A LLM-based RPA System for Optimizing Workflows in Financial Risk Management

Authors: YUAN Tianyu, YUAN Tianyu

Date: 2024-06-05T00:00:00+00:00

Abstract

This paper aims to leverage the advancements in General Computer Control (GCC) to improve the efficiency and effectiveness of risk management operations in financial institutions. Specifically, we introduce an LLM-based Robotic Process Automation (RPA) framework designed to enhance front-line employee work, adapt to the specific needs of financial institutions, and automate tasks requiring minimal cognitive effort. To demonstrate the effectiveness of our proposed framework, stress testing, a common task for risk management department, is used as a case study. The results show that the RPA system can improve efficiency, reduce costs, and minimize errors, all without significantly altering the existing workflow. Moreover, to address customer information security and prompts copyright protection issues, a storage method that separates the server from the client is used. Finally, empirical evidence implies that even models with weaker capabilities can achieve the desired work objectives when guided by detailed prompts.

Full Text

Preamble

A Large-Language-Model-based RPA System for Optimizing Workflows in Financial Risk Management

YUAN Tianyu∗
Business School, University of Hong Kong
u3588064@connect.hku.hk

June 2024

Abstract

This paper leverages advancements in General Computer Control (GCC) to enhance the efficiency and effectiveness of risk management operations in financial institutions. We introduce an LLM-based Robotic Process Automation (RPA) framework designed to augment front-line employee work, adapt to the specific requirements of financial institutions, and automate tasks requiring minimal cognitive effort. To demonstrate the framework's effectiveness, we present stress testing—a common risk management department task—as a case study. The results demonstrate that the RPA system can improve efficiency, reduce costs, and minimize errors without significantly altering existing workflows. Furthermore, to address customer information security and prompt copyright protection concerns, we implement a storage architecture that separates the server from the client. Finally, empirical evidence suggests that even models with weaker capabilities can achieve desired objectives when guided by detailed prompts.

Introduction

In the aftermath of the 2007–2009 global financial crisis, governments and financial regulatory authorities have proactively strengthened oversight by implementing comprehensive laws and regulations. They have also responded promptly to industry initiatives such as the Basel Accords to ensure a more resilient and stable financial system. While these measures aim to reduce substantive risks, they impose a heavy burden on financial institutions, particularly regarding data collection, information reporting, and disclosure requirements. These obligations, designed to promote transparency and information sharing, directly or indirectly increase operating costs, especially for medium and small-sized financial institutions.

Simultaneously, advances in information technology have not substantially resolved these challenges. Many financial institutions, particularly those in developing regions, continue to rely on working methods from the 1990s, such as using VBA to design automated processing scripts. Some practitioners lack training opportunities or motivation to learn basic programming skills and still employ extremely primitive manual processing methods. Historical operational risk events have stemmed from these approaches; for example, a Deutsche Bank employee's data entry error during a foreign exchange trade in 2015 resulted in a $6 billion loss. Research demonstrates that optimizing workflows with RPA systems can improve efficiency, reduce errors, and thereby lower operational risks, which explains why increasing numbers of financial institutions and regulatory authorities encourage employees to learn and use automation tools.

However, traditional RPA programs and advanced language scripts exhibit poor performance in flexibility, adaptability, and crisis handling. Because they consist essentially of fixed-logic and even fixed-sequence instructions, they lack self-feedback and self-adaptation capabilities and must rely entirely on practitioner maintenance and verification.

For these reasons, the middle and back-office departments of financial institutions have long been considered cost centers characterized by large staffs, miscellaneous affairs, and heavy workloads. However, this situation may undergo significant transformation with the maturation of Large Language Models (LLMs) such as GPT. First, LLMs can handle "light-brain work" at relatively low cost and with greater flexibility. Light-brain work refers to repetitive, cyclical, and rigid tasks concentrated in middle and back-office departments, particularly in risk management units established primarily to meet regulatory requirements. These departments manage numerous statistical, analytical, and reporting forms (such as KM-2 tables and R tables). LLM flexibility manifests primarily in their ability to comprehend statistical data terminology, adapt to evolving regulatory requirements, and proactively inspect and maintain models and databases to ensure maximum accuracy and compliance.

Second, LLMs complement existing tools effectively. Large language models are unsuitable for tasks requiring precise quantitative analysis of datasets, such as descriptive statistics, regression analysis, statistical inference, and machine learning models. In contrast, existing risk management tools—whether traditional ones based on numerical statistics within the value-at-risk theoretical framework or emerging ones based on various machine learning models (such as default probability models)—excel in their quantitative precision, a capability currently difficult for LLMs to achieve. LLMs function like a lever: versatile but not specialized, while existing risk management tools resemble items in a toolbox—each serving distinct purposes that cannot substitute for one another.

Third, RPA automation through LLMs can be implemented gradually, humanely, and safely while better protecting enterprise ESG scores. Compared with introducing fully automated RPA from scratch, LLM-based transformation enables human intervention in the proportion of machine-performed work because LLMs replace work by mimicking humans rather than building entirely new processes. Given the continued relevance of financial institutions' existing tools, integrating LLMs into current workflows and aligning them with established systems—rather than completely overhauling and rebuilding—will be the preferred and mainstream approach for light-brain work in the financial industry for the foreseeable future. This paper introduces a practical and adaptable LLM-based RPA framework for current financial institution operations.

2 Literature Review

Following ChatGPT's release, Large Language Model (LLM) applications in the financial industry have expanded rapidly. Yang (2023) [1] introduced FinGPT, an LLM for investment research. Building upon this, Yang (2024) [2] unveiled FinRobot, an Agent framework to automate investment research analyst tasks. Tan (2024) [3] developed intelligent agents that can explore, learn, and apply skills from games, transferable to work scenarios. Zhang (2024) [4] presented FinAgent, a decision-making intelligent agent for investment research.

However, these studies primarily provide abstract frameworks from an academic, top-down perspective and suffer from several shortcomings. First, their design thinking remains distant from actual financial institution practices. Second, they propose idealized one-step visions without considering switching costs and friction expenses for companies. Third, their lack of detailed operational steps and procedures hinders feasibility.

Adopting a bottom-up approach, this paper considers the perspective of front-line financial professionals to explore the advantages and opportunities of LLM-based automation tools. Based on existing financial institution workflows and tools and guided by the principle of marginal improvement, we propose a system that integrates market risk, credit risk, and liquidity risk teams to achieve enterprise risk management, providing a pathway to address the three limitations identified above.

The contributions of this paper are as follows:

- We propose a cost-controlled, highly feasible, and relatively non-disruptive work automation scheme from the perspective of financial institution practices.
- This represents the first public case of an LLM-Agent applied in financial risk management.
- We attempt to solve financial institutions' data security and prompt copyright problems through a server-client separation mechanism.
- We provide further evidence that models with weaker capabilities, when guided by carefully designed prompts, can produce results comparable to high-quality models.
- The automation solutions proposed may be applicable to other types of clerical work.

3 Methodology

The entire system comprises three components: the model interface, the prompt set (server), and user instructions (client), maintained by the large language model company, prompt engineers, and risk management personnel, respectively. The system's upstream involves requirement descriptions from users, while the downstream connects to specific tools such as stress testing systems and VaR calculators.

3.1 Model Interface

The maintainer of the base large language model (also known as the general large language model) provides the model layer interface, which system maintenance personnel manage. The system does not include the LLM itself but drives task execution through data exchange via the interface. This approach offers several advantages: (1) reduced economic and technical costs, (2) low dependence on a single model, which improves continuity and meets Business Continuity

Planning (BCP) requirements, and (3) freedom of choice for customers (e.g., allowing use of locally deployed large language models).

3.2 Prompt Set (Server)

System maintenance personnel design, develop, and maintain prompts for different scenarios for the financial institution's risk management. The prompt set is mapped along two dimensions: type and time. The type dimension follows industry-accepted risk definitions, including credit risk, market risk, counterparty credit risk, operational risk, liquidity risk, reputation risk, strategic risk, etc., with each major category further divided into subcategories. The time dimension follows the timeline of risk exposure, including pre-event, early-stage, mid-event, post-event, and aftermath. This classification provides a fast-locating thinking method and basis for divergent analysis for the LLM while meeting the practical requirements of financial institution risk management.

3.3 User Instructions (Client)

Risk management personnel are the ultimate clients of the system and are responsible for describing instructions and providing data. The instructions and data are isolated from each other, with data never uploaded to the server that provides the prompt set. Specifically, the first step involves the user uploading instructions to the server. The second step requires the server to determine the task category and verify the client's identity before simultaneously but separately sending a "continue" signal to the client and a prompt corresponding to the task category to the model. The third step occurs when the client receives the "continue" signal and then requests data from the user, which is sent directly to the model. The data transmission process must be confirmed by the client's biometric features. The data from the client and the prompt from the server are matched through a key and then input into the large model together. The fourth step involves the large model outputting results, presented as a code set including execution order and parameters, which are returned directly to the client. However, results can only be displayed in plain text after passing through a prompt filter.

Therefore, user instructions and the prompt set are, in principle, separated from each other and are only combined in specific situations (e.g., when the risk management personnel's institution maintains its own prompts). This dual-end separation arrangement aims to solve financial institutions' data security problems and prompt providers' copyright protection issues. However, it should be noted that the dual-end separation only addresses server and client problems but does not affect the large language model's access to data and prompts, which may require solutions such as encryption algorithms (e.g., vectorizing prompts in advance) or legal and industry self-discipline norms.

4 Case Study: Conducting Stress Test – a Common Financial Institution Risk Management Task

4.1 Background

Stress testing is a risk management tool employed by financial institutions to evaluate their resilience under distress conditions. The nature of the test determines its categorization into macro stress tests, reverse stress tests, or quantitative stress tests. This case utilizes the quantitative stress test, a widely adopted method by financial institutions. Specifically, according to an imaging requirement, responsible personnel must calculate the impact on the company's total assets when the benchmark interest rate experiences a single-day decline of 50, 75, 100, 150, and 200 basis points (bp). The comprehensive scope of work encompasses monitoring benchmark interest rate fluctuations (focusing on signals) and conducting sensitivity analysis of the base rate (adjusting parameters).

4.2 Evaluation Criteria

The criterion for a successful test is whether the Agent, constructed based on the RPA framework presented in this paper, can follow macroeconomic indicator instructions, locate the correct program among available tools, input the correct parameters, and execute the program.

4.3 Model and Tools

The Agent in this test was constructed using Autogen as the architecture and GPT-4o as the foundational model. The risk management tools utilized included electronic workbooks, scripts, and applications with practical risk management functionalities, made publicly accessible by university professors on GitHub. GitHub Copilot was also employed to facilitate composition of tool descriptions (function list available in Appendix A).

4.4 Results

The test outcome is straightforward and presented as a screenshot in Figure 2 [Figure 2: see original paper]. Comprehensive results are available in Appendix B.

4.5 Conclusion

The Agent constructed based on the framework in this paper is capable of supplanting risk management personnel in executing stress tests, as demonstrated in the case study.

5 Discussions: Prompts Can Improve the Specific Abilities of Weaker Models

In experiments with various base models, GPT-4o demonstrated strong adaptability to the system, whereas the Gemini series required improvement. Specifically, when the same unadjusted prompt command was input, GPT-4o could identify keywords in fewer dialogue rounds and successfully return the function for calling the tool.

To further investigate model performance, we issued an instruction to execute a stress test and tested each model's performance ten times. The results are presented in Table 1 : after CO#STAR adjustment, models that initially performed poorly also achieved acceptable results. This provides evidence for the viewpoint that "prompts can improve the specific abilities of weaker models."

6 Conclusions

This article discusses the current state of risk management departments in financial institutions, where work is often repetitive and time-consuming, IT technology is relatively outdated, and Robotic Process Automation (RPA) lacks flexibility. We suggest that Large Language Models (LLMs) may address these issues, offering three significant advantages: comparative advantage, complementarity with traditional risk management tools, and positive social externalities.

The paper reviews recent research and identifies a potential shortcoming: existing studies on LLMs and Agents may deviate from actual industry operations. To mitigate these issues, we introduce a new LLM-based RPA framework for risk management in financial institutions. This framework automates tasks requiring minimal cognitive effort and features a three-tiered architecture. Notably, the framework uses separate storage for instructions and prompts to address customer information security and prompt copyright protection.

We demonstrate the proposed framework's performance in a real-world risk management scenario. Our experiments observed that prompt engineering can potentially improve a suboptimal model's ability to perform specific tasks, providing preliminary empirical evidence for this phenomenon.

Risk management is a critical function of financial institutions, acting as both a backstop and a brake. It is based on information-driven judgments central to all financial activities. Different participants' varying degrees of information acquisition and understanding influence their final decisions, which in turn affect financial system variables such as interest rates and asset prices. While RPA can help financial institutions improve efficiency and reduce costs, humans ultimately serve as the measure of value and determinants of development direction. Key decisions must be made, and major responsibilities must be borne, by humans. RPA is merely a tool—like a wheel, a ruler, or a calculator—and should not be considered a substitute for human expertise and decision-making.

Future work includes:

- Continuously refining and expanding the prompt set to encompass risk management practices from a wider range of industries, regions, and positions.
- Enhancing the RPA framework's capabilities to enable autonomous assessment and analysis of input, output, and functional attributes of risk management tools.
- Fostering and maintaining a community of enthusiasts to exchange ideas, share experiences, and collaborate on projects.
- Exploring opportunities to promote and implement the framework in enterprise settings and collaborating with industry partners to tailor the framework to specific needs.
- Building upon the existing framework to investigate LLM potential in assisting and augmenting risk management decision-making processes and evaluating the effectiveness and feasibility of such approaches.

References

[1] Yang, H., Liu, X. Y., & Wang, C. D. (2023). Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

[2] Yang, H., Zhang, B., Wang, N., Guo, C., Zhang, X., Lin, L., … & Wang, C. D. (2024). FinRobot: An Open-Source AI Agent Platform for Financial Applications using Large Language Models. *arXiv preprint arXiv:2405.14767*.

[3] Tan, W., Ding, Z., Zhang, W., Li, B., Zhou, B., Yue, J., … & Lu, Z. (2024). Towards general computer control: A multimodal agent for red dead redemption ii as a case study. *arXiv preprint arXiv:2403.03186*.

[4] Zhang, W., Zhao, L., Xia, H., Sun, S., Sun, J., Qin, M., … & An, B. (2024). FinAgent: A Multimodal Foundation Agent for Financial Trading: Tool-Augmented, Diversified, and Generalist. *arXiv preprint arXiv:2402.18485*.

Appendix

A CO#STAR Prompt Used in Stress Test Scenario

The CO#STAR framework prompt structure employed in our stress testing scenario is structured as follows:

```
Prompt = """
#CONTEXT#
{tools}

#OBJECTIVE#

#STYLE# Formal
```

```
#TONE#

#AUDIENCE#

#RESPONSE#

1. Make sure what base rate is rounded to two
2. [Additional instructions...]
3. If [condition], then tell me what [action]

""".format(tools=function_{list})
```

The associated function list includes the following risk management tools:

```
[
  {
    "suffix": ".xlsm",
    "name": "financial_{predict}",
    "description": "Financial prediction tool",
    "parameters": {
      "type": "object",
      "properties": {
        "keyword": {
          "type": "string",
          "description": "The keyword is"
        }
      },
      "required": ["keyword"]
    }
  },
  {
    "suffix": ".py",
    "name": "calculateVaR",
    "description": "Value at Risk (VaR) used to assess measures",
    "parameters": {
      "type": "object",
      "properties": {
        "keyword": {
          "type": "string",
          "description": "The keyword that's used to"
        }
      },
      "required": ["keyword"]
    }
  },
  {
    "suffix": ".xlsx",
```

```
  "name": "stresstest_{abondon}",
  "description": "(Abandoned) It aims to test",
  "parameters": {
    "type": "object",
    "properties": {
      "keyword": {
        "type": "string",
        "description": "The keyword is"
      }
    },
    "required": ["keyword"]
  }
},
{
  "suffix": ".docx",
  "name": "summary",
  "description": "Summary is a document that summarizes the",
  "parameters": {
    "type": "object",
    "properties": {
      "keyword": {
        "type": "string",
        "description": ""
      }
    },
    "required": ["keyword"]
  }
},
{
  "suffix": ".xlsm",
  "name": "stresstest",
  "description": "Stress test. It aims to test",
  "parameters": {
    "type": "object",
    "properties": {
      "keyword": {
        "type": "string",
        "description": "The keyword is"
      }
    },
    "required": ["keyword"]
  }
},
{
  "suffix": ".xlsm",
  "name": "dailyreport",
```

```
    "description": "Daily report",
    "parameters": {
      "type": "object",
      "properties": {
        "keyword": {
          "type": "string",
          "description": "The keyword that's used to"
        }
      },
      "required": ["keyword"]
    }
  }
]
```

B Full Results of Stress Test Case

The complete execution trace of the stress test case demonstrates the Agent's interaction pattern:

1. **Initial Request**: The user (ManagingPanda) provides benchmark interest rate data and requests stress testing for specified basis point declines.
2. **Tool Identification**: The assistant identifies the appropriate stress testing function from the available toolset based on keyword matching.
3. **Code Generation**: The assistant generates Python code to:
   - Extract the base rate from provided data
   - Calculate stress scenarios for 50, 75, 100, 150, and 200 basis point declines
   - Check if the rate change exceeds thresholds
   - Execute the stress test simulation
4. **Execution and Error Handling**: The initial execution attempt resulted in a `ModuleNotFoundError` indicating the 'functions' module was unavailable. The assistant diagnosed this environment issue and provided corrected code that could execute within the available environment.
5. **Final Output**: The successful execution produced stress test results calculating the impact on total assets under each specified interest rate decline scenario, formatted for risk management reporting.

The execution confirmed that the Agent could autonomously complete the entire stress testing workflow, from interpreting user instructions to selecting appropriate tools, generating execution code, handling errors, and delivering results.

Note: Figure translations are in progress. See original paper for figures.