger effect on performance than increasing model parameters. Both accuracies of the smaller RoBERTa Base model are over 2.5% higher than those of the BERT Large model. This result supports the empirical evidence from the scaling law paper and suggests that the quality and quantity of pre-training data can be more critical than model parameter size. The larger pre-training datasets from RoBERTa Base are likely to contain more relevant information for our financial QA task, which generalizes better with higher accuracy and robustness.

Table 4 summarizes the results for prompt-based LLMs (see Appendix D for full results). We observe a performance improvement of around 2% in execution accuracy when switching from Gemini-1.0-pro to the latest Gemini-1.5-pro-latest for zero-shot and single retriever only. This confirms that LLMs can perform some math calculations on their own without supervision. However, these results are worse than those from supervised neural symbolic methods. One possible reason is that without explicit supervision and structured symbolic reasoning, the LLM model might not have encountered a similar paradigm to our financial QA task setting during pre-training and does not understand the expected formats of the query and the answer in the specific finance domain. Therefore, it might not generalize well for our challenging financial numerical reasoning tasks without task-specific training.

6 Analysis

6.1 Quality of the external facts

Even though SecBERT achieves higher average similarity scores, the retrieved external facts are less relevant to the query compared to DPR encoding. For example, when asked about the average payment volume per transaction for American Express, the SecBERT encoder achieves a 0.9359 similarity score and retrieves the definition for "zero plus tick" (a security sale term), which is entirely irrelevant to the original query. In contrast, the DPR encoder, with a 0.8413 similarity score, correctly retrieves the definition for American Express (as a financial institution). One possible explanation is that DPR is specifically trained for passage retrieval and fine-tuned with a contrastive loss for question answering tasks, optimizing it for higher relevance between financial queries and term definitions. Additionally, we randomly select 50 examples from the dev set and conduct human evaluations on the quality of retrieved external facts using SecBERT and DPR encodings. The evaluations confirm the better retrieval quality of the DPR encoding, which we choose as our optimal encoding to incorporate external knowledge for our downstream generator.

6.2 Ablation experiments for the external retriever

In neural symbolic models, our best model (SecBERT Internal Retriever + DPR-FAISS External Retriever + RoBERTa Large Encoder Generator) outperforms the baseline by 3.46% in execution accuracy and 2.67% in program accuracy, with most of the gains (around 2%) seem to come from the stronger internal retriever SecBERT. The performance improves with better internally retrieved facts.

We generally observe slight (<1.5%) performance improvements when adding the DPR-FAISS external retrieved facts, especially with larger encoders such as RoBERTa Large and BERT Large. The added external facts help the model better locate the correct position to extract information from the internal contexts and reinforce the key terms in the query. For example, for the dev set query "What was the change in millions of operating income from 2016 to 2017?", without external facts, our best model only extracts the operating income in 2016 and subtracts its closest number. After adding external facts about operating incomes (how to calculate operating income), the model correctly subtracts the two operating income numbers.

However, with smaller models for the generator encoders such as RoBERTa Base, the performance remains roughly the same after including the external facts, and it even drops around 1% for FinBERT Base with external facts. Adding external information causes about 1.4% of the training data inputs (87/6251) to exceed the 512 max token length, resulting in truncation when fed into the model. These incomplete sentences may become irrelevant and increase hallucinations for the generator, especially

for smaller models. Consequently, even with added external facts, the model might still be distracted by these incomplete sentences and generate incorrect outputs.

Therefore, we observe a trade-off between domain knowledge and hallucinations. Generally, for larger models, the gains from external facts outweigh the hallucination loss. Our ablation experiments on LLMs also confirm this hypothesis, showing solid improvements of over 2% when adding external facts. However, for smaller models the hallucination loss starts to increase and may even offset the external knowledge gain. This interesting finding highlights the complex dynamics between model size, external knowledge integration, and hallucination effects for future study.

6.3 Ablation experiments for the few-shot prompts in LLM

When we add few-shot prompts to Gemini-1.0-pro, we surprisingly see a large performance drop from 39.30% to 22.03%. Analyzing the generated outputs, it seems the LLM sometimes uses numbers from the contexts of few-shot examples to answer the new query. For example, fo the question "what percentage of total purchase commitments are due after 2014?", Gemini-1.0-pro generates the incorrect output 49.49% by directly using the numerical answer 0.4949 from the first example in few-shot prompts, which is entirely unrelated to the target query.

Moreover, the LLM also tends to ignore the numbers in the given query contexts and can disregard the actual context. For example, even with an extremely short query context such as "2012: 720; thereafter: 4717; total debt: $7680; what percentage of total debt is due after 2012?", Gemini-1.0-pro still provides the incorrect answer 84.37%, whereas the correct answer is 61.42% (4717/7680). Gemini-1.5-pro-latest also gives the wrong answer 81.76%. Without additional instructions, the LLM may prefer to use the knowledge acquired during pre-training rather than the actual context for QA.

However, after applying the same few-shot prompts to Gemini-1.5-pro-latest, we find it can answer the above question perfectly. By adding the few-shot prompts, the execution accuracy greatly improves to 66.02%, and we achieve our SOTA model by adding external facts with 69.37%. These results suggest that Gemini-1.5-pro-latest may have significantly improved capabilities in numerical reasoning and arithmetic operations, especially with better optimized few-shot learning capabilities. One possible explanation is that Gemini-1.5-pro-latest includes advancements that specifically address the issue of hallucination, where the model generates information not grounded in the input context. This helps the model rely more on the provided context and less on unrelated pre-training knowledge.

6.4 Performance breakdown across sub-datasets

Sub Dataset	**Best Neural Symbolic Execution Accuracy %**	**Best Prompt-based LLM Execution Accuracy %**
Full Test Set	61.28	68.39
Table-only Questions	68.55	71.87
Text-only Questions	56.13	67.41
Table-Text-mixed Questions	44.26	64.72
Long Context Questions	59.35	67.24
Short Context Questions	62.11	68.70
Single Reasoning Step Questions	69.97	75.48
Multi Reasoning Step Questions	46.60	58.62

Table 5: Performance of models on different types of questions sub-datasets

Table 5 shows the performance breakdown of our best models across sub test sets for various types of questions. With only 20 epochs of training, our best neural symbolic model even slightly outperforms the best model from the original FinQA paper (61.24%) that had 300 epochs of training in terms of execution accuracy. Our best LLM model achieves the SOTA with 68.39% execution accuracy. We observe that both selected models perform better on table-only questions.

Both models perform slightly worse on originally long context questions with more irrelevant content. This robustness could further confirm the effectiveness of using internal retrievers to filter out unimportant sentences, ensuring all model input texts are short enough regardless of the original context length. It seems that originally long questions are more challenging than short ones.

Most questions require single-step reasoning, and both models perform much better on these easier questions. Not surprisingly, they both perform much worse on multi-step reasoning questions compared to their accuracy on the full set. Overall, the best LLM model achieves better results on all subsets compared to the best neural symbolic model. This further confirms the strong numerical reasoning ability of Gemini-1.5-pro-latest in domain-specific QA tasks.

6.5 Error Analysis

We randomly select 30 error cases from our best model results. Our analysis indicates that over 65% of the errors are due to poor numerical reasoning for numerical unit conversions and incorrect number retrieval, especially for multi-step questions. For example, a question asks for the net sales in billions, but our model predicts 7,222.22 (the correct answer is 7.22). See Appendix E for more details.

7 Conclusion & Future Work

This project implements a multi-retriever RAG system and leverages the latest LLM to address numerical reasoning QA in the finance domain. We conduct comprehensive ablation experiments and error analysis across various models. Largely due to the domain-specific training of the SecBERT encoder, the best neural symbolic model outperforms our baseline from the best models in the FinQA paper, and the best prompt-based LLM generator achieves SOTA with significant improvements (>7%). The results demonstrate the trade-off effects of hallucinations for added external knowledge on smaller models and extra few-shot examples. Our analysis further confirms the improved numerical reasoning ability of the latest LLM, which is better optimized for few-shot learning.

Computing power was a significant limitation for our project. Despite training each model for only 20 epochs, compared to over 300 epochs by the FinQA authors, our best symbolic neural model still outperforms their best model. Figure 3 shows that the loss continues to decrease after 20 epochs, suggesting that additional future training could further improve performance.

Another limitation we identified was the length of the context we could retrieve. Some added contexts are truncated due to max token length limits, which could reinforce the hallucination effects. In the future, we could annotate some data and fine-tune an NER system to identify key financial terms for each query for better and shorter external knowledge retrieval.

8 Ethics Statement

An ethical challenge in our project is the potential presence of biases in the dataset. Over 60% of the questions require information only from in the question's table to reach the answer. This indicates a bias towards certain types of questions, limiting the model's external validity and potentially leading information misuse. Fully understanding a company's financial health and performance requires multiple back-and-forth references between the pages of a financial statement and footnote reviews to understand the assumptions that contextualize a number. For example, the classification of a security as either available for sale or held to maturity can inflate net income. A question such as "ETR/2016/page_396.pdf" in our dataset, which asks about the increase in net income over a period of time, only considers one source of net income. The answer that net income has increased does not provide a full picture of that metric and can overlook these types of accounting tricks. A way to mitigate this issue in the model's current state is to add a disclaimer about how the answers to the questions in the data should only be used in context. Future work can benefit by expanding the dataset to address the dependency of questions on multiple pages and footnotes and introducing models with longer context windows to accommodate that information within the model's retrieved facts.

A societal issue is the potential displacement of jobs if our model achieves performance beyond that of an expert. In such a case, people may stop consulting financial experts and instead rely on our model. We posit that our model is meant to augment human decisions, not replace human experts. Even if a model were to have significantly higher performance than our current baseline, a potential solution for this problem is the relocation, not displacement, of jobs. Experts could shift into a consultant role where they verify the model's output to prevent people from blindly following financial advice.

References

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, and et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, page 9459–9474, 2020.

[2] Jean Lee, Nicholas Stevens, Soyeon Caren Han, and Minseok Song. A survey of large language models in finance (finllms). *arXiv preprint arXiv:2402.02315*, 2024.

[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. FinQA: A dataset of numerical reasoning over financial data. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[4] Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6279–6292, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[5] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, and et al. Chain-of-thought prompting elicits reasoning in large language models. In *arXiv preprint arXiv:2201.11903v6*, 2022.

[6] 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 Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[8] Machel Reid, Nikolay Savinov, Denis Teplyashin, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. In *Proceedings of the Conference on Artificial Intelligence*. arXiv, February 2024.

[9] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2019.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Proceedings of the 9th International Conference on Neural Information Processing Systems (NIPS)*, pages 173–180. MIT Press, 1997.

[11] Sohom Ghosh, Shovon Sengupta, Sudip Kumar Naskar, and Sunny Kumar Singh. Finrad: Financial readability assessment dataset - 13,000+ definitions of financial terms for measuring readability. In *Proceedings of the The 4th Financial Narrative Processing Workshop @LREC2022*, pages 1–9, Marseille, France, June 2022. European Language Resources Association.

[12] Lefteris Loukas, Manos Fergadiotis, Ilias Chalkidis, Eirini Spyropoulou, Prodromos Malakasiotis, Ion Androutsopoulos, and Paliouras George. Finer: Financial numeric entity recognition for xbrl tagging. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)*. Association for Computational Linguistics, 2022.

[13] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models, 2019.

[14] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77, 2020.

[15] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 2020.

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. Technical report.

[17] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. volume abs/2001.08361, 2020.

9 Appendices

A: Prompts Used for the LLM-based Approach:

The instruction prompt at the end of the input: "given the contexts, your generated response to the above question must only be a single numerical number only (without any symbols nor texts), or a numerical number with a percentage sign only, or just yes/no, whichever applicable."

The first few-shot example prompt at the beginning of the input: "as of and for the years ended december 31 , the operating income of 2003 is 1039 ; the operating income of 2002 (1) is 695 ; the operating income of 2001 (1) is 1717 ; what was the percentage change in operating income for entities in which the company has the ability to exercise significant influence but does not control and that are accounted for using the equity method between 2002 and 2003? given the contexts, your generated response to the above question must only be a single numerical number only (without any symbols nor texts), or a numerical number with a percentage sign only, or just yes/no, whichever applicable. 0.4949."

The second few-shot example prompt at the beginning of the input: "in millions except for per share data the diluted-as reported of 2005 is $ 4.55 ; in millions except for per share data the diluted-pro forma of 2005 is 4.52 ; was diluted-as reported net income per share greater than diluted-pro forma net income per share? given the contexts, your generated response to the above question must only be a single numerical number only (without any symbols nor texts), or a numerical number with a percentage sign only, or just yes/no, whichever applicable. Yes."

B: Definitions for Special Tokens

Operation tokens: 'EOF', 'UNK', 'GO', ')', 'add(', 'subtract(', 'multiply(', 'divide(', 'exp(', 'greater(', 'table_sum(', 'table_average(', 'table_max(', 'table_min('.

Step memory tokens: #0, #1, #2, #3, #4, #5, #6, #7, #8, #9, #10.

Constant tokens: CONST_2, CONST_1, CONST_3, CONST_4, CONST_5, CONST_6, CONST_7, CONST_8, CONST_9, CONST_10, CONST_100, CONST_1000, CONST_10000, CONST_100000, CONST_1000000, CONST_10000000, CONST_1000000000, CONST_M1.

C: Training Loss Trend

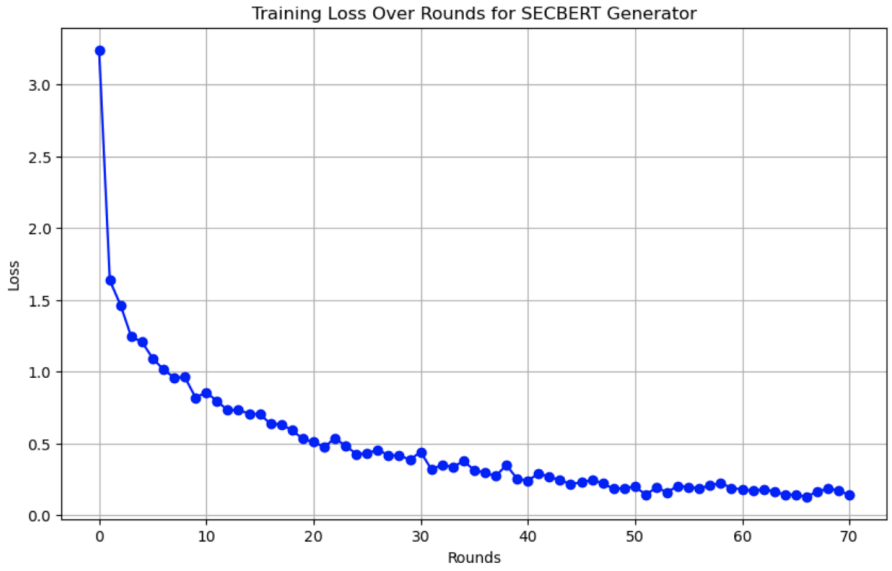

Figure 3: Training Loss Over Rounds (20 epochs) for the Generator with SEC-BERT encoder and LSTM decoder.

D: Full Results Table

We split the table with all of our full results into two so it can fit into one page:

Architecture (20 epochs of training)	Dev Program Accuracy %	Dev Execution Accuracy %
BERT Base Internal Retriever ONLY + RoBERTa Large Encoder Generator (Baseline)	57.87	60.02
BERT Base Internal Retriever + DPR-FAISS External Retriever + RoBERTa Large Encoder Generator	58.49	60.96
SecBERT Internal Retriever ONLY + RoBERTa Large Encoder Generator	59.70	62.11
SecBERT Internal Retriever + DPR-FAISS External Retriever + RoBERTa Large Encoder Generator	60.54	63.48
SecBERT Internal Retriever ONLY + BERT Large Encoder Generator	53.95	55.76
SecBERT Internal Retriever + DPR-FAISS External Retriever + BERT Large Encoder Generator	54.64	56.97
SecBERT Internal Retriever ONLY + SecBERT Encoder Generator	51.76	54.36
SecBERT Internal Retriever + DPR-FAISS External Retriever + SecBERT Encoder Generator	51.84	54.52
SecBERT Internal Retriever ONLY + BERT Base Encoder Generator	49.78	52.61
SecBERT Internal Retriever + DPR-FAISS External Retriever + BERT Base Encoder Generator	49.69	52.63
SecBERT Internal Retriever ONLY + FinBERT Encoder Generator	49.53	52.08
SecBERT Internal Retriever + DPR-FAISS External Retriever + FinBERT Encoder Generator	48.77	50.90
SecBERT Internal Retriever ONLY + RoBERTa Base Encoder Generator	56.49	58.33
SecBERT Internal Retriever + DPR-FAISS External Retriever + RoBERTa Base Encoder Generator	56.75	58.81
BERT Base Internal Retriever ONLY + FinBERTQA Encoder Generator	40.53	44.22

Table 6: Results table (Part 1)

Architecture (20 epochs of training)	Dev Program Accuracy %	Dev Execution Accuracy %
SpanBERT Internal Retriever ONLY + FinBERT Encoder Generator	45.23	44.12
SpanBERT Internal Retriever ONLY + SecBERT Encoder Generator	44.03	42.45
SpanBERT Internal Retriever ONLY + BERT Base Encoder Generator	42.11	43.68
Gemini-1.0-pro Zero-shot	N/A	34.76
Gemini-1.5-pro-latest Zero-shot	N/A	36.27
BERT Base Internal Retriever ONLY + Gemini-1.0-pro Generator	N/A	35.56
BERT Base Internal Retriever + DPR-FAISS External Retriever + Gemini-1.0-pro Generator	N/A	36.35
BERT Base Internal Retriever + DPR-FAISS External Retriever + Gemini-1.0-pro Generator + Few shot Prompt	N/A	21.32
SecBERT Internal Retriever ONLY + Gemini-1.0-pro Generator	N/A	39.30
SecBERT Internal Retriever ONLY + Gemini-1.5-pro-latest Generator	N/A	41.84
SecBERT Internal Retriever ONLY + Gemini-1.0-pro Generator + Few Shot Prompt	N/A	22.03
SecBERT Internal Retriever ONLY + Gemini-1.5-pro-latest Generator + Few Shot Prompt	N/A	66.02
SecBERT Internal Retriever + DPR-FAISS External Retriever + Gemini-1.0-pro Generator	N/A	41.75
SecBERT Internal Retriever + DPR-FAISS External Retriever + Gemini-1.5-pro-latest Generator	N/A	45.33
SecBERT Internal Retriever + DPR-FAISS External Retriever + Gemini-1.0-pro Generator + Few Shot Prompt	N/A	24.69
SecBERT Internal Retriever + DPR-FAISS External Retriever + Gemini-1.5-pro-latest Generator + Few Shot Prompt	N/A	69.37
SecBERT Internal Retriever + DPR-FAISS External Retriever + BERT encoder + gpt2 decoder	N/A	0
T5-Base Generator (zero shot)	N/A	2.83
BERT Base Internal Retriever + T5-Base Generator (zero shot)	N/A	2.15
BERT Base Internal Retriever + DPR-FAISS External Retriever + T5-Base Generator	N/A	2.15

Table 7: Results table (Part 2)

E: Error Cases

Case One: numerical unit conversions. The question asks "what is the applied 2019s net sales in 2018, (in billions)?". The correct reasoning steps are: divide(1.3, divide(18, 100)) = 7.22, but the model answers 7,222.22 without properly identifying the requirements for the question and converting the number in billions.

Case Two: umber retrieval. The question asks "what is the average amortization amount, in millions, from 2015-2019?". The correct reasoning steps are: "divide(add(multiply(45, 4), 44), 5)" = 44.8, but the model incorrectly retrieves the 36 million amortization expense ending in December 31 , 2014 instead of 45 million for 2015.