Financial QA System: Approach, Metrics and Findings

Summary of the findings

This report presents a Large Language Model (LLM) driven prototype that answers questions based on financial documents, including both text and tabular data. The system was developed to demonstrate the capabilities of modern LLMs in understanding and reasoning about complex financial information.

The system was evaluated on the ConvFinQA `train.json` dataset, achieving an overall accuracy of 47.61% with varying performance across different question types and difficulty levels. While this accuracy seems low, it is an aggregation of accuracies over multiple question types so, it might look like it is performing worse than a random guess (50% accuracy), but it is not. Later, you will see about 50% of the error crept in because of the missing "%" symbol. The evaluation reveals both strengths and opportunities for improvement, particularly in handling percentage calculations and formatting.

Approach

System Architecture and Design Choices

I designed a modular system architecture with clear separation of concerns:

1. **Document Retrieval**: The system identifies the most relevant document for a given question using keyword extraction and document scoring
2. **Question Analysis**: The LLM analyzes the question to determine the required information and calculations
3. **Information Extraction**: Relevant data is identified from both textual and tabular content
4. **Reasoning & Calculation**: The system performs necessary calculations to derive the answer
5. **Answer Formulation**: The final answer is generated with proper formatting

This part handles the Q&A section of the solution. There is another evaluate section that evaluate the model's performance on the samples from `train.json` data. I ran a complete evaluation on all 2109 testable samples from the data, the results of which are presented later in this report.

Document Retrieval Design Choice

For document retrieval, I considered several approaches:

1. **Simple Keyword Matching (Implemented)**: Extracting keywords from the question and scoring documents based on keyword presence. This approach is computationally efficient and works well for straightforward queries.
2. **Enhanced LLM-based Keyword Extraction (Implemented)**: Using an LLM to identify the most semantically important terms from questions, improving relevance scoring.
3. **Dense Vector Embedding (Alternative)**: Embedding both questions and documents using models like BERT and calculating similarity scores. While this would potentially capture deeper semantic relationships, it would add significant complexity and computational cost without necessarily improving performance on our specific dataset, which has relatively simple retrieval needs.

I chose the hybrid approach (basic + LLM-enhanced keyword extraction) as it balances computational efficiency with retrieval accuracy. The choice was validated by the system's performance on factual questions, which rely heavily on effective document retrieval.

Reasoning Approach Design Choices

For the reasoning component, I explored several options:

1. **End-to-end LLM (Implemented)**: Having the LLM handle the entire reasoning process, from data extraction to calculation to answer generation. This leverages the LLM's built-in reasoning capabilities.

2. **Structured Parsing/Outputs + Programmatic Calculation (Alternative)**: Using the LLM to extract relevant values, then performing calculations programmatically. While this would provide more precise calculations, it would significantly increase complexity for the wide variety of calculations needed.
3. **Multi-stage LLM Chain (Alternative)**: Breaking the reasoning into discrete steps with separate LLM calls for extraction, calculation planning, and execution. This would potentially improve accuracy but at the cost of increased latency and token usage.

I chose the end-to-end LLM approach because:

- It minimizes latency by requiring only a single API call
- It leverages the LLM's ability to handle diverse question types without specialized logic
- It provides a natural "chain of thought" in its reasoning

To mitigate calculation accuracy concerns, I implemented post-processing to normalize answers and detect calculation errors, allowing us to identify when the LLM is likely to have made a numerical error. I agree that this is quite a rigid architecture and a lot of things can be solved within the LLM realm without the need for rigid/harcoded code patterns. This could be an improvement for the next version.

Answer Extraction and Normalization

For answer extraction and normalization, I implemented:

1. **Explicit Answer Marking (Implemented)**: The prompt instructs the LLM to prefix the final answer with "Final Answer:", making extraction more reliable.
2. **LLM-based Answer Extraction (Implemented)**: For cases where formatting wasn't consistent, a second LLM call extracts the answer.
3. **Rules-based Normalization (Implemented)**: A comprehensive set of normalization rules for different answer formats, particularly for percentage values.
4. **Approximate Matching (Implemented)**: For numeric answers, allowing for small differences in values using relative tolerance.

An alternative approach would have been end-to-end prompting without any post-processing, relying solely on the LLM to format answers correctly. I rejected this approach due to the inconsistency in LLM output formatting, especially for numerical values. Again, this approach seems to be more rigid or "brute-force" and implementing structured outputs from Langchain or a complete end-to-end LLM pipeline in which an agent can make sense from the raw response provided by another agent.

Technical Implementation

The implementation leverages the following technologies:

- **OpenAI API (GPT-4o)**: For natural language understanding and financial reasoning
- **Python**: For the core implementation and data processing
- **Prompt Engineering**: Carefully crafted prompts that instruct the LLM to:
  - Analyze questions systematically
  - Extract relevant data from text and tables
  - Show step-by-step reasoning
  - Format answers consistently

Alternative Models Considered

While GPT-4o was selected as the primary model, I also considered:

1. **Open-source Models (Alternative)**: Models like Llama 3 or Mistral could reduce operational costs but would likely sacrifice accuracy on complex financial reasoning.
2. **Specialized Financial Models (Alternative)**: Fine-tuned models specifically for financial tasks could improve performance but would require significant additional development time.

3. **Claude or Gemini (Alternative)**: Other commercial LLMs with similar capabilities to GPT-4, which could serve as alternatives.

The choice of GPT-4o was based on its strong performance on complex reasoning tasks and ability to handle both textual and tabular data effectively.

Evaluation Methodology

To thoroughly assess the system's performance, I developed a comprehensive evaluation framework with multiple metrics:

1. **Basic Metrics**:
    - **Accuracy**: Percentage of questions answered correctly
    - **Exact Match Rate**: Percentage of answers matching exactly
2. **Advanced Metrics**:
    - **MAPE (Mean Absolute Percentage Error)**: Measuring error magnitude for numerical predictions
    - **Binned Accuracy**: Performance broken down by question difficulty
    - **Response Time Analysis**: Processing efficiency metrics
3. **Error Analysis**:
    - **Error Type Distribution**: Categorization of errors by type
    - **Question Type Performance**: Accuracy for different question categories

Metrics Selection Rationale

The selection of these metrics was deliberate:

1. **Accuracy vs. Exact Match**: I chose to distinguish between exact matches and close matches (semantically correct but formatted differently) because financial information can be correctly presented in multiple formats (e.g., "14%" vs "14.0%").
2. **MAPE**: While F1-score is common in NLP evaluation, MAPE is more appropriate for financial calculations where the magnitude of error matters. However, I recognized its limitations with outliers.
3. **Binned Accuracy**: Breaking down performance by question difficulty provides insights into the system's strengths and weaknesses on different types of reasoning.
4. **Error Categorization**: I implemented detailed error type analysis rather than simple wrong/right classification to identify specific improvement areas.

Evaluation Results

Overall Performance

- **Overall accuracy**: 47.61%
- **Exact match rate**: 32.86%
- **Close matches**: 14.75%
- **Incorrect answers**: 52.39%
- **Mean Absolute Percentage Error (MAPE)**: 1,216,462.84%

Note: The extremely high MAPE is likely influenced by outliers with very large calculation errors.

Performance by Question Type

Question Type	Accuracy	Sample Size
Explanation	100.00%	2
Factual	56.99%	472
Other	46.05%	76
Percentage	45.59%	1145

Question Type	Accuracy	Sample Size
Change	43.51%	370
Quantity	34.09%	44

The system performs best on factual questions, which typically require straightforward information retrieval without complex calculations. Percentage and change calculations show moderate performance, while quantity questions appear more challenging.

Performance by Question Difficulty

Difficulty	Accuracy	Sample Size
Moderate	59.56%	319
Complex	46.06%	1661
Simple	37.98%	129

Interestingly, the system performs better on moderate difficulty questions than on simple ones. This could indicate that the system is well-suited for questions requiring some reasoning but might be overthinking simpler questions or encountering formatting issues with them.

Error Analysis

The most common error types were:

1. **Missing percentage symbol**: 545 occurrences (49.3% of errors)
2. **Unknown error type**: 92 occurrences (8.3% of errors)
3. **Various calculation errors**: Ranging from minor to major

The predominance of "missing percentage symbol" errors suggests a formatting issue that could be relatively easy to fix.

Key Findings

Strengths

1. **Reasoning Ability**: The system demonstrates strong reasoning on moderate complexity questions, showing the LLM's capability to understand financial concepts.
2. **Factual Question Handling**: At 56.99% accuracy, the system shows good performance on factual questions, effectively extracting information from documents.
3. **Close Match Performance**: The inclusion of "close match" criteria improves overall accuracy by 14.75%, capturing answers that are semantically correct but formatted differently.

Shortcomings

1. **Formatting Issues**: The predominance of "missing percentage symbol" errors (49.3% of all errors) indicates inconsistent formatting of answers, particularly for percentages.
2. **Calculation Accuracy**: Various calculation errors suggest the system sometimes struggles with numerical reasoning or data extraction from tables.
3. **Simple Question Performance**: The relatively low performance on simple questions (37.98%) compared to moderate ones (59.56%) suggests potential overthinking or format issues.
4. **MAPE Concerns**: The extremely high MAPE indicates some severe calculation errors that skew this metric, making it less reliable for this evaluation.

Analysis of Design Trade-offs

The evaluation results reveal several insights about the design choices:

1. **End-to-End LLM Approach**: The good performance on moderate questions validates the end-to-end LLM approach for complex reasoning. However, the calculation errors suggest that hybrid approaches with programmatic calculation might improve accuracy for specific question types.
2. **Document Retrieval**: The strong performance on factual questions indicates that the keyword-based retrieval approach is effective for this dataset. The more complex embedding-based approaches might not have yielded significant improvements.
3. **Answer Normalization**: The high number of formatting errors, particularly missing percentage symbols, indicates that the current normalization approach needs refinement. A more aggressive post-processing approach specifically targeting percentage formatting would likely yield substantial improvements.
4. **Evaluation Metrics**: While MAPE was extremely high due to outliers, the binned accuracy and error type analysis provided valuable insights that wouldn't have been captured by simple accuracy metrics.

Future Improvements

Based on the evaluation results, I propose the following improvements:

1. **Answer Normalization**: Enhance post-processing to ensure consistent formatting, especially for percentage values. Implementing a more robust percentage detection and correction system could address the most common error type.

2. **Calculation Validation**: Implement validation checks for numerical calculations, particularly for percentage changes. For critical calculations, a hybrid approach could be used where the LLM extracts values but calculations are performed programmatically.

3. **Question Type Handling**: Develop specialized prompts for different question types, particularly for quantity questions where performance is lowest. This could include specific instructions for formatting quantity answers.

4. **LLM Chain of Thought**: Further optimize prompts to encourage more explicit step-by-step reasoning for complex calculations.

5. **Outlier Detection**: Implement detection of unreasonable calculations to prevent extreme errors that affect metrics like MAPE.

6. **Alternative Approaches to Explore:**

    - **Fine-tuning**: Explore fine-tuning a model specifically for financial QA tasks to improve performance across all question types.
    - **Retrieval-Augmented Generation (RAG)**: Implement a more sophisticated RAG approach that could improve performance on complex documents.
    - **Multi-step Reasoning**: For complex calculations, break down the reasoning process into explicit steps with verification at each stage.
    - **Table Parsing**: Implement specialized table parsing to extract structured data more reliably before calculation.

Conclusion

The financial QA system demonstrates promising capabilities in understanding and reasoning about financial documents. With an overall accuracy of 47.61%, it shows the potential of LLM-based systems for financial question answering.

The design choices made in this implementation—modular architecture, end-to-end LLM reasoning, keyword-based retrieval, and comprehensive evaluation metrics—have proven effective for building a functional financial QA system while revealing specific areas for improvement.

The comprehensive evaluation framework provides valuable insights into the system's performance across different dimensions, revealing both strengths and specific areas for improvement. The most pressing issue—formatting inconsistencies—appears relatively straightforward to address and could significantly improve performance.

With the proposed improvements, particularly in answer normalization and calculation validation, the system could reach substantially higher accuracy levels while maintaining its strong reasoning capabilities.

This project demonstrates both the current capabilities and limitations of LLM-based financial question answering systems, providing a foundation for further research and development in this domain.