# FlowMind: Automatic Workflow Generation with LLMs

Zhen Zeng
J. P. Morgan AI Research
New York, NY, USA
zhen.zeng@jpmchase.com

William Watson
J. P. Morgan AI Research
New York, NY, USA
william.watson@jpmchase.com

Nicole Cho
J. P. Morgan AI Research
New York, NY, USA
nicole.cho@jpmorgan.com

Saba Rahimi
J. P. Morgan AI Research
New York, NY, USA
saba.rahimi@jpmorgan.com

Shayleen Reynolds
J. P. Morgan AI Research
New York, NY, USA
shayleen.reynolds@jpmchase.com

Tucker Balch
J. P. Morgan AI Research
New York, NY, USA
tucker.balch@jpmchase.com

Manuela Veloso
J. P. Morgan AI Research
New York, NY, USA
manuela.veloso@jpmchase.com

## ABSTRACT

The rapidly evolving field of Robotic Process Automation (RPA) has made significant strides in automating repetitive processes, yet its effectiveness diminishes in scenarios requiring spontaneous or unpredictable tasks demanded by users. This paper introduces a novel approach, **FlowMind**, leveraging the capabilities of Large Language Models (LLMs) such as Generative Pretrained Transformer (GPT), to address this limitation and create an automatic workflow generation system. In FlowMind, we propose a generic prompt recipe for a lecture that helps ground LLM reasoning with reliable Application Programming Interfaces (APIs). With this, FlowMind not only mitigates the common issue of hallucinations in LLMs, but also eliminates direct interaction between LLMs and proprietary data or code, thus ensuring the integrity and confidentiality of information — a cornerstone in financial services. FlowMind further simplifies user interaction by presenting high-level descriptions of auto-generated workflows, enabling users to inspect and provide feedback effectively. We also introduce **NCEN-QA**, a new dataset in finance for benchmarking question-answering tasks from N-CEN reports on funds. We used NCEN-QA to evaluate the performance of workflows generated by FlowMind against baseline and ablation variants of FlowMind. We demonstrate the success of FlowMind, the importance of each component in the proposed lecture recipe, and the effectiveness of user interaction and feedback in FlowMind.

## CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Cognitive robotics**; • **Information systems** → **Users and interactive retrieval**.

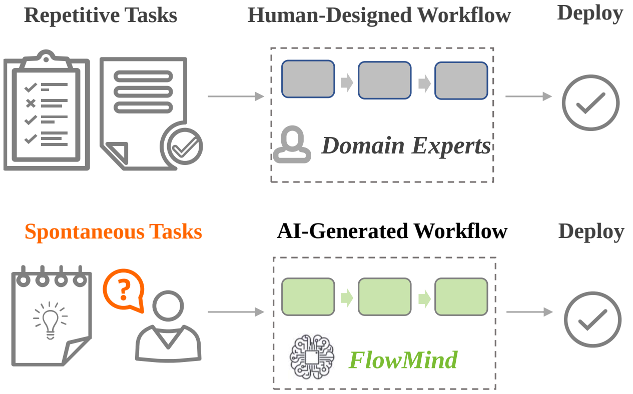


**Figure 1: FlowMind automates spontaneous tasks demanded by users through on-the-fly workflow generation, advancing beyond traditional automation of repetitive tasks designed by domain experts.**

## KEYWORDS

cognitive workflow, user query, information retrieval

**ACM Reference Format:**
Zhen Zeng, William Watson, Nicole Cho, Saba Rahimi, Shayleen Reynolds, Tucker Balch, and Manuela Veloso. 2023. FlowMind: Automatic Workflow Generation with LLMs. In *4th ACM International Conference on AI in Finance (ICAIF '23), November 27–29, 2023, Brooklyn, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3604237.3626908

## 1 INTRODUCTION

The paradigm of Robotic Process Automation (RPA) has vastly transformed the landscape of task execution by enabling the automation of repetitive processes. However, its reliance on expert knowledge and well-defined procedures can fall short in situations where more spontaneous or unpredictable tasks arise. To address these challenges, we explore the capabilities of Large Language Models (LLMs) and present a framework that leverages their potential to create a versatile, dynamic, and secure workflow generation system.

*ICAIF '23, November 27–29, 2023, Brooklyn, NY, USA*

ACM ISBN 979-8-4007-0240-2/23/11…$15.00
https://doi.org/10.1145/3604237.3626908

In this paper, we introduce a novel approach, **FlowMind**, that enables automatic workflow generation using LLMs, following the proposed generic lecture recipe of prompt design. Through an extensive and rigorous study, we dissect each component of our prompt design to demonstrate its importance and contributions to the overall effectiveness of automatic workflow generation. FlowMind allows us to harness the vast capabilities of LLMs, specifically Generative Pretrained Transformer (GPT) model, in a more defined and structured manner, leading to robust and efficient code generation for workflow execution.

A key feature of our framework lies in its robustness against hallucinations often experienced with LLMs. We ground the reasoning of LLMs with the aid of Application Programming Interfaces (APIs). These APIs are reliable functions developed and tested by domain experts, ensuring their accuracy and reliability. Proprietary software developed in industry is typically composed of such reliable APIs. FlowMind is able to leverage APIs provided to it while ensuring that the LLMs do not directly interact with any proprietary code or data, protecting code and data privacy. This protection is achieved by allowing LLMs to act only on the high-level descriptions of the APIs, enhancing security and ensuring a reliable generation of workflows.

Understanding the necessity for human oversight, our system also integrates user feedback. Without assuming the programming experiences of the user, the system provides a high-level description of the auto-generated workflow, allowing novice users to inspect and provide feedback. FlowMind then takes the user feedback and adjusts the generated workflow if needed. This two-way interaction empowers users to enhance the workflow based on their knowledge and the unique demands of their tasks, thus enhancing the flexibility and adaptability of our system.

As part of our contribution to the broader research community, we provide a benchmark dataset **NCEN-QA** in finance, built on top of N-CEN reports[1]. This dataset serves as a robust platform for evaluating workflow generation systems, particularly in question-answering tasks in the domain of funds. Our experiments on **NCEN-QA** demonstrate the effectiveness of our proposed method. We believe this dataset will open new avenues for performance comparisons and drive the advancement of workflow generation research in finance.

Overall, our contributions are three-fold:

- We propose **FlowMind**, a novel approach that facilitates automatic workflow generation using LLMs, which addresses hallucination and data privacy concerns, underpinned by a generic lecture recipe of prompt design.
- Our system incorporates a feedback mechanism enabling users to effectively inspect and provide inputs to refine the generated workflow when needed.
- We offer a benchmark dataset **NCEN-QA** in the field of Finance for evaluating workflow generation in handling question-answering tasks on funds.

Next, we will introduce and discuss the relevant literature in section 2, the proposed FlowMind method 3, as well as experiments results and analysis in section 4.

# 2 RELATED WORK

## 2.1 Robotic Process Automation

Robotic Process Automation (RPA) has been widely recognized for its potential to automate repetitive tasks in various industries [11, 22, 30]. For instance, repetitive tasks such as data entry, invoice processing, or customer service responses have been successfully automated using RPA [10, 28]. These tasks often involve pre-defined workflows that, once created, can be reused multiple times, significantly enhancing the efficiency and accuracy of the tasks. Based on rule-based business processes, or by observing human digital actions, RPAs provide a software solution to automating repetitive tasks and has seen success across various domains. However, this approach requires expert knowledge and well-defined procedures or steps [9], which are not always available, especially for more spontaneous or unpredictable tasks. Additionally, RPAs lack objective reasoning around its application or development [26]. When dealing with such tasks, which users might interactively demand, the limitation of RPA becomes apparent. Pre-defined workflows are inadequate for handling such situations, necessitating automatic workflow generation to manage them effectively.

## 2.2 LLMs for Coding

Language Models (LLMs), especially in their use for code generation, have seen considerable exploration and advancement [16, 18, 27]. The rise of the Generative Pretrained Transformer (GPT) models [19, 20] has spurred further exploration, specifically regarding their potential to generate code in various domains [13, 29]. Furthermore, prior work has explored chain of thought through code, as demonstrated in [12] for robotic programs, action plan generation [6], web browsing [15], learning tools [24], or generating valid arithmetic programs [8]. Recently, LLMs have shown the ability to construct modular code for visual question answering based on abstractions of high-level APIs [25]. These studies utilized techniques like prompt engineering and few-shot learning to harness the capabilities of GPT for code generation. However, while promising, much of this work has been somewhat anecdotal, providing qualitative examples and demonstrations but lacking rigorous analysis of quantitative performance against benchmark datasets. Few LLMs have been fine-tuned on finance-related tasks [14, 31–33], however, none of these focused on leveraging LLMs for coding. Our work leverages LLMs for coding workflow structures and we introduce a new dataset in Finance for rigorous evaluation.

## 2.3 Workflow Generation with LLMs

Building on the capabilities of LLMs for code generation, recent advancements have seen the development of application tools that leverage LLMs for workflow generation, such as Langchain [2], HuggingFace's Transformer Agent [5], and AutoGPT [4]. Langchain, for instance, focuses on integrating LLMs with external data, utilizing a scalable approach to chunk data for interaction with a large database. However, applying this in fields like finance raises serious data privacy concerns because the model directly interacts with the data. With powerful LLMs like GPT, where inference doesn't occur locally, it is critical to limit direct interaction with private data as

[1] https://www.sec.gov/files/formn-cen.pdf

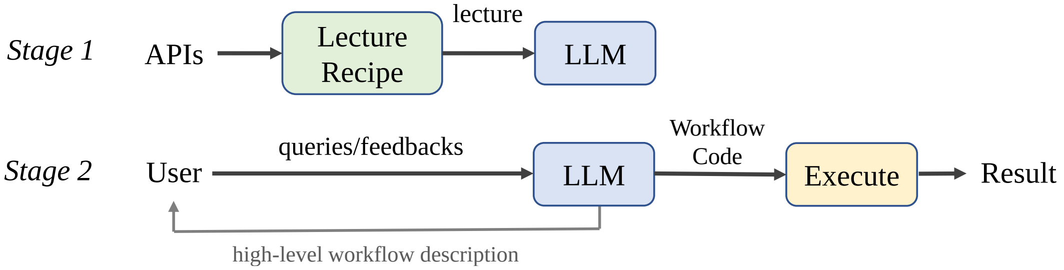


**Figure 2: Overview of FlowMind framework. (a) Stage 1: we follow the proposed generic *lecture* recipe to generate a lecture prompt, which educates the LLM about the context, APIs, and get ready to write code; (b) Stage 2: LLM can then take user queries or tasks and auto-generate the workflow code that makes use of the introduced APIs. The workflow code is executed to deliver the result. During stage 2, we enable a feedback loop between FlowMind and the user, where FlowMind provides high-level description of the generated workflow in plain-language, and the user inputs feedback to FlowMind to approve or refine the workflow if needed.**

much as possible. On the other hand, Transformer Agent, while being a novel approach to task automation, is primarily aimed at combining different transformer models for user-requested tasks. Thus, it might not be flexible enough to easily incorporate other types of functions/models typically developed as proprietary software in various industries. AutoGPT provides limited problem-solving capabilities due to the limited set of functions in its library, such as web browsing and executing code. Our proposed method, in contrast to the aforementioned tools, offers a more generic and adaptable approach to workflow generation, allowing user requests to be handled without 1) breaching data privacy or 2) making assumptions about the types of functions/models to be combined during code generation.

## 3 METHOD

The FlowMind framework functions in two primary stages as illustrated in Figure 2: 1) initial lecture to the LLM on the task context and available APIs, and 2) subsequent workflow generation using the APIs and workflow execution to deliver the result to user. During stage 2, we enable an optinal feedback loop between FlowMind and the user such that user can easily review and provide feedback to FlowMind, and FlowMind can adjust the generated workflow accordingly.

### 3.1 Lecture to LLM

The first stage of the FlowMind framework involves a *lecture* on the context, available APIs, and the need to generate workflow code for the LLM. We adhere to our proposed **generic lecture recipe** to generate an informative lecture on the context and APIs, ensuring the LLM has a clear understanding of the overall goal, as well as the scope, inputs, and outputs of the functions in the APIs. The lecture recipe is crafted with three components, each with a distinct role. Specifically, the proposed lecture prompt recipe covers:

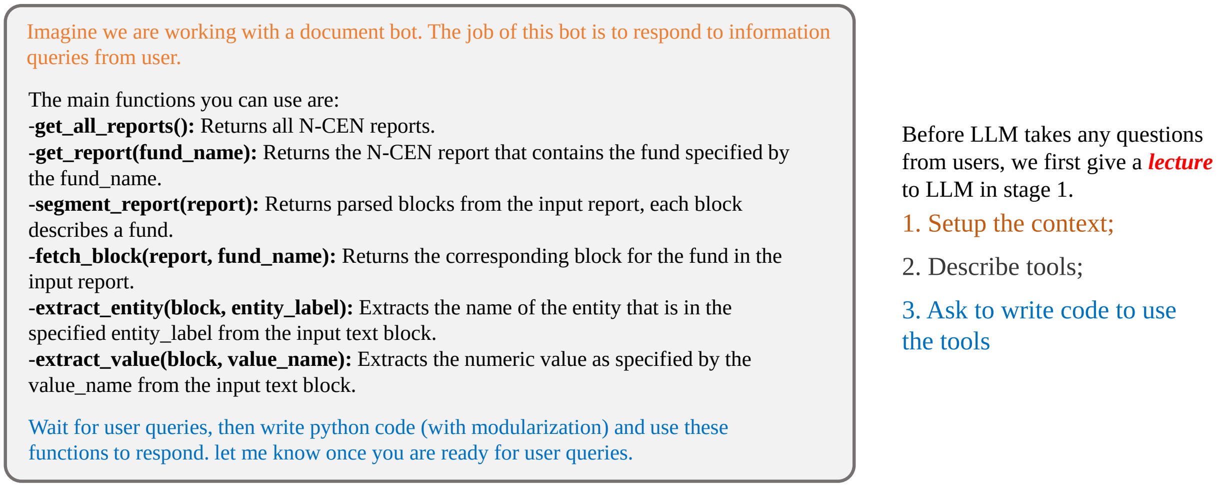
- *Context*: First we introduce the context which covers the domain of the expected tasks/queries from the user. For example, in our experiments, we set up the context as handling information queries from user, as shown in Figure 3.
- *APIs*: Then we provide a list of structured descriptions of the available APIs to use for the LLM. Importantly, we introduce the name of the function, the input arguments, and the output variables. Note that the function names, input arguments, and output descriptions must be semantically meaningful and relevant to the context above such that the LLM can comprehend to make good use of the functions.
- *Code*: Lastly we ask the LLM to prepare to write the workflow code using the provided APIs upon receiving user query/task.

An example of such a lecture in our experiments is shown in Figure 3. The crafted prompt following the lecture recipe enables the LLM to gain the necessary understanding of the context and available APIs, to utilize them in the subsequent stage of workflow generation effectively. We highlight the effect of each component in the lecture recipe in our experiments.

### 3.2 Workflow Generation and Execution

In the second stage, LLM leverages the API knowledge gained from the first stage to take user queries or tasks and generate corresponding workflow code. This stage involves two key components: code generation and code execution. In code generation, LLM creates a workflow, making use of the introduced APIs to address the user's query or task effectively. The workflow is then executed to generate the output to user. We show examples of the auto-generated workflows by FlowMind and derived answers to several sample user questions in our Experiments (Figure 4, 5, 6).

A distinct feature of FlowMind is the ability to take user feedback during the second stage. The system presents a high-level description of the generated workflow to the user, enabling users to understand the workflow's functionality and structure without the need to closely examine the underlying code. This allows the users to effectively provide feedback on the generated workflow, which the LLM can then incorporate to refine the workflow if necessary, ensuring that the system accurately addresses the user's needs. We prompt LLM with "Could you provide a concise high-level summary of the flow of code? Then take feedback to see if code needs to be updated" to get the high-level workflow description. Examples are discussed and shown in Experiments (Figure 7).



Imagine we are working with a document bot. The job of this bot is to respond to information queries from user.

The main functions you can use are:
-**get_all_reports():** Returns all N-CEN reports.
-**get_report(fund_name):** Returns the N-CEN report that contains the fund specified by the fund_name.
-**segment_report(report):** Returns parsed blocks from the input report, each block describes a fund.
-**fetch_block(report, fund_name):** Returns the corresponding block for the fund in the input report.
-**extract_entity(block, entity_label):** Extracts the name of the entity that is in the specified entity_label from the input text block.
-**extract_value(block, value_name):** Extracts the numeric value as specified by the value_name from the input text block.

Wait for user queries, then write python code (with modularization) and use these functions to respond. let me know once you are ready for user queries.

Before LLM takes any questions from users, we first give a ***lecture*** to LLM in stage 1.

1. Setup the context;

2. Describe tools;

3. Ask to write code to use the tools

**Figure 3: Before an LLM takes any queries or tasks from users, we first give a *lecture* to it. We show an example of such a lecture above. The proposed generic lecture recipe includes: 1) setting up the context, 2) enumerating the available APIs with each function declaration, parameters, and high-level descriptions, and 3) prompting the LLM to write workflow code using these APIs.**

# 4 EXPERIMENTS

To evaluate the effectiveness of FlowMind, we propose a benchmark dataset, NCEN-QA, derived from N-CEN reports on funds in the domain of Finance. In addition, N-CEN APIs were provided to FlowMind to consume and extract information from N-CEN reports. We used GPT as the LLM in the proposed framework.

By lecturing FlowMind on the N-CEN APIs, we carried out rigorous assessments of FlowMind's ability to auto-generate workflows to handle question-answering tasks in NCEN-QA. We demonstrate that FlowMind, even without user feedbacks, already significantly outperformed GPT question answering based on context retrieval, a commonly adopted methodology in the literature [17, 21, 23]. Our ablation studies reveal the importance of each component in our lecture prompt recipe introduced in Section 3.1. Furthermore, we show that FlowMind's performance can be further boosted when user feedbacks are incorporated.

Next, we first introduce N-CEN reports in the finance domain and how we composed the NCEN-QA dataset in Section 4.1, discuss the N-CEN APIs in Section 4.2, then the benchmarked methods in Section 4.3, and experiments results in Section 4.4. Throughout the experiments, we used `gpt-3.5-turbo` with temperature 0 to minimize randomness.

## 4.1 NCEN-QA

We composed the NCEN-QA dataset by creating 600 question-answer pairs centered on fund data found in N-CEN reports. N-CEN reports are mandatory annual filings for registered investment companies in the US, with each report submitted on a trust level, meaning each company could submit one report representing numerous funds. These reports provide a wealth of information on a range of funds, including their custodians, pricing services, investment advisors, gross commission, fund net assets, etc.

We crawled the most recently filed N-CEN reports from each reporting investment company on the public database Edgar [1] from the U.S. Securities and Exchange Commission (SEC) site. We scraped the reports across the past 3 years (prior to May 11th, 2023), at a maximum throughput of 10 reports per sec (SEC site limits for a single host), yielding 8,548 reports. Among these reports, there are duplicates of reported funds, and we cleaned the data to retain the most recent filing of each fund. As a result, we gathered 2,794 reports which covered a grand total of 12k funds. Based on the crawled data, we then composed sets of fund questions at varying difficulty levels **NCEN-QA-Easy**, **NCEN-QA-Intermediate**, and **NCEN-QA-Hard**, each containing 200 question-answer pairs, as explained below.

### *4.1.1* ***NCEN-QA-Easy.***

In this set, each question focuses on a single piece of information about a fund, requiring the investigation of only one N-CEN report in which the fund was reported, and in particular the specific block of texts describing the queried fund. Example questions include:

- Q1: Who is the custodian for the Precious Metals Mutual Fund?
- Q2: What is the gross commission for the Rule One Fund?

We sampled 200 funds and derived the answers accordingly from N-CEN reports, encompassing both types of questions as shown in Q1 and Q2. Q1-type questions are focused on entities that provide certain services for the queried fund, including the custodian, investment advisor, collateral manager, administrator, and pricing services. On the other hand, Q2-type questions are focused on numbers such as gross commission and purchase sales. We kept the 200 question-answer pairs roughly evenly spread over the aforementioned fund information.

### *4.1.2* ***NCEN-QA-Intermediate.***

In this set, each question also focuses only on one fund. However, the question is slightly more complex than the ones in NCEN-QA-Easy because it requires mathematical operations (e.g. division) on top of multiple pieces of information regarding the queried fund. Example questions include:

- Q1: What is the ratio of the gross commission against fund net assets for the SFT International Bond Fund?
- Q2: What is the ratio of the total purchase sale against fund net assets for the Core Fixed Income Portfolio?

Similarly to NCEN-QA-Easy, we sampled 200 funds and derived the answers manually from N-CEN reports. We evenly spread the 200 question-answer pairs over the Q1- and Q2-type of questions.

#### 4.1.3 ***NCEN-QA-Hard***.

In this set, each question focuses on multiple funds, thus presenting a bigger challenge than NCEN-QA-Easy and NCEN-QA-Intermediate. Some example questions are shown below. Q1-type questions require aggregation across multiple funds. Q2-type questions require a full investigation of **all** reported funds and gather all the funds which are using the specified services (e.g. investment advisor being AlphaMark Advisors), the answer could range from one to multiple funds.

- Q1: What is the gross commission aggregated over funds ClearBridge Dividend Strategy Fund, Baird Mid Cap Growth Fund, and Baron Discovery Fund?
- Q2: What funds do the investment advisor company AlphaMark Advisors, LLC manage?

For Q1-type questions, we sampled around 70 tuples of funds and varied the questions between "gross commission" and "total purchase sale". For Q2-type questions, we sampled around 130 service companies across custodians, investment advisors, collateral managers, administrators, and pricing services, and asked about the funds that are using each of these service companies. The longest answer for Q2-type questions in our dataset contained 12 funds. In total, we again had 200 question-answer pairs in this set, same as the other two sets.

## 4.2 N-CEN APIs

In FlowMind, we ground the ability of LLMs to reason with reliable Application Programming Interfaces (APIs), as discussed in Section 3. The strength of APIs lies in their robustness, having been designed by domain experts capable of handling vast amounts of data in a structured, parallelized, and deterministic manner. Combining LLMs with APIs stands in stark contrast to solely relying on LLMs alone, which can exhibit limitations such as hallucination and token restrictions.

In our experiments, we developed domain-specific APIs for processing N-CEN reports. FlowMind, in turn, is provided with high-level natural language descriptions of these APIs and is tasked with autonomously generating workflows. These workflows consist of a series of calls to the given APIs, ultimately formulating the answer to the user's query step by step.

We incorporated 6 crucial APIs into our framework as enumerated in Figure 3. These APIs encapsulate specific subject matter expertise concerning fund structures and the SEC reporting mechanism, enabling accurate parsing and extraction of information from structured content. Our APIs cover 3 main aspects: *retrieval*, *partition*, and *extract*, as explained below in detail.

#### 4.2.1 *Retrieval.*

In order to fetch the correct N-CEN report, we rely on *rapidfuzz* [7] to disambiguate the input fund name into an N-CEN Accession Number, a unique identifier linking that fund to its most recent report in the `get_report` function. `get_all_reports` gathers all the cached results of `get_report` applied to every fund name in our dataset. In addition, since a single report may contain multiple funds, `fetch_block` can retrieve a single fund block of text within the input report given a specified fund name.

#### 4.2.2 *Partition.*

`segment_report` function segments an input N-CEN report into a list of fund blocks. This is useful for cases where all funds in a report must be examined for a query.

#### 4.2.3 *Extract.*

`extract_entity` and `extract_value` are responsible for extracting entity names and certain numbers, respectively. Entities are reported in XML-like blocks with names, identifiers, and other pertinent information such as state, country, classification, etc. Therefore, `extract_entity` discovers all named entities in a fund block. We again use *rapidfuzz* to match a query entity label to the ones discovered in a fund block (e.g., 'custodian', 'collateral manager'). Similarly, `extract_value` extracts named values such as 'gross commission' and 'purchase sales'.

## 4.3 Benchmarks

We benchmarked the proposed method FlowMind with and without user feedback, against the commonly adopted methodology of LLMs question answering based on context retrieval (**GPT-Context-Retrieval**). We also carried out an ablation study where we altered each component in the proposed generic lecture recipe (Section 3.1) and benchmarked the performance of each ablation variant of FlowMind.

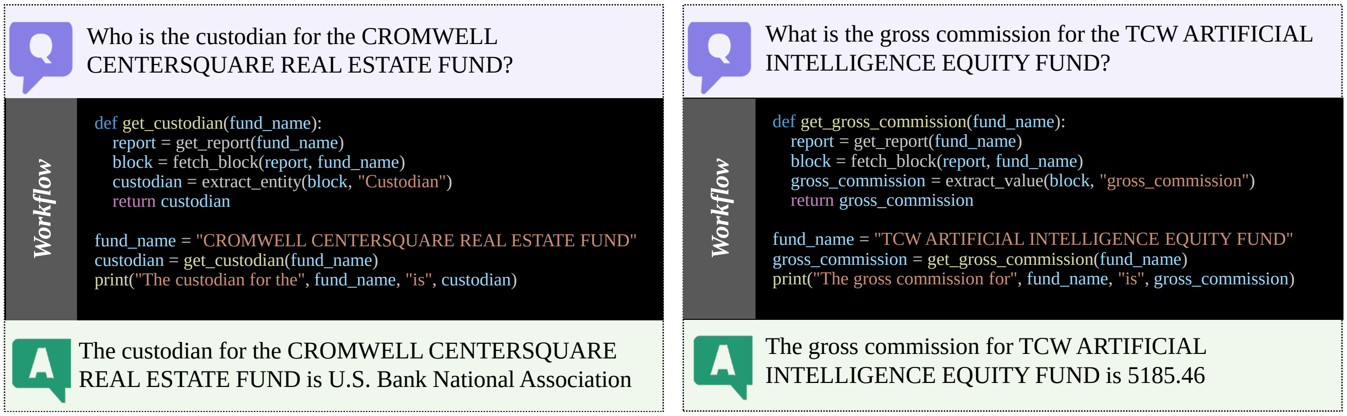
#### 4.3.1 *GPT-Context-Retrieval.*

A common strategy of question answering without re-training LLMs is to retrieve relevant context to a user question, and prepend the retrieved context to the user question when prompting LLMs [17, 21, 23]. In our experiments, we used GPT as the LLM and `text-embedding-ada-002` [3] to measure the relevancy between a given context and a question. First, we used `get_all_reports` and `segment_report` (as introduced in Section 4.2) to get the full list of segmented fund blocks, where each fund block contains the information about a fund. Then, we embed each fund block and the user question into a latent vector using `text-embedding-ada-002`. We then compare the embedded fund blocks and the embedded user question via cosine similarity to identify the top $k$ fund blocks that are most relevant to the question. These fund blocks are then prepended to the question when prompting GPT to output the answer. We used $k = 1$ when evaluating on NCEN-QA-Easy and NCEN-QA-Intermediate because each question is regarding a single fund, and $k = 3$ when evaluating on NCEN-QA-Hard because the questions are regarding multiple funds.

Note that some fund blocks are too long during embedding and must be truncated to 8,191 tokens due to the input limit of `text-embedding-ada-002`. When prompting GPT, we also had to truncate the retrieved top $k$ fund blocks so the prompt is within the 4,096 token limit. These truncations inevitably threw away information that could be useful for question answering.

| | Baseline | Ablation | | | Proposed | |
|---|---|---|---|---|---|---|
| **Accuracy** ↑ | GPT-Context-Retrieval | FlowMind-NCT | FlowMind-BA | FlowMind-NCP | FlowMind | FlowMind+feedback |
| NCEN-QA-Easy | 63.5% | 88.0% | 58.0% | 2.5% | **99.5%** | **100.0%** |
| NCEN-QA-Inter | 28.0% | 91.0% | 52.5% | 0.0% | **99.0%** | **100.0%** |
| NCEN-QA-Hard | 8.5% | 67.8% | 28.6% | 6.5% | **89.5%** | **96.0%** |

**Table 1: Accuracy of outputs from all benchmarked methods. The proposed methods outperformed the baseline significantly. The ablation study reveals the importance of each component in the proposed generic lecture recipe.**



Who is the custodian for the CROMWELL CENTERSQUARE REAL ESTATE FUND?

```
def get_custodian(fund_name):
    report = get_report(fund_name)
    block = fetch_block(report, fund_name)
    custodian = extract_entity(block, "Custodian")
    return custodian

fund_name = "CROMWELL CENTERSQUARE REAL ESTATE FUND"
custodian = get_custodian(fund_name)
print("The custodian for the", fund_name, "is", custodian)
```

The custodian for the CROMWELL CENTERSQUARE REAL ESTATE FUND is U.S. Bank National Association

What is the gross commission for the TCW ARTIFICIAL INTELLIGENCE EQUITY FUND?

```
def get_gross_commission(fund_name):
    report = get_report(fund_name)
    block = fetch_block(report, fund_name)
    gross_commission = extract_value(block, "gross_commission")
    return gross_commission

fund_name = "TCW ARTIFICIAL INTELLIGENCE EQUITY FUND"
gross_commission = get_gross_commission(fund_name)
print("The gross commission for", fund_name, "is", gross_commission)
```

The gross commission for TCW ARTIFICIAL INTELLIGENCE EQUITY FUND is 5185.46

**Figure 4: NCEN-QA-Easy: example questions, corresponding workflow and result generated by FlowMind.**

*4.3.2 FlowMind-NCT.* The first ablation variant of FlowMind is FlowMind-**N**o**C**on**T**ext, which means we don't provide the context when giving the lecture to GPT in FlowMind. This is to study the effect of the *Context* component as explained in Section 3.1. Specifically, we removed the first context sentence from the lecture prompt to GPT.

*4.3.3 FlowMind-BA.* The second ablation variant of FlowMind is FlowMind-**B**ad**A**pis, which means we don't provide semantically meaningful names for the input arguments in the APIs' high-level descriptions. This is to study the effect of the *APIs* component as explained in Section 3.1. For example, the description of `get_report(fund_name)` becomes `get_report(x)`, followed by "Returns the N-CEN report that includes the fund x".

*4.3.4 FlowMind-NCP.* The third ablation variant of FlowMind is FlowMind-**N**o**C**ode**P**rompt, which means we don't explicitly ask GPT to write code. This is to study the effect of the *Code* component as explained in Section 3.1. In particular, the last sentence in the lecture prompt to GPT becomes "Wait for user queries, then try to use these functions to respond assuming you have access to these functions. Let me know once you are ready for user queries".

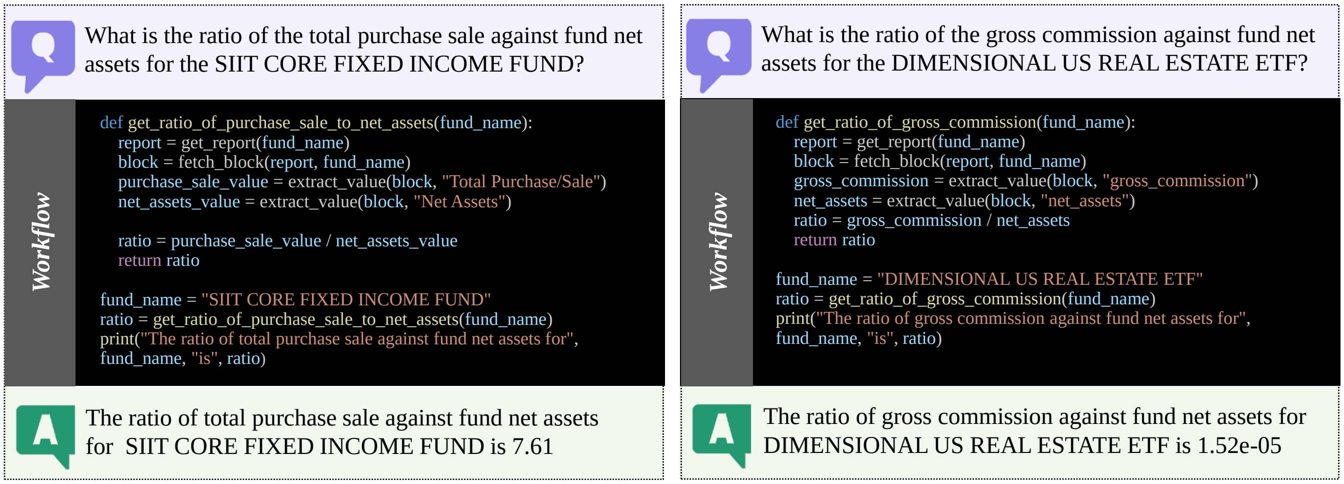
## 4.4 Results

We evaluated the proposed methods and all the methods mentioned in Section 4.3 using the curated NCEN-QA dataset. We report the benchmarked accuracy of each method in Table 1. The accuracy was measured by comparing the output answer to the ground truth. For questions related to entity names, an answer is considered accurate only if it captures all the entity names in the ground truth. For questions related to numbers, an answer was deemed accurate if it matched the ground truth number after rounding to the ground truth's precision.

Our results revealed that FlowMind, even without user feedback, significantly outperformed the GPT-Context-Retrieval baseline method. We show some qualitative examples of FlowMind in Figure 4, 5, 6. The baseline's performance notably degrades as the complexity of questions increases, often due to 1) truncation on the retrieved context to fit within GPT input token limits, 2) hallucination, 3) retrieval of incorrect context, and 4) inaccurate number calculation.

In our ablation study, we found each component of the lecture recipe to be vital for FlowMind's success. FlowMind-NCT, which excluded context in the lecture prompt, performed worse than the standard FlowMind, emphasizing the need for context. FlowMind-BA's poor performance highlighted the importance of quality API descriptions with semantically meaningful argument names; it often failed due to incorrect API calls or use of wrong arguments. FlowMind-NCP, which did not generate any executable workflow code in most cases, yielded the lowest accuracy, indicating the necessity of explicit prompting LLM to write code to express the workflow that derives the answer.

We further improved FlowMind's performance by incorporating user feedback, allowing it to update the workflow after generating the first executable workflow code based on user suggestions. This led to increased accuracy, as evident in NCEN-QA-Easy shown in Figure 7, where a question was about the purchase sale of a fund with 'February' in its name. Initially, FlowMind misunderstood the question, interpreting 'February' as the month of a fund's purchase sale, rather than recognizing it as part of the fund's name. With user feedback, however, this confusion was clarified, and FlowMind adjusted its workflow to correctly interpret 'February' as part of the



What is the ratio of the total purchase sale against fund net assets for the SIIT CORE FIXED INCOME FUND?

```
def get_ratio_of_purchase_sale_to_net_assets(fund_name):
    report = get_report(fund_name)
    block = fetch_block(report, fund_name)
    purchase_sale_value = extract_value(block, "Total Purchase/Sale")
    net_assets_value = extract_value(block, "Net Assets")

    ratio = purchase_sale_value / net_assets_value
    return ratio

fund_name = "SIIT CORE FIXED INCOME FUND"
ratio = get_ratio_of_purchase_sale_to_net_assets(fund_name)
print("The ratio of total purchase sale against fund net assets for",
fund_name, "is", ratio)
```

The ratio of total purchase sale against fund net assets for SIIT CORE FIXED INCOME FUND is 7.61

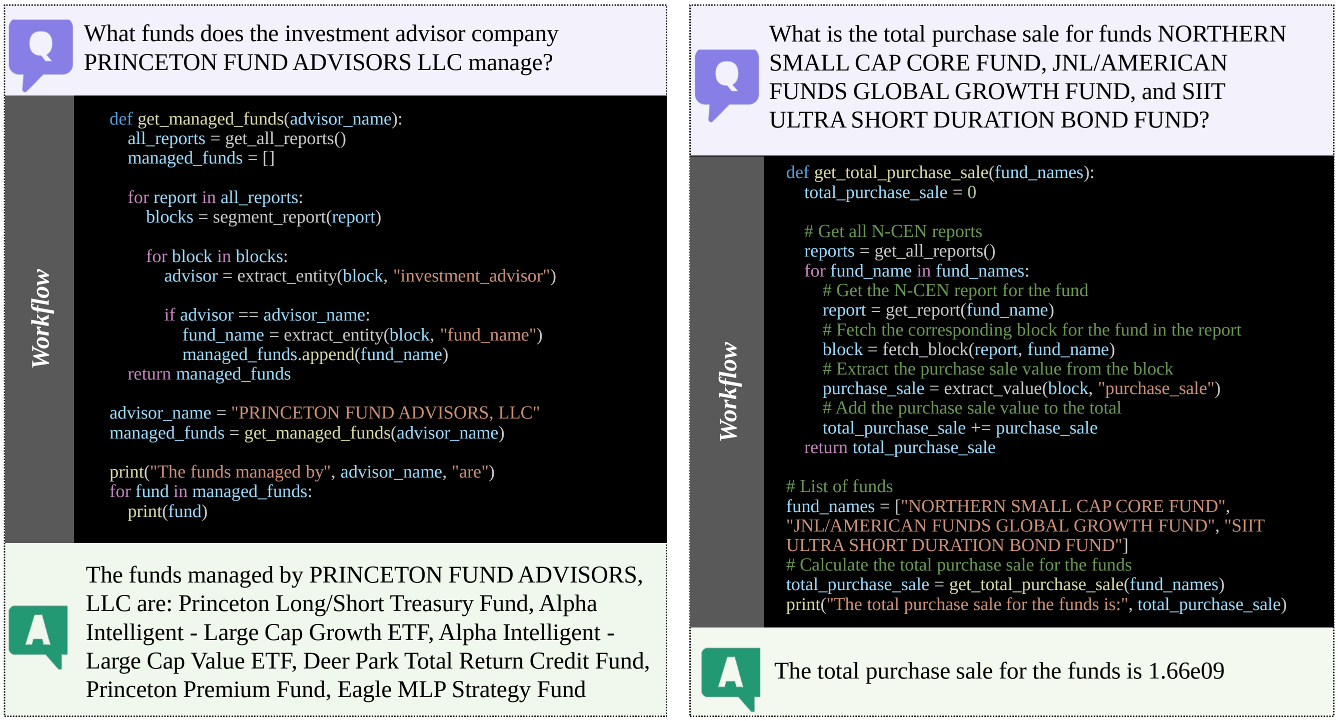
What is the ratio of the gross commission against fund net assets for the DIMENSIONAL US REAL ESTATE ETF?

```
def get_ratio_of_gross_commission(fund_name):
    report = get_report(fund_name)
    block = fetch_block(report, fund_name)
    gross_commission = extract_value(block, "gross_commission")
    net_assets = extract_value(block, "net_assets")
    ratio = gross_commission / net_assets
    return ratio

fund_name = "DIMENSIONAL US REAL ESTATE ETF"
ratio = get_ratio_of_gross_commission(fund_name)
print("The ratio of gross commission against fund net assets for",
fund_name, "is", ratio)
```

The ratio of gross commission against fund net assets for DIMENSIONAL US REAL ESTATE ETF is 1.52e-05

**Figure 5: NCEN-QA-Intermediate: example questions, corresponding workflows and results generated by FlowMind.**



What funds does the investment advisor company PRINCETON FUND ADVISORS LLC manage?

```
def get_managed_funds(advisor_name):
    all_reports = get_all_reports()
    managed_funds = []

    for report in all_reports:
        blocks = segment_report(report)

        for block in blocks:
            advisor = extract_entity(block, "investment_advisor")

            if advisor == advisor_name:
                fund_name = extract_entity(block, "fund_name")
                managed_funds.append(fund_name)
    return managed_funds

advisor_name = "PRINCETON FUND ADVISORS, LLC"
managed_funds = get_managed_funds(advisor_name)

print("The funds managed by", advisor_name, "are")
for fund in managed_funds:
    print(fund)
```

The funds managed by PRINCETON FUND ADVISORS, LLC are: Princeton Long/Short Treasury Fund, Alpha Intelligent - Large Cap Growth ETF, Alpha Intelligent - Large Cap Value ETF, Deer Park Total Return Credit Fund, Princeton Premium Fund, Eagle MLP Strategy Fund

What is the total purchase sale for funds NORTHERN SMALL CAP CORE FUND, JNL/AMERICAN FUNDS GLOBAL GROWTH FUND, and SIIT ULTRA SHORT DURATION BOND FUND?

```
def get_total_purchase_sale(fund_names):
    total_purchase_sale = 0

    # Get all N-CEN reports
    reports = get_all_reports()
    for fund_name in fund_names:
        # Get the N-CEN report for the fund
        report = get_report(fund_name)
        # Fetch the corresponding block for the fund in the report
        block = fetch_block(report, fund_name)
        # Extract the purchase sale value from the block
        purchase_sale = extract_value(block, "purchase_sale")
        # Add the purchase sale value to the total
        total_purchase_sale += purchase_sale
    return total_purchase_sale

# List of funds
fund_names = ["NORTHERN SMALL CAP CORE FUND",
"JNL/AMERICAN FUNDS GLOBAL GROWTH FUND", "SIIT
ULTRA SHORT DURATION BOND FUND"]
# Calculate the total purchase sale for the funds
total_purchase_sale = get_total_purchase_sale(fund_names)
print("The total purchase sale for the funds is:", total_purchase_sale)
```

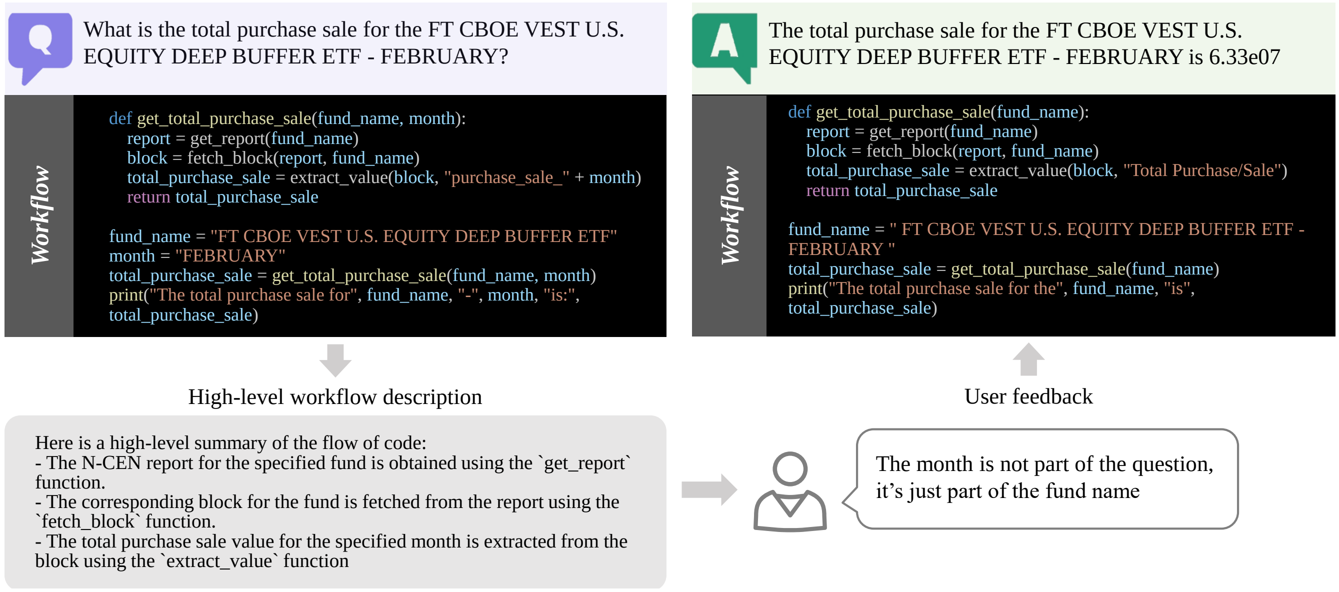
The total purchase sale for the funds is 1.66e09

**Figure 6: NCEN-QA-Hard: example questions, corresponding workflows and answers generated by FlowMind.**

fund name, not a time reference. For NCEN-QA-Intermediate and NCEN-QA-Hard, user feedback helped correct false assumptions regarding the information that could be extracted using the *extract* APIs (Section 4.2.3). For instance, FlowMind initially operated under the assumption that ‘gross commission ratio’ was a value that could be directly extracted, when it actually required computation. Another example was with the terms ‘purchase’ and ‘sale’, which FlowMind initially considered as separate values for extraction. However, they were part of a single term, "purchase sale", that needed to be extracted as a whole. FlowMind was able to correct its workflow based on user feedback, thus boosting its performance to 100% or near 100% across the datasets.

## 5 CONCLUSION

In conclusion, our work presents a significant leap in using LLMs for automatic workflow generation. By combining lecture prompt design, user feedback, and secure, grounded reasoning, FlowMind provides a reliable, adaptable, and efficient solution for handling



What is the total purchase sale for the FT CBOE VEST U.S. EQUITY DEEP BUFFER ETF - FEBRUARY?

```
def get_total_purchase_sale(fund_name, month):
    report = get_report(fund_name)
    block = fetch_block(report, fund_name)
    total_purchase_sale = extract_value(block, "purchase_sale_" + month)
    return total_purchase_sale

fund_name = "FT CBOE VEST U.S. EQUITY DEEP BUFFER ETF"
month = "FEBRUARY"
total_purchase_sale = get_total_purchase_sale(fund_name, month)
print("The total purchase sale for", fund_name, "-", month, "is:",
total_purchase_sale)
```

Here is a high-level summary of the flow of code:
- The N-CEN report for the specified fund is obtained using the `get_report` function.
- The corresponding block for the fund is fetched from the report using the `fetch_block` function.
- The total purchase sale value for the specified month is extracted from the block using the `extract_value` function

The month is not part of the question, it’s just part of the fund name

The total purchase sale for the FT CBOE VEST U.S. EQUITY DEEP BUFFER ETF - FEBRUARY is 6.33e07

```
def get_total_purchase_sale(fund_name):
    report = get_report(fund_name)
    block = fetch_block(report, fund_name)
    total_purchase_sale = extract_value(block, "Total Purchase/Sale")
    return total_purchase_sale

fund_name = " FT CBOE VEST U.S. EQUITY DEEP BUFFER ETF -
FEBRUARY "
total_purchase_sale = get_total_purchase_sale(fund_name)
print("The total purchase sale for the", fund_name, "is",
total_purchase_sale)
```

**Figure 7: Example of FlowMind correcting workflow given user feedback.**

spontaneous tasks with auto-genenated workflows. Our work opens new avenues for more widespread adoption of LLMs, particularly in industries where data security and the spontaneity of tasks are of paramount importance.

We also introduce a new finance dataset NCEN-QA, which serves as a robust benchmark platform for automatic workflow generation systems on N-CEN reports question-answering tasks about funds, thus providing a valuable resource for the broader research community.

In the future, it’s worth investigating crowdsourcing user feedback to refine workflows in FlowMind at scale, as well as life-long learning over past user-approved examples to evolve its performance over time. In addition, FlowMind can be expanded in the future to handle big libraries of APIs by retrieving the most relevant APIs for a given task given embedding similarity.

## ACKNOWLEDGMENTS

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates (“J.P. Morgan”) and is not a product of the Research Department of J.P. Morgan. J.P. Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

## REFERENCES

[1] 1984. Edgar. https://www.sec.gov/edgar.

[2] 2022. LangChain. https://github.com/langchain-ai/langchain.

[3] 2022. OpenAI API. https://platform.openai.com/docs/guides/embeddings.

[4] 2023. AutoGPT. https://github.com/Significant-Gravitas/Auto-GPT.

[5] 2023. Transformer Agent. https://huggingface.co/docs/transformers/main_classes/agent.

[6] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691* (2022).

[7] Max Bachmann. 2021. *maxbachmann/RapidFuzz: Release 1.8.0.* https://doi.org/10.5281/zenodo.5584996

[8] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. PAL: Program-aided Language Models. *ArXiv* abs/2211.10435 (2022).

[9] Lukas-Valentin Herm, Christian Janiesch, Alexander Helm, Florian Imgrund, Kevin Fuchs, Adrian Hofmann, and Axel Winkelmann. 2020. A Consolidated Framework for Implementing Robotic Process Automation Projects. In *Business Process Management*, Dirk Fahland, Chiara Ghidini, Jörg Becker, and Marlon Dumas (Eds.). Springer International Publishing, Cham, 471–488.

[10] Peter Hofmann, Caroline Samp, and Nils Urbach. 2020. Robotic process automation. *Electronic Markets* 30 (2020), 99–106. https://doi.org/10.1007/s12525-019-00365-8

[11] Toru Kobayashi, Kenichi Arai, Tetsuo Imai, Shigeaki Tanimoto, Hiroyuki Sato, and Atsushi Kanai. 2019. Communication Robot for Elderly Based on Robotic Process Automation. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2. 251–256. https://doi.org/10.1109/COMPSAC.2019.10215

[12] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 9493–9500.

[13] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210* (2023).

[14] Xiao-Yang Liu, Guoxuan Wang, and Daochen Zha. 2023. FinGPT: Democratizing Internet-scale Data for Financial Large Language Models. *arXiv preprint arXiv:2307.10485* (2023).

[15] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. WebGPT: Browser-assisted question-answering with human feedback. arXiv:2112.09332 [cs.CL]

[16] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*

(2022).

[17] Jayr Pereira, Robson Fidalgo, Roberto Lotufo, and Rodrigo Nogueira. 2023. Visconde: Multi-document QA with GPT-3 and Neural Reranking. In *European Conference on Information Retrieval*. Springer, 534–543.

[18] Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint arXiv:2201.11227* (2022).

[19] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[20] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[21] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083* (2023).

[22] M. Ratia, J. Myllärniemi, and N. Helander. 2018. Robotic Process Automation - Creating Value by Digitalizing Work in the Private Healthcare?. In *Proceedings of the 22nd International Academic Mindtrek Conference* (Tampere, Finland) *(Mindtrek '18)*. Association for Computing Machinery, New York, NY, USA, 222–227. https://doi.org/10.1145/3275116.3275129

[23] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633* (2021).

[24] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools.

[25] Sanjay Subramanian, Medhini Narasimhan, Kushal Khangaonkar, Kevin Yang, Arsha Nagrani, Cordelia Schmid, Andy Zeng, Trevor Darrell, and Dan Klein. 2023. Modular Visual Question Answering via Code Generation. arXiv:2306.05392 [cs.CL]

[26] Rehan Syed, Suriadi Suriadi, Michael Adams, Wasana Bandara, Sander J.J. Leemans, Chun Ouyang, Arthur H.M. ter Hofstede, Inge van de Weerd, Moe Thandar Wynn, and Hajo A. Reijers. 2020. Robotic Process Automation: Contemporary themes and challenges. *Computers in Industry* 115 (2020), 103162. https://doi.org/10.1016/j.compind.2019.103162

[27] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*. 1–7.

[28] Wil M. P. van der Aalst, Martin Bichler, and Armin Heinzl. 2018. Robotic Process Automation. *Business & Information Systems Engineering* 60 (2018), 269–272. https://doi.org/10.1007/s12599-018-0542-4

[29] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. 2023. Chatgpt for robotics: Design principles and model abilities. *Microsoft Auton. Syst. Robot. Res* 2 (2023), 20.

[30] Alice Saldanha Villar and Nawaz Khan. 2021. Robotic process automation in banking industry: a case study on Deutsche Bank. *Journal of Banking and Financial Technology* 5, 1 (2021), 71–86.

[31] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A Large Language Model for Finance. arXiv:2303.17564 [cs.LG]

[32] Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. FinGPT: Open-Source Financial Large Language Models. *arXiv preprint arXiv:2306.06031* (2023).

[33] Boyu Zhang, Hongyang Yang, and Xiao-Yang Liu. 2023. Instruct-FinGPT: Financial Sentiment Analysis by Instruction Tuning of General-Purpose Large Language Models. *arXiv preprint arXiv:2306.12659* (2023).