# 1. Introduction

Overview of the Task

The task involves developing a prototype system that can answer questions based on financial documents using an LLM (large language model). The primary goal is to demonstrate the ability to extract accurate information from complex financial documents and respond to specific financial queries, which may involve interpreting text, tables, and figures.

An important aspect of this task is also to create a reliable self-evaluating scoring system in order to compare the ground truth values in the dataset with the model’s responses. The final ground truths ostensibly were reached following a turn based Q&A of a user and chatbot.

Dataset Overview

The dataset provided for this project is ***train.json*** from the **ConvFinQA** dataset. ConvFinQA is designed for conversational question answering (QA) in a financial context, featuring realistic financial documents and complex, quantitative questions. Each question-answer pair is crafted to test the system’s ability to interpret data, perform necessary calculations, and provide accurate responses. For instance, a sample question might be, "What was the percentage change in net cash from operating activities from 2008 to 2009?" where the system is expected to return a precise numerical answer, such as "14.1%."

The dataset contains Type I simple conversations (derived from one question) and Type II complex conversations (derived or two). As these are conversations, there is also granular information on each question and answer in the interaction leading up to the final “ground truth” answer.

# 2. Problem Definition and Objectives

2.1. Problem Statement

To replace the iterative nature of conversational communication with a ***single point of contact*** for the user: namely, it attempts to use LLM agents to replicate this chain of thought and arrive at a final response from a single user interaction.

2.2. Objective

1. **Data Processing**: Ensuring the data is parsed correctly for model input, particularly extracting relevant numbers and performing necessary operations for quantitative questions.
2. **Model Selection and Training**: Choosing an appropriate model architecture capable of handling both the text-based context and the specific quantitative requirements of the financial QA.
3. **Evaluation and Metrics**: Measuring the accuracy and effectiveness of the model using a suitable scoring function and metrics. The system should consider numerical accuracy, conceptual correctness and contextual relevance in order to arrive at a final score.
4. **User Interface:** providing a platform to test the LLM with, as well as additional functionality for uploaded documents.

# 3. Model Architecture and Approach

3.1. Overview  
The system is a complex state machine designed to handle mathematical reasoning and financial question-answering tasks. It leverages LangChain’s AzureChatOpenAI and LangGraph to implement a multi-stage reasoning process for accurate financial question answering. The process includes both basic and advanced scoring mechanisms to assess the model’s accuracy at multiple levels.

3.2. Tool and LLM Setup  
The system integrates a custom mathematical **calculator** function using the numexpr library for efficient expression evaluation, which is beneficial for handling arithmetic operations directly. This calculator function is added to the LLM toolset, allowing the model to perform specific calculations when required.

The **LLM instance**, AzureChatOpenAI, serves as the core language model to interpret and generate responses. The LLM's temperature is set to zero to prioritize accuracy and minimize randomness, essential for consistent financial question answering.

3.3. ChainState Structure & Flow  
The ChainState class defines the structure of the state across stages, with key fields such as messages, ground\_truth, score\_reasoning, detailed\_score, and response. This structured state management allows the system to maintain relevant information, such as user questions, LLM responses, and scoring data, throughout the answer generation and evaluation process.

3.4. Processing and Scoring Workflow

The workflow follows a defined path through the state graph, involving:

* **Tool Execution**: Calls call\_tool and execute\_tool nodes to generate an initial response.
* **Model Call**: The model is called directly if no tool is required.
* **Response Cleaning**: The clean\_response function formats the LLM output into a concise answer without redundant details, using a prompt template to ensure clarity.
* **Scoring Modules**: Two scoring functions, llm\_score and advanced\_scorer, evaluate the response. The llm\_score function provides a high-level binary assessment, while advanced\_scorer offers a detailed breakdown.

3.5. Advanced Scoring with JSON Parsing

The advanced scoring method evaluates multiple aspects:

1. **Numerical Accuracy**: Compares values between the response and the ground truth with a margin of error.
2. **Conceptual Correctness**: Validates whether the model’s reasoning aligns with the question requirements.
3. **Context Relevance**: Ensures the answer addresses the question specifically and maintains contextual accuracy.

The scores are aggregated into a JSON object, providing an overall score and a breakdown by category. This approach enables nuanced feedback on the model’s performance.

# 4. Evaluation Metrics

4.1. Metrics Selection  
The evaluation approach leverages both binary and advanced scoring metrics to assess the model’s accuracy across multiple dimensions, using the following metrics:

* **Binary Accuracy**: Provided by the llm\_score function, this metric checks whether the response is numerically equivalent to the ground truth within a specified tolerance.
* **Detailed Scoring Breakdown**: The advanced\_scorer function evaluates responses based on numerical accuracy, conceptual correctness, and context relevance, returning a structured JSON with scores for each category as well as an overall score. This allows for a nuanced view of performance beyond simple correctness.

4.2. Scoring Criteria

The detailed scoring function (advanced\_scorer) breaks accuracy down into three core categories:

1. **Numerical Accuracy**: Compares numerical values with the ground truth, applying a margin of error to account for acceptable variations. This ensures responses are mathematically aligned with the expected answer, which is critical for financial question-answering tasks.
2. **Conceptual Correctness**: Evaluates whether the model's response adheres to correct reasoning. For instance, it assesses whether all relevant components of a financial calculation are included.
3. **Context Relevance**: Ensures the answer directly addresses the user’s question and includes the necessary contextual information. This is especially important in financial contexts where precise interpretation matters.

Each of these categories contributes to the **overall score** in the JSON output, and a final **boolean flag** (is\_correct) indicates whether the answer is satisfactory.

# 5. Experimental Results and Analysis

Performance Analysis  
The model was run on a random sample of 100 questions (k=20 sets of n=5 questions).

The average score was 67%, but varies between 60 and 80 depending on the sample taken. The median score was typically between 97 and 100%.

A graph with a blue line

Description automatically generated

Error Analysis

A row-wise look at the questions and answers for score=0 responses suggests the following:

* Some ground truth values were low quality, leading to a correct response for the model but a discrepancy with the final answer in the training datatset.
  + **Ground-truth:** 21
  + **Model response:** $21 million
  + **Score:** 0.167
  + The ground truth in this case is ambiguous and should not be used to test with.
* Often some questions are not clear enough, for instance asking for values for two years. The model responds with two values (one for each year) rather than a combined figure. This is also shortcoming of only using a single question with no additional prompting, and could be improved by upgrading to a turn based agent approach.
  + **Question**: what is the variation observed in the percentual decrease of the large market same-store and the secondary market same-store during 2014 and 2015?
  + **Ground truth**: 2.3%
  + **Model response**: 2.3 percentage points; 1.5 percentage points
* Some answers are marked as incorrect because of units.
  + **Question:** what was the percentage increase in the pension plan contributions from 2015 to 2016
  + **Ground truth:** 3.57
  + **Model response:** 3.57%
* Some answers were marked as wrong due to a threshold
  + **Question:** what percent of the ratings profile of derivative receivables were junk rated in 2013?
  + **Ground truth:** 20
  + **Model response:** 19.09%
* Some questions were marked as wrong because the model only answers numerically.
  + **Question:** was the c series 2008 annual return greater than the s&p 500?
  + **Ground truth:** yes
  + **Model response:** 2.53%
* When asked for the increase or decrease in a value, the model is marked as incorrect if it gives as a absolute value rather than a percentage (and vice versa).
  + **Question:** considering the years 2017-2018 , what is the increase observed in payment amount per share?
  + **Ground truth:** 27.51%
  + **Model response:** 0.41

# 6. Challenges and Shortcomings

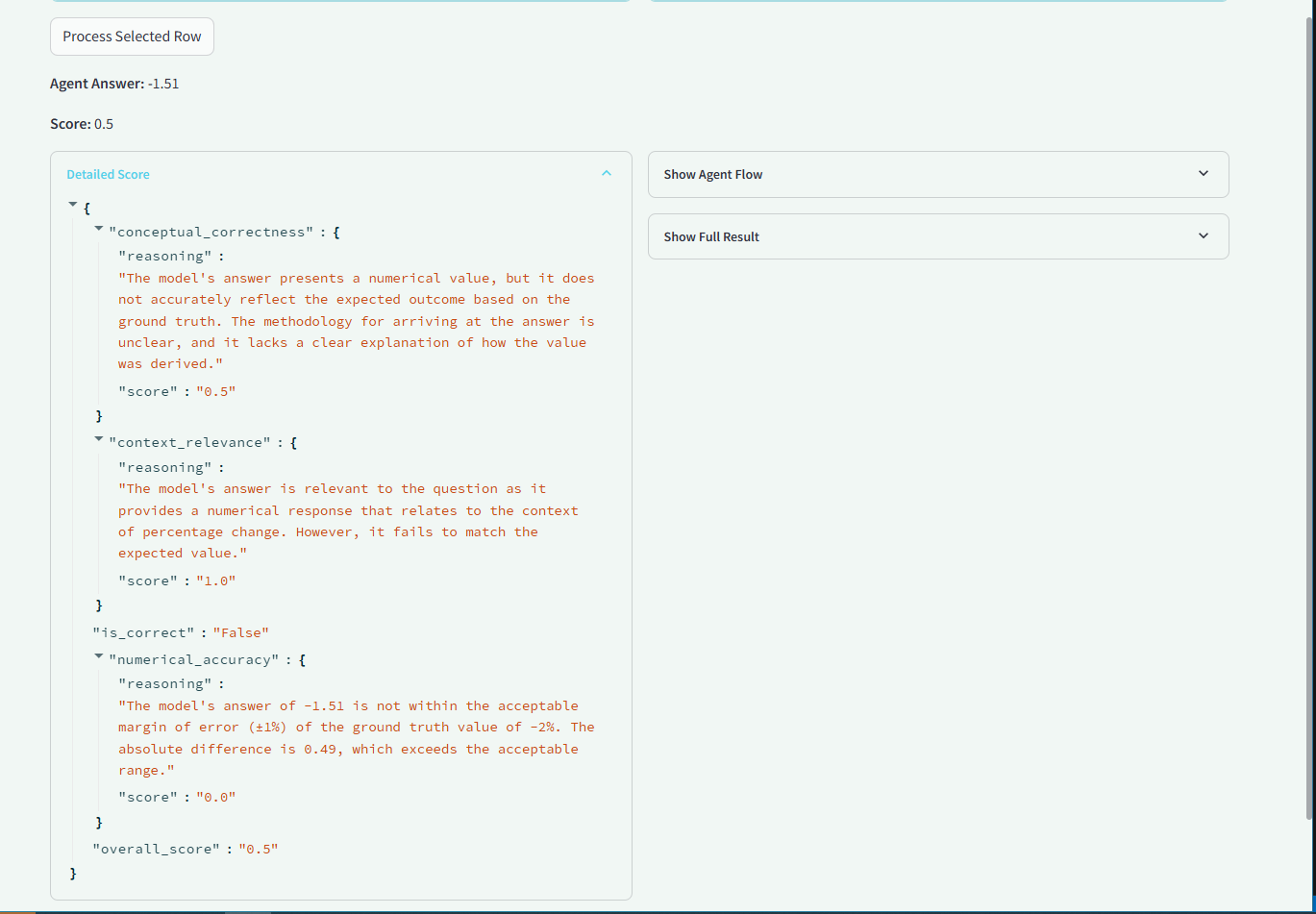
* The model demonstrated promising accuracy and relevance; however, certain challenges were observed in handling specific question types and inconsistencies in response formats. Key areas of concern include:
* **Ambiguities in Ground Truth Data:** In some cases, low-quality or ambiguous ground truth values led to score discrepancies despite correct model responses. For example, the model correctly responded with "$21 million" when the ground truth was simply "21." Here, the lack of context in the ground truth created an unfair evaluation. Such instances highlight the need for improved ground truth consistency, especially in financial contexts where precise units and values matter.
* **Lack of Clarification in Questions:** Certain questions were unclear, leading the model to provide multiple values when a single combined value was expected. For instance, when asked for variations across two years, the model correctly responded with two separate values but was penalized for not combining them. A turn-based approach, where additional prompts clarify the question, could mitigate this issue and improve response accuracy.
* **Unit Consistency Issues:** Some responses were marked incorrect due to unit discrepancies. For example, a response of "3.57%" was penalized against a ground truth of "3.57," even though both values were technically correct. This points to the need for a consistent unit standardization in the scoring process to avoid penalizing correct answers.
* **Threshold and Precision Variances:** The model occasionally provided answers that were slightly off due to rounding or minor precision differences. For example, a response of "19.09%" for a question with a ground truth of "20" was marked incorrect due to a small variance. Introducing an acceptable tolerance range for numerical answers could improve accuracy scores in these cases.
* **Single Metric Responses for Binary Questions:** The model sometimes answered binary (yes/no) questions numerically, such as returning "2.53%" when asked if the annual return exceeded a certain threshold. While the answer technically reflected the correct underlying value, the model’s numerical output format led to it being marked incorrect. Addressing this could involve refining the model to recognize binary question patterns or adjusting evaluation metrics for numerical responses that imply binary conclusions.
* **Absolute vs. Percentage Value Interpretations:** When asked for changes in values, the model occasionally returned absolute values instead of percentages, or vice versa, resulting in penalties. For example, when asked for the "increase in payment amount per share," the model responded with "0.41" instead of the percentage value of "27.51%." Standardizing expectations for absolute or percentage responses within similar question contexts could help reduce this issue.
* Overall, these challenges emphasize the need for clearer ground truth data, question disambiguation strategies, unit standardization, tolerance thresholds, and context-aware responses. Addressing these areas will likely lead to improved scoring consistency and better alignment of the model’s responses with real-world financial information needs.

# 7. User Interface

* To build the application, use “docker-compose up” in the root directory of the repository. It then exposes the port 8501, which means once the container has been built and executed, you can access from your browser at ***localhost:8501***

A screenshot of a computer

Description automatically generated

* The web app opens up on the “Training Data” tab, which gives a view of the questions in the training dataset.
* The pre\_text/table/post\_text data has been concatenated into “source\_text” field for the purpose of testing and scoring. The question is extracted from the qa and qa\_0/qa\_1 fields, depending on if it is of a Type I or Type II conversation.
* You can select a row to process the question for. There is a ground truth assigned to that row.
* You can also click on “**Show Source Text**” to expand into a scrollable container with the original source text the question is asked on.
* Below this, you are also able to customise the question and ground truth.
* You can “**Process Selected Row”** to execute the agent.
* 
* Once the agent has run, you have visibility on the score below, with some explanation on the three pillars of scoring (numerical accuracy, context relevance and conceptual correctness).
* There are also expanders on the right hand side which show the full result and agent flow.

# 8. Conclusions and Future Work

8.1. Future Improvements

* **Adoption of Turn-Based Multi-Prompts:** Transitioning from a single-prompt approach to a turn-based, multi-prompt system could significantly improve the model’s ability to handle complex or ambiguous questions. By allowing follow-up clarification prompts, the model could better interpret nuanced queries—such as those requiring answers for multiple years or calculations—leading to more accurate, context-sensitive responses.
* **Enhanced Scoring Mechanism**: Refining the scoring system to account for acceptable answer variations, minor unit discrepancies, and tolerances in numerical precision would provide a fairer evaluation. Adjustments such as standardized unit recognition, rounding allowances, and differential scoring for absolute vs. percentage values would address issues that arise from overly rigid evaluation criteria.
* **Improving Contextual Awareness in Code Logic:** The model's code could benefit from logic enhancements to better interpret question types and expected answer formats. For example, incorporating logic that distinguishes between binary, numeric, and percentage-based answers could help avoid situations where numeric responses are penalized in binary questions. Additionally, implementing a post-processing layer to convert responses into consistent units or formats before evaluation would further align outputs with ground truth expectations.
* **Automatic Quality Filtering for Training Data:** To reduce ambiguities arising from low-quality or inconsistent ground-truth values, an automated quality-check mechanism could be added to filter out problematic examples before training. This would help ensure the model is trained and evaluated on data that reflects the precise, real-world standards needed for reliable answers.
* **Dynamic Evaluation Margins:** Currently, a static ±1% margin is applied when comparing numerical values. This could be made adaptive:
  + For smaller numbers, a lower tolerance might be more appropriate, while for larger figures, a slightly higher tolerance could prevent minor inaccuracies from impacting the score. Making the margin dynamic could also help address discrepancies in decimal rounding or percentage precision.
* **Improvement of clean\_response Logic:** The clean\_response function is designed to strip extraneous information and standardize the response format, but it could be improved:
  + Integrate unit conversion checks or simple patterns to standardize answers, helping ensure consistency across different questions.
  + Add automated removal of redundant phrases like “The answer is…” or superfluous explanations.
* **Prompt Adjustments Based on Question Type:** At the moment, the system is only able to he model’s responses could be optimized based on question types:
  + For instance, binary questions (yes/no) could trigger a specialized prompt that guides the model to give direct answers without numerical values.
  + Alternatively, questions asking for values across multiple years could activate specific logic to help distinguish between multiple answers.
* **Tool Customization and Expansion**: Currently, only a basic calculator tool is used. To further improve mathematical reasoning, additional or more complex tools could be introduced:
  + A statistics or financial tool could assist with percentage change calculations, and a specialized text-processing tool could help refine qualitative answer evaluation.