Imperial College London

Department of Earth Science and Engineering

MSc in Environmental Data Science and Machine Learning

Independent Research Project

Final Report

Exposure and Performance Analysis of Investment Funds using Large Language Models (LLMs)

by

Sara Lakatos

Email: sara.lakatos23@imperial.ac.uk

GitHub username: acse-sl4623

Repository: https://github.com/ese-msc-2023/irp-sl4623

Supervisors:

Rhodri Nelson

Marijan Beg

August 2024

Table of Contents

1. Abstract ... 3

2. Introduction ... 3

3. Methodology ... 3

3.1. Data Extraction ... 4

3.2. Data Processing ... 4

3.2.1. Tabular Information (Structured) ... 5

3.2.2. Fund Information (Unstructured) ... 5

3.3. Knowledge Base ... 5

3.3.1. Vector Embeddings ... 5

3.3.2. Chunking Strategy ... 6

3.3.3. Retrieval ... 7

3.4. Finbot Final RAG Implementation ... 7

4. Results ... 8

4.1. General Questions ... 9

4.2. Description Questions ... 10

4.2.1. Performance ... 10

4.2.2. Exposure ... 11

4.3. Peer Group Comparisons ... 12

4.4. Identification of performance outliers within Peer Groups ... 14

4.5. Evaluation of Peer Groups ... 14

5. Discussion ... 15

6. Conclusion ... 16

7. Appendix ... 16

7.1 Tabular Information Extraction and Processing Pipeline: ... 16

7.2 Tables Preprocessing Challenges: ... 17

7.3. React Agent Prompt Template ... 18

7.4. General Questions (For Evaluation) ... 19

7.5.Route Chain Implementation: ... 19

References*:* ... 19

1. Abstract

Portfolio analysis is conducted by several business functions across corporate finance for investment management and oversight. Automating the process of portfolio analysis of individual funds would help streamline business processes and facilitate top-level cross-fund analysis. In this paper we focus specifically on individual fund and peer analysis. Through LangChain, we built a Python package called finbot which uses the Llama 3 Large Language Model (LLM) with an expanded knowledge base of investment fund factsheets through a multitude of Retrieval Augmented Generation (RAG) chains to generate insightful descriptive narratives of this particular universe of funds and identify performance outliers within customized peer groups for comparative analysis. The aim of this package is to explore the capabilities of a pre-trained LLM to analyse its knowledge base such that it generates new insights within the domain of the knowledge base for descriptive and comparative purposes.

2. Introduction

Fund managers often have numerous funds to systematically monitor and ensure regulatory compliance for, generally through internal or external reporting. Analysis conducted on the investment fund universe requires systematic review of structured and unstructured fund data on performance and investment allocation to generate insightful observations.

Given this problem statement, we explore the feasibility of using a Large Language Model (LLM) with an expanded knowledge base of a universe of investment funds to converse and draw insights from this universe. LLMs have so far showed great promise in improving the efficiency and effectiveness of information retrieval and knowledge synthesis[1, 2]. Some financial applications of particular interest where LLMs have been deployed include sentiment analysis and forecasting[3]. An example used prompt engineering on the GPT-4 model to conduct fundamental analysis on historical company financial statements, which demonstrated remarkable promise in using such models to mimic, albeit not entirely replace, human judgement[4].

The objective of this paper is to develop an investment fund chatbot, that can not only describe and retrieve information from a domain-specific knowledge base, but also conduct cross-sectional calculations between the documents, therefore generating information not directly available in the knowledge base to draw observations from. Such quantitative reasoning passed through the context of an LLM in combination with prompt engineering has been previously used across financial and scientific applications[5], however not applied to the investment fund universe.

3. Methodology

To implement the objective, the example implementation explored makes use of investment fund factsheets, a regulatory required report available in portable document format(PDF) consisting of both structured and unstructured data.

The particularity of this investment fund dataset, is that to answer meaningful questions from this universe such as “What was the historical performance of the fund that outperformed its peers in the last year?” or “What is the sector allocation of a fund invested in the UK which has performed the worst in the last quarter?”, one has to first define a peer group for any or all funds, conduct performance or exposure comparison and then, finally describe the fund attributes in isolation or in relation to its peers.

Popular methods of cross-sectional comparisons include: performance metric comparison and style analysis, a method to investigate a fund's asset allocation through the MorningStar Style boxes within a MorningStar defined peer group. MorningStar, is an investment research company that represents the industry standard in peer grouping for investment funds. It defines a "peer" as a comparable investment fund with similar investment objective and strategy[6].If we had access to the MorningStar API, we could simply label each investment fund based on its MorningStar Category and Style Box and then use the percentile ranking methods of van Dijk et al. (2011)[7] with the arithmetic reasoning methods passed to the context of an LLM in .Huang et al. (2022)[8].In the absence of access to the MorningStar API, we consider a novel approach in using the embedding distance of points in a vector store to dynamically create peer groups for every fund in the knowledge base.

The most common metrics used for peer evaluation are historical performance and holdings data. Factsheets disclose both, performance across 5 periods and holdings for a given date. Both disclosures are used to generate comparative and descriptive narratives. To facilitate comparisons, we implement a simple percentile rank for each peer group to identify any 'outliers' as van Dijk et al. (2011)[7]]. Percentile ranking is a simple, consistent metric that allows for intuitive comparison, however is sensitive to the sample size and distribution and may not generalize well across time. As funds rebalance their positions, statements on the percentile rank at one point in time may not hold in subsequent periods. This metric also lacks the context of the investment strategy, but limiting the comparison set to its peers mitigates this, albeit not eliminating it entirely, as investment funds within a peer group can vary in the execution of their investment strategy[7].

In the following subsections we describe the steps taken to create the finbot package which represents a Retrieval Augmented Generation (RAG) chain that retrieves information from the factsheets embedded and stored in a vector store to answer questions on this knowledge base through a Large Language Model (LLM) based on the Meta Llama 3 model. By using RAG chains, we implement the descriptive and analytical capabilities for fund assessment through prompt engineering, defining custom prompt templates for the LLM per query type and conduct context management, manipulating the information from the relevant factsheets retrieved for the query that get passed to the LLM.

3.1. Data Extraction

For data extraction we use the BeautifulSoup Python package as a web scraping tool to download the investment fund factsheets from Trustnet, a 3rd party site, that centralizes these reports from across the different asset managers websites from their first quarter reporting (data as of March 2024). However not all asset managers have opted in for this scheme, so peer groups as defined by MorningStar are likely incomplete. This implies that not every fund may have a complete set of peers by industry standards, which is already a limitation of the dataset. 804 PDFs were downloaded from Trustnet in a standardized template format.

3.2. Data Processing

Each investment fund factsheet contains both structured and unstructured data divided into 3 general categories: fund information (key identifiers), investment objective, performance and allocation tables.

3.2.1. Tabular Information (Structured)

Each factsheet must have two tables for historical performance and at least one investment allocation table, exposure across assets, sector, region and holdings. For data extraction and processing we use a combination of Optical Character Recognition (OCR) and specialized PDF Python packages to extract the tables as dictionaries. This dataset presented unique challenges in terms of document processing and information extraction due to its varying sizes and layouts, and variety of tabular information (see Appendix). Given LLMs are trained to process textual data, the dictionary tables are summarized through string concatenation to be embedded into the vector store.

At this point, we consider the scope of finbot, to conduct cross-fund comparisons, and realize the numerical figures extracted need to be used for computation. As such, there is a need to store the float values, separately to the text used to define the content uploaded to the vector store. As such, the extracted dictionaries are also stored as associated metadata for each table individually. By linking the metadata to the points in the vector store, numerical data can be retrieved alongside the text content, allowing for "real-time" retrieval and manipulation.

3.2.2. Fund Information (Unstructured)

The fund information consists of the investment objective and fund identifiers such as fund name and asset class. Given the objective, these are the most relevant attributes, available across all factsheets, the remaining attributes are disregarded. The processing methodology used for the unstructured information uses regex pattern matching, if this fails, the open-source Gemma LLM model from Ollama library is used, resulting in successful processing of 89% of investment objectives and 100% fund names and asset classes (see Appendix). Similar to the structured data, the fund attributes are concatenated to the investment objective string to form complete descriptive chunks for vector store upload and for metadata storage.

3.3. Knowledge Base

The vector store chosen for the processed factsheet content is Qdrant due to its payload flexibility, which permits storage of metadata alongside its vector embeddings, but also permits filtered retrieval of "points", the records of embedded text (vectors), based on the metadata. Qdrant uses Approximate Nearest Neighbours (ANN) algorithms to retrieve the vectors that best match the embedded query through a distance metric such as cosine similarity, the preferred distance metric for representational learning applications and our choice, henceforth referred to as a "similarity score".

At this stage, the processed data is sectioned into fund information and tables with their associated metadata. The next step is to define how the processed data should be embedded and uploaded.

3.3.1. Vector Embeddings

For embedding, the Sentence Transformers library has shown superior performance in embedding tabular data[8], however given the intended use of numerical data (section 3.2.1), an embedding model specialized in textual coherence was preferred over one with enhanced numerical tokenisation.

As such, we chose the Masked and Permuted Pre-training (MPNet) model from Sentence Transformers available through the HuggingFace Transformers library. MPNet outperforms masked language modelling (MLM) used in BERT models and permuted language modelling (PLM) used in XLNet models, by building on both approaches and therefore capturing local and global dependencies in textual information, allowing for enhanced contextual understanding [10]. The HuggingFace Transformers wrapper also allows for easy integration with Qdrant and LangChain for optimized processing.

3.3.2. Chunking Strategy

Given the processed sectioned data we could have implemented the chunking strategy of Jimeno Yepes et al., 2024 applied to financial reports which shows how an element-wise approach guided by the inherent structure of the documents enhances retrieval for Retrieval Augmented Generation (RAG)[9]. This method was explored in preliminary implementations of the finbot, where each "chunk" represented a section of the factsheet, however it would likely not have been appropriate to generate peer groups.

Based on the MorningStar Category classification definition[6], peer groups are generated through a combination of their investment objective and holdings to reflect their actual investment style, not just their stated investment objectives. The classification is based off Morningstar research that found, the investment objective listed in a fund's prospectus often did not adequately explain how the fund actually invested[6]. As such, for the purposes of peer grouping, given Qdrant retrieves "points" relevant to the query based on their "similarity score", there's a need to embed the entirety of the investment fund factsheet which includes the investment objective and holdings data to generate peer groups.

Peer groups should refer to funds that follow the same investment style, an implied necessity of this requirement becomes that they should have, at the very least, a comparable investment universe, the opportunity set of investments a fund manager is allowed to choose from within the constraints of the fund's investment objective. For example, a US Technology fund would be constrained to the IT companies within the US. In the absence of MorningStar Style boxes across the standardized factsheets, we consider the fund's asset class, group of financial investment that have similar characteristics, as a minimal indicator of investment universe similarity.

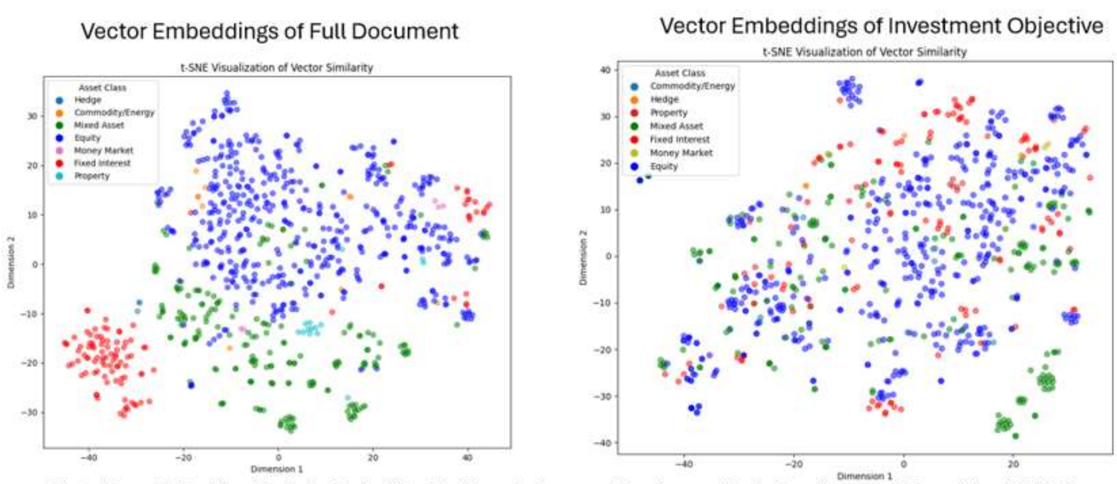


Fig 1. Vector Embedding Similarity: Embedding Model used: Sentence Transformers Masked and Permuted Pre-training (MPNet). The t-SNE (t-distributed Stochastic Neighbor Embedding) plots show the embeddings of the entire factsheet (LHS) for all funds and just the investment objective embeddings (RHS). The scope is to represent high-dimensional data (768 is the dimension of the vectors using the embedding model) in a 2D plane. The x and y axis do not have any interpretable meaning, the numerical values are abstractions such that the proximity of the embeddings in the higher-dimension are represented 'close' to each other in a lower dimension.

Fig. 1, motivate the choice of embedding the entirety of the factsheet by comparing the asset class distribution of vector embeddings of each document in its entirety to the vector embeddings of just their investment objective. Using the entire document, there's a clearer separation between the asset classes, notably Fixed Interest funds are isolated from Equity funds, as expected since these funds have completely disjoint investment universes (investing in bonds and stocks respectively). Similarly, Property funds, with a unique investment universe concentrated in real estate also stand out as their own tight cluster. Mixed Asset funds, as the name infers, allow for both equity and fixed income investments as such their scatter in between the Fixed Interest and Equity fund embeddings seems reasonable. On solely the investment objective side, there is no clear asset class separation, but tighter overlaps between clusters. The looser scattered embeddings, therefore support the observations from MorningStar that the investment objective, on its own, is not sufficient to provide a clear distinction between the investment styles, hence its' "chunk" would not be appropriate to define a peer group at retrieval

Based on Fig. 1 the investment universe similarity appears reflected in the embeddings of the factsheets. As such, the implemented knowledge base in finbot uses a document based chunking strategy. Hence the "points" retrieved by Qdrant in the RAG chain will represent entire factsheets.

3.3.3. Retrieval

For "point" (factsheet) retrieval, finbot uses 3 types of retrievers:

a) Fund Name Filtered Retriever: A limitation of the embedding model in extracting the correct factsheet when using the fund name due to the uniqueness of fund names as words was observed during implementation. In the factsheet dataset there are multiple funds that correspond to the same asset manager such as Fidelity (expl. Fidelity Multi Asset Allocator Defensive, Fidelity Multi Asset Allocator Growth). There are also different fund structures that have similar fund names such as L&G Multi-Index Income 6 and L&G Multi-Index Income 5. The string similarity, translates to embedding similarity given the tokenisation of each individual word which posed a challenge for accurate retrieval. This was overcome by incorporating query parsing as a separate Retrieval Augmented Generation (RAG) chain which extracts the fund name from the query then prior to retrieval, finds the best fund name match from the knowledge base, that is passed as a parameter to the retriever (possible through metadata).

b) Number of Points Filtered Retriever: For queries that do not include fund name, to avoid passing irrelevant or insufficient information to the context, another RAG chain is invoked when defining the retriever to infer the number of funds queried. The exact number of funds is then passed as a parameter to the retriever, knowing that each point corresponds to a complete investment fund factsheet.

c) Similarity Score Threshold Retriever: For peer group based comparative analysis, similarity score based retrieval was implemented with 80% threshold (i.e. only funds with similarity score above 80% are retrieved). This filtering method utilizes the embedding observations in Fig.1. The threshold was set with an intent to identify at least the two most similar funds to allow for meaningful computations of averages and percentile ranks.

3.4. Finbot Final RAG Implementation

Given retrieval of peer groups is different from general queries, there was a need to split one RAG chain by similarity score and other filtered retrieval. However, descriptions are different from comparisons, so to generate meaningful insights, description prompts had to be different from comparison prompts. However, allocation and performance data is at a given time and a time series respectively so comparisons had to be implemented differently, also one does not describe fund returns in the same manner as exposure, so separate prompts were required for description. Finally, the finbot should also be able to converse on and off its domain knowledge and answer basic information requests.

Hence, the final finbot implementation had allow for multiple response types, implying the need for multiple template prompts to guide the LLM's response, these in turn had to be routed based on the query. Hence, we implemented a similar approach to PaperQA in Lála et al. (2023) [11] which used a customized React Agent to select from a tool set of RAG chains. A React Agent is a technique which enables LLMs to do reasoning through a specific (React) prompt inducing the LLM to iteratively choose from a set of "tools" (actions) until they collect enough information to answer the query[12].

Finbot is also a React Agent with a customized Chain-of-Thought (CoT) reaction (React) prompt (see Appendix), notably to limit the number of iterations and guide the agent's choice among 10 tools The tools implemented for Finbot are described in Fig 2. Each RAG chain tool has a customized prompt template, exposure is described relative to its investment objective and performance is described across time and relative to peers. Given the chunking strategy, not to overload the context, each RAG chain tool also has its own context building function.

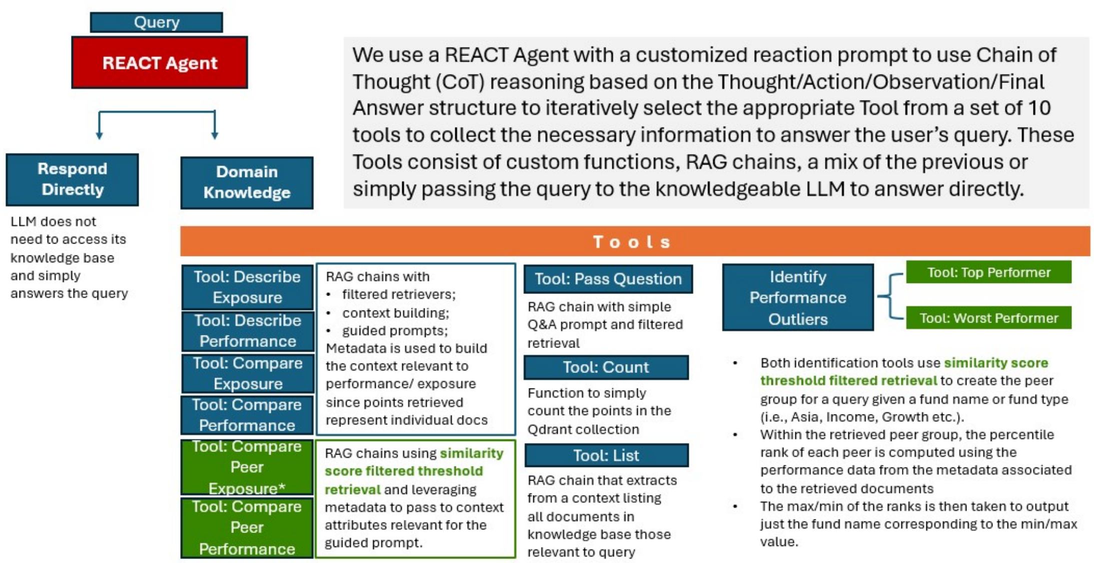


Fig 2. Finbot Final Chain - Tools. The Compare Peer Exposure was ultimately removed from the final implementation.

The LLM model chosen to underpin the entirety of the chain is Llama 3 with 70 billion parameters, chosen for its scale within the constraints of resource availability in the Ollama open-resource library and Imperial's server. A large model was preferred given the implementation of CoT reasoning which is limited on small models and because arithmetic reasoning has been observed to improve with model scaling[13]. The same model underpins every tool as RAG chain.

4. Results

To evaluate the Finbot (agent chain), we need to evaluate its responses on different sets of questions across the following categories: general, description, comparison, identification. We are interested in testing information retrieval, routing (Agent's ability to select the appropriate tool) and the quality of responses given the customised prompt templates. Evaluations across these categories are done using LangSmith on custom built question datasets.

In the absence of ground truth responses to queries we implement a capture and run chain function that is iteratively called when creating the set of questions for evaluation for each category in order to capture the retrieved documents for any given query in the dataset. The metadata of the retrieved documents with the corresponding context building function for the category is used to build the reference output.

The evaluation metric used is the in-built LangSmith metric for contextual accuracy that measures correctness of response to the reference output. To ensure an unbiased evaluation, a different Llama 3 model is used, with 8 bits, which is less computationally expensive allowing for faster evaluation and greater scalability.

To facilitate testing, we leverage the following “benchmark” chains to highlight certain features of the agent chain using just the contextual accuracy metric from LangSmith. All these benchmark chains use the same underlying Llama 3 model as the agent chain.

- simple chain: investment fund ignorant chain that cannot access the investment fund knowledge base. Comparisons to this chain indicate if the agent chain shows more domain knowledge.

- domain chain: knowledgeable chain that retrieves from the knowledge base (in fact it’s the underlying chain of the “Pass Question” tool), but is not prompt engineered. Comparisons to this chain show both differences in response quality and routing effectiveness.

- a route chain: similar chain to the agent chain, however it routes to underlying chains based on two routing functions and two query classification chains (details in Appendix) Comparisons to this chain indicate if the Agent’s tool calling is more effective than simple routing.

Subsequent observations on the results are based on inspection of the referenced datasets, please refer to links available in the README in the GitHub repository.

4.1. General Questions

The General Questions dataset consists of 69 questions regarding information requests of the investment objective, performance and allocation information requests. On this dataset, the Agent is expected to call the “Pass Question to Domain Knowledge” tool, however there are 11 where the “List” tool is expected and 1 for the “Count” tool.

As reference output, the general context building function is used to merge the document content of the retrieved points from the vector store.

Experiment Name	Cot Contextual Accuracy		P50 Latency	P99 Latency	Run Count	Error Rate
agent_chain_gen_un...	CORRECT (40)	INCORRECT (15	73.10s	364.08s	69	4%
route_chain_gen_und...	INCORRECT (39)	CORRECT (15	22.88s	39.15s	69	0%
domain_chain_gen_u...	CORRECT (52)	INCORRECT (8)	22.53s	72.06s	69	0%
simple_chain_gen_un...	INCORRECT (28)	CORRECT (23	14.31s	98.37s	69	0%

Fig 3. Reference: LangSmith, Project: Finbot, Dataset: General Questions (all types: Docs KB) **CoT Contextual Accuracy**: Chain of thought question answering evaluator, which grades answers to questions using chain of thought 'reasoning' given a reference output. **P50 Latency**: median processing time of the questions in the dataset; **P99 Latency**: 99% of the questions are processed under this time; **Run Count**: number of invocations of the chain (in our case equal to the number of questions in dataset); **Error Rate**: Percentage of runs that resulted in errors.

<u>Contextual Accuracy</u>: The agent chain has higher contextual accuracy than the simple chain given access to its knowledge base, whereas the simple chain offers seemingly reasonable responses, however not necessarily truthful. By regressing, comparing the agent chain's responses to the simple chain's, LangSmith provides a 17% improvement in response quality.

<u>Routing Efficiency</u>: The agent chain has higher accuracy to the route chain indicating that the Agent performs better than the simple functions, but not as well as the domain chain.

By inspecting the incorrect runs, for generic queries (no fund name specified), the Agent correctly utilizes the "Get Relevant Documents" tool to identify a fund within that asset class and then uses the "Pass Question to Domain Knowledge" tool to answer the query. When this occurs the retrieved document may not match exactly the document passed to the reference output therefore the answer gets classified as incorrect. As such for these queries the evaluation results don't necessarily imply that the agent responded incorrectly, rather it's a limitation of the evaluation process.

For some allocation/performance questions the Agent does not invoke the "Pass Question to Domain Knowledge" tool, preferring one of the Description tools, resulting in a very detailed response (per React prompt). These prompts are tailored to generate a specific narrative, which may not contain the correct response.

The low accuracy of the route chain is due to an unfiltered retrieval for this question category. Hence, the accuracy score for route chain also partly encompasses the retrieval inefficiency based on just the vector embeddings discussed in section 3.3.3.

The 4% error rate is on a query to list "all" funds where the context is overloaded as all factsheets are passed to it.

4.2. Description Questions

The Description dataset consists of 100 description questions designed to prompt the agent to "describe" (using various synonyms of the word to summarize the historical performance or investment allocation for a given fund.

4.2.1. Performance

For the performance description dataset, the context building function for the performance description chain is used to generate the reference output.

The expectation for this dataset is that the “Provide Description of Performance” tool is called.

Experiment Name	Cot Contextual Accuracy	P50 Latency	P99 Latency	Run Count	Error Rate
agent_chain_gen_per...	CORRECT (99) INCORRECT (1)	123.75s	174.26s	100	0%
route_chain_gen_per...	CORRECT (92) INCORRECT (5)	43.52s	68.17s	100	0%
domain_chain_gen_p...	CORRECT (77) INCORRECT (3)	47.82s	79.57s	91	0%
simple_chain_gen_pe...	CORRECT (49) INCORRECT (40	42.18s	68.16s	100	0%

Fig 4. Reference: LangSmith, Project: Finbot, Dataset: Description Questions – Performance (100 Sample) **CoT Contextual Accuracy**: Chain of thought question answering evaluator, which grades answers to questions using chain of thought ‘reasoning’ given a reference output. **P50 Latency**: median processing time of the questions in the dataset; **P99 Latency**: 99% of the questions are processed under this time; **Run Count**: number of invocations of the chain (in our case equal to the number of questions in dataset); **Error Rate**: Percentage of runs that resulted in errors.

Contextual Accuracy: The agent chain has higher contextual accuracy than the simple chain. The LangSmith regression results show a 40% improvement in the agent’s response.

Prompt Engineering: By regressing the agent chain onto the domain chain, LangSmith estimates only a 3% improvement in response. However, through a side-by-side comparison of the domain chain (using a general prompt) and the agent chain (using a customized description prompt), we observe the difference in quality of responses. Despite both being contextually accurate, the agent’s response is a narrative with complementary computational proof of its observations, whereas the domain chain responses are mere reiterations of the tabular data.

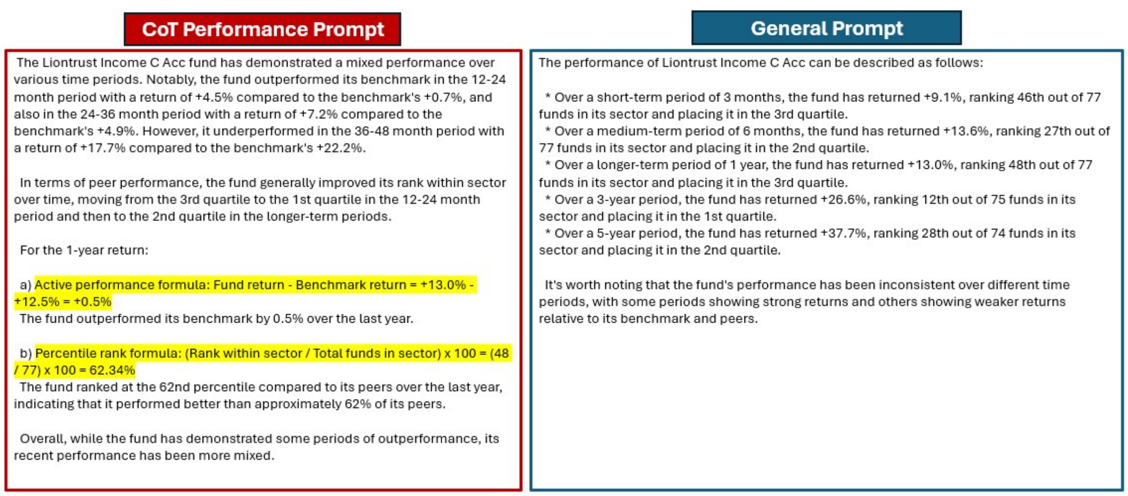


Fig 5. Reference: Example from LangSmith: Project: Finbot, Dataset: Description Questions – Performance (100 Sample) (agent and domain chains)

Routing Efficiency: Both agent and route chains have high contextual accuracy for this dataset indicating effective routing to the performance description tool and chain respectively.

4.2.2. Exposure

For the exposure description dataset, the context building function for the exposure description chain is used to generate the reference output.

The expectation for this dataset is that the “Provide Description of Exposure” tool is called.

Experiment Name	Cot Contextual Accuracy	P50 Latency	P99 Latency	Run Count	Error Rate
agent_chain_gen_ex...	CORRECT (78) INCORRECT (19	91.19s	116.58s	100	0%
route_chain_gen_exp...	CORRECT (58) INCORRECT (38	29.28s	51.91s	100	0%
domain_chain_gen_e...	CORRECT (85) INCORRECT (14	27.49s	51.71s	100	0%
simple_chain_gen_ex...	INCORRECT (65) CORRECT (19	26.18s	50.54s	100	0%

Fig 6**.** Reference: LangSmith, Project: Finbot, Dataset: Description Questions – Exposure (100 Sample) **CoT Contextual Accuracy**: Chain of thought question answering evaluator, which grades answers to questions using chain of thought ‘reasoning’ given a reference output. **P50 Latency**: median processing time of the questions in the dataset; **P99 Latency**: 99% of the questions are processed under this time; **Run Count**: number of invocations of the chain (in our case equal to the number of questions in dataset); **Error Rate**: Percentage of runs that resulted in errors.

Contextual Accuracy: The agent chain has high contextual accuracy relative to the simple chain. The LangSmith regression results show a 51% improvement in the agent’s response. The exposure description context building function and prompt leverage the investment objective for a better narrative. As this context function generates the reference and the questions only asks for allocation, the simple chain is disadvantaged by design given the evaluation methodology.

Prompt Engineering: Despite agent’s advantage, regressing the agent chain onto the domain chain only generates a 9% improvement in terms of contextual accuracy. From the response review in Fig 7, the same difference in narrative structure is highlighted that is consistent across the dataset. The exposure prompt contextualizes the tabular information in terms of its investment objective which allows the LLM to draw insights such as whether this exposure is expected or not.

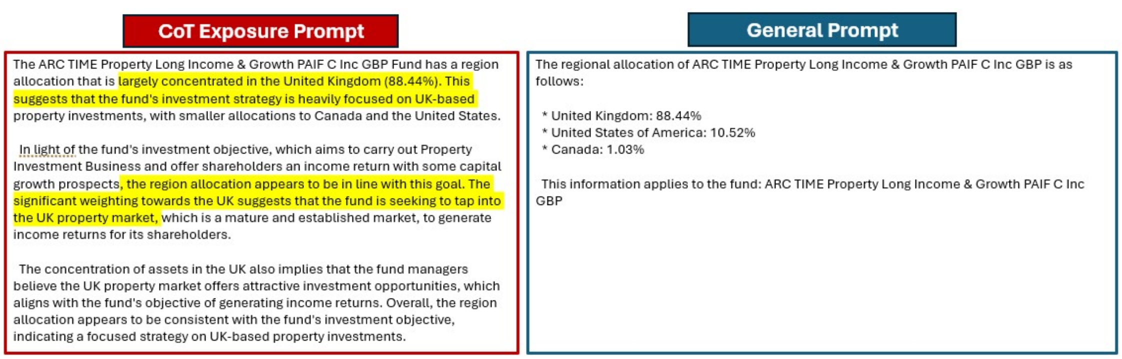


Fig 7 Reference: Example from LangSmith: Project: Finbot, Dataset: Description Questions – Exposure (100 Sample) (agent and domain chains)

Routing Efficiency: The agent chain has higher accuracy than the route chain, upon inspection, route chain tends to misclassify the query as a “specific information request” instead of a “describe” task through the first task classification routing function (see Appendix), whereas the agent calls the expected tool the majority of the time.

4.3. Peer Group Comparisons

For the peer comparison dataset, we use the context building function for the peer performance description chain to generate the reference output.

The expectation for this dataset is that the “Compare Peer Performance” tool is called.

Experiment Name	Cot Contextual Accuracy		P50 Latency	P99 Latency	Run Count	Error Rate
agent_chain_peer_co...	CORRECT (57)	INCORRECT (39	84.82s	111.51s	100	1%
simple_chain_peer_c...	CORRECT (67)	INCORRECT (28	53.44s	97.08s	100	0%
domain_chain_peer_...	CORRECT (74)	INCORRECT (23	50.48s	84.62s	100	0%
route_chain_peer_co...	CORRECT (53)	INCORRECT (44	42.68s	54.21s	100	0%

Fig 8. Reference: LangSmith, Project: Finbot, Dataset: Comparison Questions – Performance (100 Sample) **CoT Contextual Accuracy**: Chain of thought question answering evaluator, which grades answers to questions using chain of thought ‘reasoning’ given a reference output. **P50 Latency**: median processing time of the questions in the dataset; **P99 Latency**: 99% of the questions are processed under this time; **Run Count**: number of invocations of the chain (in our case equal to the number of questions in dataset); **Error Rate**: Percentage of runs that resulted in errors.

Contextual Accuracy: The agent chain has the lowest contextual accuracy. This is driven by a mismatch in the expected metrics passed to the reference output, computed from the factsheets, rather than extracted (average, percentile rank) and the ones used in the agent chain’s response. The domain and simple agents are evaluated higher by the evaluation LLM as they provided extracted and general responses of potential peer analysis despite also not incorporating the expected metrics.

Prompt Engineering: By comparing the prompts in Fig 9, we observe the main difference is that the comparison prompt uses information inferred from the knowledge base (peer groups, percentile rank and average return), rather than information extracted directly from the factsheet (by domain chain). This prompt changes the sort of peer analysis provided as it conducts an analysis of the peers determined within the knowledge base, rather than using the external information.

CoT Comparison Prompt

Based on the performance metrics, the Royal London Short Duration Gilts M Inc fund has outperformed its peers over a 1-year period. With an average 1-year return of 1.05%, the fund ranks at the 100th percentile among its peer group, indicating exceptional performance.

In comparison, Scottish Widows Gilt A Acc and iShares Index Linked Gilt Index (UK) D Acc, which are part of the same peer group, have underperformed the Royal London fund over the same period. The fund's strong returns can be attributed to its effective investment strategy and management team.

Given its exceptional performance, investors seeking exposure to the gilt market may consider allocating a portion of their portfolio to the Royal London Short Duration Gilts M Inc fund. However, it is essential to evaluate the fund's performance over multiple time periods and assess its risk profile before making an investment decision.

General Prompt

The performance of Royal London Short Duration Gilts M Inc compared to its peers is as follows:

Short-term performance (3m and 6m):
The fund has outperformed its benchmark in the short term, with a 3-month return of +0.7% vs -0.2% for the benchmark, ranking 3rd out of 33 funds in the sector. Over 6 months, the fund returned +1.7%, underperforming the benchmark (+2.0%) and ranking 22nd out of 33 funds.

Medium-term performance (1yr):
The fund has outperformed its benchmark over 1 year, with a return of +4.7% vs +3.1% for the benchmark, ranking 2nd out of 33 funds in the sector.

Long-term performance (3yrs and 5yrs):
Over 3 years, the fund has significantly outperformed its benchmark, with a return of +0.0% vs -21.9% for the benchmark, ranking 2nd out of 30 funds. Similarly, over 5 years, the fund has returned +1.0%, outperforming the benchmark (-19.8%) and ranking 1st out of 28 funds.

Discrete performance:
The fund's discrete performance over various periods also shows a strong track record, with the fund consistently outperforming its benchmark and ranking in the top quartile (1st or 2nd) across most time periods.

Overall, Royal London Short Duration Gilts M Inc has demonstrated a strong performance relative to its peers and benchmark across various time horizons.

Fig 9. Reference: Example from LangSmith: Project: Finbot, Dataset: Comparison Questions – Performance (100 Sample) (agent and domain chains)

Routing Efficiency: Despite an almost even accuracy score with the route chain, the agent chain calls the expected tool the vast majority of the time, as well as the route chain, hence the relatively low accuracy score on this dataset does not imply better routing.

4.4. Identification of performance outliers within Peer Groups

The outlier identification process is evaluated by checking for a specific tool calling sequence on a dataset of size 10, with questions prompting to describe either the sector or historical allocation of the best/worst performer within a "peer group". We apply discretion in generating these peer group questions, using broad investment universe categories such as "US Technology", "Growth".

The sequence we expect the agent chain to follow is to choose one of "Identify the Top/Worst Performer" and then to "Provide Description of Exposure/Performance" tool. This logic is a specific instruction in the React Prompt. Since these tools are not chains, the process of building the reference output is more involved (details in Appendix), but ultimately consists of the peer group and the performance or allocation table per query.

1. Evaluation on performance prompts that identify best/worst performers within a peer group

Prompts are phrased as: "Describe historical performance of the fund with the best/ worst performance in the {*peer group*}

Experiment Name	Cot Contextual Accuracy	P50 Latency	P99 Latency	Run Count	Error Rate
agent_chain_id_perf_...	CORRECT (7) INCORRECT (3)	147.59s	173.86s	10	0%
domain_chain_id_per...	INCORRECT (8) CORRECT (1)	37.37s	56.13s	10	0%

Reference: LangSmith, Project: Finbot, Dataset: Identification Questions – Performance; **CoT Contextual Accuracy**: Chain of thought question answering evaluator, which grades answers to questions using chain of thought 'reasoning' given a reference output. **P50 Latency**: median processing time of the questions in the dataset; **P99 Latency**: 99% of the questions are processed under this time; **Run Count**: number of invocations of the chain (in our case equal to the number of questions in dataset); **Error Rate**: Percentage of runs that resulted in errors

2. Evaluation on exposure prompts that identify best/worst performers within a peer group

Prompts are phrased as: "Describe the sector allocation of the fund with the best/worst performance in the {*peer group*}

Experiment Name	Cot Contextual Accuracy	P50 Latency	P99 Latency	Run Count	Error Rate
agent_chain_id_exp -...	CORRECT (6) INCORRECT (4)	113.16s	133.26s	10	0%
domain_chain_id_exp...	INCORRECT (10)	39.27s	47.71s	10	0%

Reference: LangSmith, Project: Finbot, Dataset: Identification Questions – Exposure; **CoT Contextual Accuracy**: Chain of thought question answering evaluator, which grades answers to questions using chain of thought 'reasoning' given a reference output. **P50 Latency**: median processing time of the questions in the dataset; **P99 Latency**: 99% of the questions are processed under this time; **Run Count**: number of invocations of the chain (in our case equal to the number of questions in dataset); **Error Rate**: Percentage of runs that resulted in errors

Fig 10. LangSmith Evaluations of Identification Questions

The results in Fig. 10 show high contextual accuracy for the agent chain relative to the domain chain as the domain chain is not capable of identifying peers, by inspecting the traces of each run for both datasets, we observe the expected sequence is followed.

4.5. Evaluation of Peer Groups

To evaluate the similarity score based retrieval, we conduct an asset class evaluation of the retrieved peer groups for reasons as in section 3.3.3., to review the effectiveness of this method for peer group generation. This evaluation method is inspired by Qdrant's method to measure retrieval quality of its Approximate Nearest Neighbors (ANN) algorithms by identifying the relevant documents in the top k results[14]. The evaluation is presented in Fig 11.

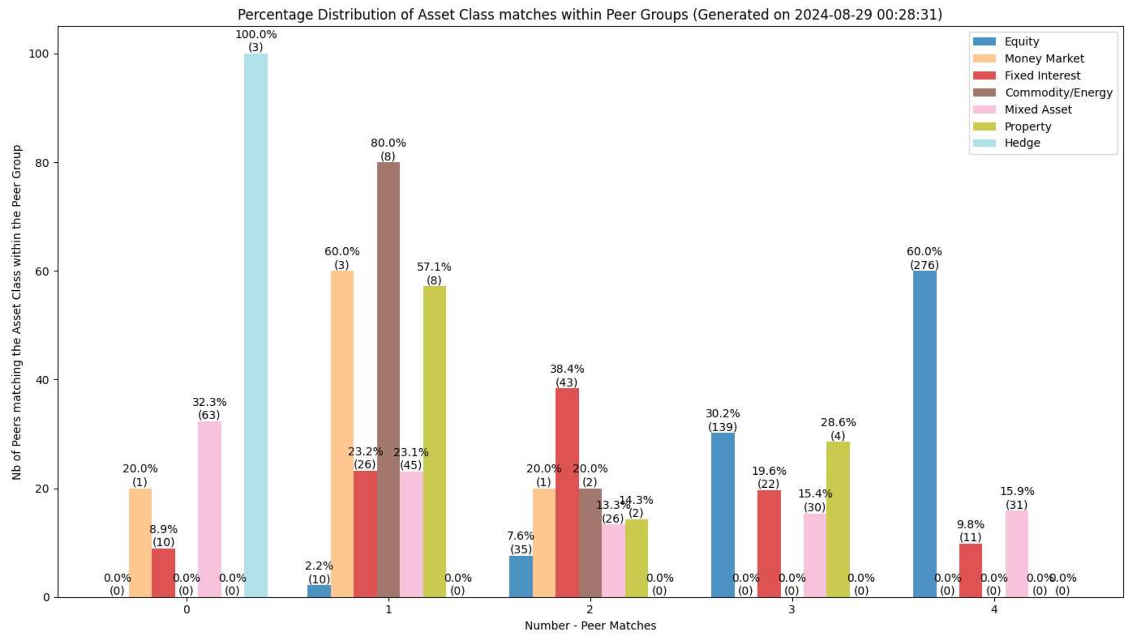


Fig. 11 Reference: Generated (with ChatGPT) through the extract_peers.py script in the GitHub repository under peer analysis

For each fund in the knowledge base we apply filtered similarity score retrieval (80%), store the asset class attributes of the identified peers and match it to the asset class of the given fund. We proceed to count the number of matches within a peer group (with ~4 peers) which are then aggregated across the peer groups. The aggregate numbers represent the number of asset class matches of a given asset class category within a peer group across all the peer groups.

For 60% of the Equity funds in the knowledge base, all the peers retrieved were Equity funds, whereas for another 30%, only 3 peers retrieved were Equity funds. For Fixed Interest and Mixed Asset there is a more even distribution of peer group retrieval, implying that for the majority of these fund types, the peer groups contained at least 1 peer of the same asset class, however for 9% and 32% of funds respectively, none of the peers matched the asset class. The minority asset classes in the sample, despite being enough to form a peer group, were never grouped entirely together, amongst these minorities the Property funds were most often grouped together, while each Hedge fund had peer group of completely different asset funds.

5. Discussion

From the results, finbot (agent chain) shows enhanced descriptive capabilities relative to the domain chain due to prompt engineering. The responses across the datasets consistently follow the style in the prompt templates. Since the questions have randomly generated features from the metadata, finbot is robust enough to respond on general information requests, description, comparison (on performance) and identification prompts on any fund in the knowledge base. However as the datasets do not include rephrases of questions, it's not clear how robust finbot is to vague or poorly phrased queries, nor how responses vary across iterations.

The results on the general and description exposure questions show that finbot doesn't consistently invoke the optimal tool calling sequence such as using the Description tools to answer General Questions. This indicates that the React prompt needs to be optimized. An approach would be to leverage genetic algorithms [15] to iteratively improve the Agent Actions in the React prompt used to ensure sequence adherence, optimizing for correctness and performance.

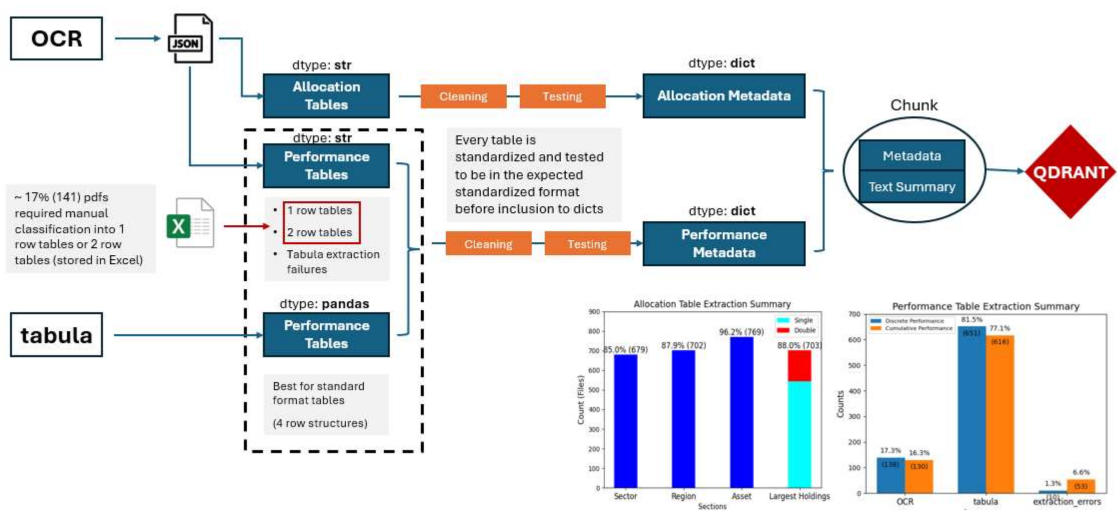
Throughout the datasets, the agent chain (finbot) is the most inefficient with a median time(P50) ranging from 0.5-2 minutes. Agents are generally more computationally expensive given iterative queries to the base LLM to execute the CoT, however the fund name filtering slows it further as before retrieval it iterates through every fund name in the knowledge base through the metadata. performance is adversely impacted to the point that the Ollama library times out. Alternatives to the filtered retrieval would have been to different embedding models (i.e. FinBert) or fine-tuning models on this dataset.

6. Conclusion

In conclusion, the finbot package has some analytical and descriptive portfolio analysis capacity on the investment fund universe. It can make descriptive and comparative observations on information retrieved or computed within the knowledge base, such as generate peer groups based on similarity score retrieval. This feature shows promise and if developed further might be useful to challenge industry standard classification, on a large enough knowledge base or for monitoring the “drift” of funds from their expected investment style.

7. Appendix

7.1 Tabular Information Extraction and Processing Pipeline:



We use two approaches to extract the table information: the tabula python package and Optical Character Recognition (OCR) through the unstructured package. We first read all the pdf files using OCR to identify the table elements in the pdf structure (not all the tables are actually identified as table elements and are categorized as text elements so we search through both structures) There are 6 potential tables: Cumulative Performance, Discrete Performance, Sector Allocation, Asset Allocation, Region Allocation and Largest Holdings. Every investment fund factsheet must have the two performance tables, however the remaining tables aren’t always necessarily disclosed in the factsheets (based on the investment strategy, asset managers can deem that some allocation tables may not be relevant or are justifiably merged as 1 table). As such not all factsheets have the same number of pages with the following distribution

Investment fund page distribution:

- Number of pages: 4, Count: 605 (75.44%)
- Number of pages: 3, Count: 158 (19.70%)

- Number of pages: 2, Count: 25 (3.12%)
- Number of pages: 5, Count: 11 (1.37%)

*Two of the files were corrupt from the web-scraping extraction and could not be read

OCR was able to successfully identify all the allocation tables where these were disclosed in the factsheets, with the final percentage of tables extracted and processed specified in Fig 1. Given all allocation table structures are consistent across factsheets, we were able to process the OCR string outputs relatively easy through regex to generate the metadata and textual summary for the vector store upload. However, this was not the case for the performance tables due to the difference in structure of the performance tables. During the processing pipeline we encountered two significant challenges in reading the pdf files

7.2 Tables Preprocessing Challenges:

a) Challenge: Identifying the structure of the performance table

Most performance tables have 4 rows (fund return, benchmark return, rank within sector, quartile) where the cumulative performance table is located on the first page and the discrete performance table is located on the second page. We refer to this form as the 'standard' form of performance tables. These standard form performance tables are easily extracted using the tabula package as pandas data frames. A subset of factsheets (~20%) have a non-standard form which we classify in 3 ways:

- '**absolute'**: performance tables have 1 row, the fund return, these funds are not monitored against a benchmark (57% of non-standard performance tables)
- '**relative'**: performance tables have 2 rows, the fund and benchmark return, the peer comparison metrics are not provided (28% of non-standard performance tables)
- '**complete'**: performance tables have 4 rows but they are not located on the 1st and 2nd pages as expected, rather they are shifted to the 2nd and 3rd pages due to a large investment objective (15% of non-standard performance tables)

To classify investment fund factsheets into these 3 categories we had to manually investigate each factsheet whose performance tables could not be processed through the pipeline as they were in a non-standard form. This manual investigation consisted of mapping each of these factsheets' file names in an excel file to their performance table's number of rows and page number. Based on this excel mapping the table processing pipeline was enhanced to use tabula to extract the performance for the 'complete' set from the appropriate indices and for the 'relative' and 'absolute' sets we used the OCR string outputs which were parsed according to their specific structure to re-build the table as a dictionary for further metadata storage and summarization.

b) Challenge: Split Largest Holdings table

The investment fund factsheets also include a Largest Holdings table, which if present is always the last table disclosed in the factsheet. Because of its location it's also the only table that gets split between pages and typically the first two rows (the largest two holdings) will appear on the second-to-last page and the remaining table rows appear on the last page. Since the significant portion of the table is located on the last page, tabula only identifies the second portion of the table, although from a portfolio analysis perspective, the top two holdings are the most relevant data points. As such we overcame this challenge in two ways:

a) ~20% of the time when the Largest Holdings table was split, OCR identified both tables as separate elements, in this case we simply merged the two elements that went through the same cleaning and processing pipeline as the rest of the allocation tables to form 1 complete table for upload.

b) ~80% of the time, OCR was also unable to identify the first two rows of the split table. In this case we used the Python package, pdfplumber to explicitly identify the bottom section of the second-to-last-page where these two rows are located and merge the pandas data frame extracted from pdfplumber with the one from tabula.

7.3. React Agent Prompt Template

The highlighted and red boxed represents the customization of the standardized prompt found on LangChain.

Customized ReAct Agent prompt:

Question: the input question you must answer
Thought: I will find the correct tool and use it to get the answer.
Action: If a tool is needed, choose the appropriate action from [{tool_names}]; otherwise, respond directly.
Action Input: The input to the tool (i.e., the question or any specific details)
Observation: The result returned by the tool
Thought: I now know the final answer
Final Answer: The response provided by the tool, do not modify the answer provided by the tool.

Action:
- If the tool used is one of [Compare Performance, Compare Peer Performance, Compare Exposure, Provide Description of Exposure, Provide Description of Performance],
- If the tool is one of [Identify Top Performer, Identify Worst Performer], then call one of the tools [Provide Description of Exposure, Provide Description of Performance] return the response from the tool as the Final Answer. Otherwise, select another tool from {tool_names} and use it.
- If the question lacks explicit fund names, use the **Get Relevant Funds** tool first, then select the appropriate tool from {tool_names}.
- If the question specifically asks to `compare` funds, use one of the **Compare** tools.

Thought: {agent_scratchpad})

We also provide additional instructions within the prompt template on what Actions the Agent should take based on the query and tool choice.

- If the tool used is one of [Compare Performance, Compare Peer Performance, Compare Exposure, Compare Peer Exposure, Provide Description of Exposure. Provide Description of Performance], return the response from the tool as the Final Answer. Otherwise, select another tool from {tool_names} and use it.
- If the tool used is one of [Identify Top Performer, Identify Worst Performer] then use one of the relevant tools from [Provide Description of Exposure, Provide Description of Performance].
- If the question lacks explicit fund names, use the **Get Relevant Funds** tool first, then select the appropriate tool from {tool_names}.
- If the question specifically asks to `compare` funds, use one of the **Compare** tools.

- If the question asks for a specific information request from the investment fund factsheets, use the **Pass Question to Domain Knowledge** tool.

7.4. General Questions (For Evaluation)

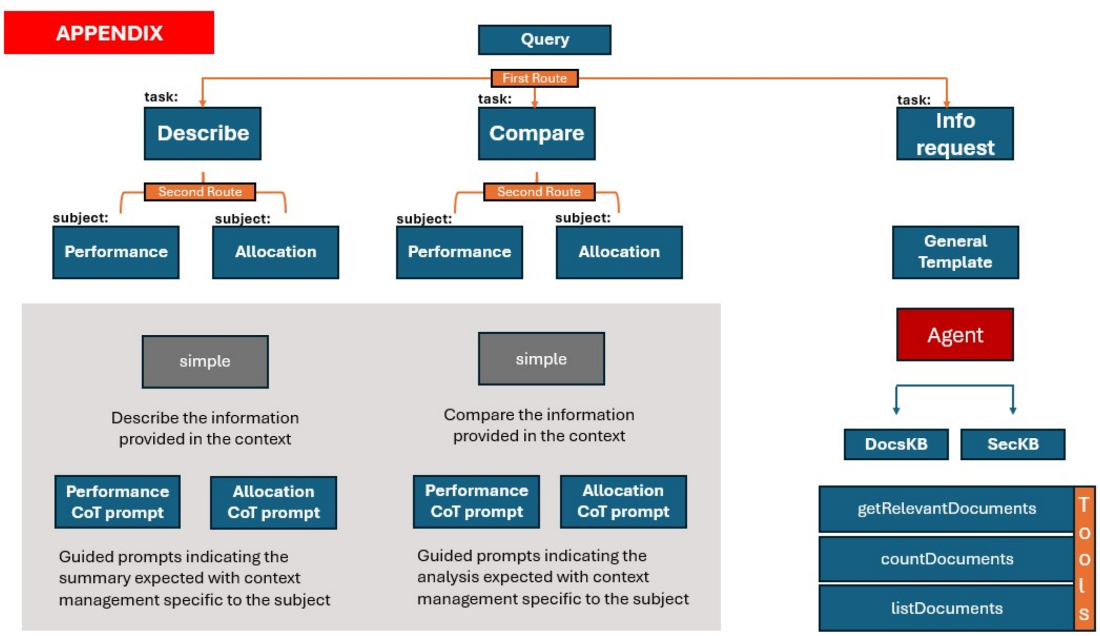
The values within curly brackets are randomized from a set list.

a) Specific investment objective questions: “What’s the investment objective of {fund_name}?”
b) General investment objective questions: “What is the investment objective of a {asset} fund?”
c) Knowledge base questions: Name {(1, 10)} {asset_class} fund from the knowledge base.
d) Performance information questions: ”What is the {random_period} {performance type} return of a {asset} fund?”
e) Exposure information questions: "What's the {request_type} {sec} allocation of {fund_name}?"

7.5.Route Chain Implementation:

This is the legacy implementation of Finbot. The Agent execution chain was never implemented however, the chain for information request uses simple retrieval and a general prompt.

- First Route: task classification RAG chain;
- Second Route: subject classification RAG chain;



References*:*

[1] Yager, K. G. (2023). Domain-specific chatbots for science using embeddings. *Digital Discovery, 2*(6), 1850-1861. https://doi.org/10.1039/D3DD00112A

[2] Polak, M. P., & Morgan, D. (2023). Extracting accurate materials data from research papers with conversational language models and prompt engineering. *arXiv preprint arXiv:2303.05352*. Retrieved from https://arxiv.org/abs/2303.05352

[3] Revolutionizing finance with LLMs: An overview of applications and insights. (2024). arXiv. https://ar5iv.org/pdf/2401.11641

[4] Kim, A., Muhn, M., & Nikolaev, V. V. (2024). Financial Statement Analysis with Large Language Models. Chicago Booth Research Paper Forthcoming, Fama-Miller Working Paper.

[5] Huang, R., Wang, Y., & Wang, H. (2023). *Exploring Natural Language Processing Techniques for Financial Narrative Disclosure*. Journal of Financial Data Science, 5(1), 45-63. doi:10.3905/jfds.2023.1.005

[6] Morningstar. (2024). Morningstar Category. Morningstar. Retrieved from https://admainnew.morningstar.com/webhelp/glossary_definitions/mutual_fund/glossary_mf_ce_Morningstar_Category.html

[7] van Dijk, K. S., De Bondt, W. F. M., & Kraussl, R. (2011). Geographic diversification and fund performance: Evidence from U.S. equity funds. *Journal of Banking & Finance, 35*(12), 3202-3211. https://doi.org/10.1016/j.jbankfin.2011.05.003

[8] Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022). *Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents* (Version 2). arXiv. https://arxiv.org/abs/2201.07207v2

[9] Jimeno Yepes, A., You, Y., Milczek, J., Laverde, S., & Li, R. (2024) Financial Report Chunking for Effective Retrieval Augmented Generation. arXiv. https://arxiv.org/abs/2402.05131v3

[10] Song K., Tan X., Qin T., Lu J., Liu. TY (2020). MPNet: Masked and Permuted Pre-training for Language Understanding. arXiv preprint arXiv:2004.09297. arXiv. https://arxiv.org/abs/2004.09297

[11] Lála, J., O'Donoghue, O., Shtedritski, A., Cox, S., Rodriques, S. G., & White, A. D. (2023). PaperQA: Retrieval-Augmented Generative Agent for Scientific Research. *arXiv*. https://arxiv.org/abs/2312.07559v2

[12] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2022). *Chain-of-thought prompting elicits reasoning in large language models*. arXiv. https://arxiv.org/abs/2201.11903

[13] Z. Yuan, H. Yuan (2023) Scaling Relationship on Learning Mathematical Reasoning with Large Language Models. arXiv. https://arxiv.org/abs/2308.01825

[14] Qdrant (2023). "Improving Retrieval Quality." Qdrant Documentation. Available at: https://qdrant.tech/documentation/tutorials/retrieval-quality/#:~:text=Qdrant%20provides%20a%20built%2Din,is%20the%20most%20important%20factor

[15] T. Alam, S. Qamar, A. Dixit, M. Benaida (2020). Genetic Algorithm: Reviews, Implementations, and Applications. *arXiv*. https://arxiv.org/abs/2007.12673

Data Source: Trustnet. (n.d.). *Managers list*. Retrieved June 14, 2024, from https://www2.trustnet.com/Managers/ManagersList.aspx