Irving Fisher Committee on
Central Bank Statistics

BIS

12th biennial IFC Conference: "Statistics and beyond: new data for decision making in central banks"

22-23 August 2024

Leveraging custom large language models for economic insights[1]

Nik Ahmad Akram and Eilyn Chong,
Central Bank of Malaysia

[1] This contribution was prepared for the conference. The views expressed in this publication are those of the authors and do not necessarily represent the official views of the Committee, its members, or the BIS.

Leveraging Custom Large Language Models for Economic Insights

Nik Ahmad Akram, Eilyn Chong

Abstract

Recent advancements in natural language processing, particularly with Large Language Models (LLMs), have shown promise across various domains. This paper explores the application of LLM technology to facilitate information retrieval from macroeconomic datasets. Our approach utilizes a custom-trained LLM to efficiently retrieve insights from extensive datasets in a user-friendly, natural human language format, thereby reducing dependence on data analytics tools. Preliminary comparison with commercially available, general-purpose LLMs, such as GPT, suggests that the LLMs should be fine-tuned with proper economic contexts to be more effective in delivering economic insights from vast datasets.

Keywords: Artificial Intelligence, Large Language Model, Generative AI, Economic Analysis

1. Introduction

In the pursuit of data-driven policymaking, dashboards have become increasingly prevalent for both internal users in policymaking institutions and the public. These dashboards often come loaded with extensive datasets, enabling users to self-explore and analyse the variety of data available. However, this extensive functionality can overwhelm lay users who seek quick insights. Similar to how a content-heavy website can hinder quick access to information, dashboards packed with a multitude of datasets can become difficult to navigate for users seeking rapid insights.

In this paper, we explore the potential of Large Language Models (LLMs) for question answering and information retrieval of economic data using natural language queries. Given the complexity and unique terminology of the economics domain, we fine-tune general-purpose LLMs using economic data and concepts. We draw inspiration from specialized LLMs in the finance industry, which outperform general-purpose LLMs in various natural language processing (NLP) tasks, including sentiment analysis, named entity recognition, news classification, and question answering (Wu et al., 2023; Chen et al., 2021; Sohangir et al., 2018).

The use of LLMs in central banks has remain focused on a specific range of tasks such as summarising and extracting insights from large volume of public and supervisory documents, conducting sentiment analyses of central bank communications, and identifying trending topics (Araujo, 2024). This paper aims to extend the application of LLMs in this domain by facilitating economic analysis through question answering and quick information retrieval.

We opt for a fine-tuned, contextual LLM for this specific NLP task. Several key considerations influenced this decision. First, the format of economic datasets, often stored in tabular form, presents a challenge. LLMs are designed to process text sequentially, rather than to interpret and process tabular data. Understanding a tabular structure requires LLMs to recognize patterns across rows and columns, understand relationships between different data points, and interpret the meaning of headers and cell values. Existing models like TAPAS and TaBERT attempt to address these issues by integrating specific embeddings and attention mechanisms to capture tabular structures. Despite these advancements, LLMs still struggle with directly interacting with tabular data (Herzig et al., 2020; Yin et al., 2020).

The extensive nature of datasets in central banks presents a second challenge. Supplying LLMs with large amounts of data can result in large prompts, leading to context dilution and risking the model's ability to respond accurately. Models like GPT-4 have a limited context window, meaning if a prompt is too long, essential parts of the context might get pushed out of this window, causing the model to miss critical information and respond inaccurately. Additionally, longer prompts increase complexity, which can lead to less relevant or coherent responses (theMind.io, 2023; Povio.com, 2023).

Third, the processing of economic data, including time series, may involve complex statistical analysis and modelling, which are beyond the scope of standard LLM capabilities at the time of writing.

In terms of output relevance, popular techniques such as the Retrieval-Augmented Generation (RAG) could improve the LLM's responses by retrieving relevant contextual information and passing it alongside the user's prompt. However, RAG works well with smaller number of tables that have good metadata and

descriptions. Given that policymaking institutions often deal with databases with multitude of tables with different schemas, a fine-tuned, contextual LLM that generates relevant Python code offers a more promising solution.

Preliminary results from our chosen approach show improvements in efficiency and accuracy over the general LLM. Implementing a fine-tuned, contextual LLM for economic analysis and embedding it in applications such as dashboards of economic indicators, allow users to gain quick insights from extensive tabular datasets by asking questions in a user-friendly, natural language format.

2. Methodology

Model Selection and Fine-Tuning

The model selected is GPT-4o, chosen as the base for our custom LLM due to its accessibility via OpenAI's API that allows for fine-tuning. Access via API also makes it easy to integrate the model into various applications. Fine-tuning involves uploading the training data through the API, which then adjusts the model to learn and perform specific tasks more effectively. In this case, the focus is on extracting and analysing information from tabular data.

Our fine-tuned model is developed with economists in mind, enabling them to use natural language to query, transform, and visualize data from a given database. While prompt engineering can yield better performance without fine-tuning, we consider a production use case where the model would be integrated into an application (e.g. a dashboard) that serves many users who demand quick economic insights but may have diverse prompt styles.

The fine-tuning process specifically aims to enhance the model's reliability to perform tasks not explicitly detailed in user prompts by training it with an understanding of economic relationships and conventions. For instance, when encountering terms like "growth", "GDP growth" or "YoY growth" in prompts, economists often refer to *year-over-year changes in real* GDP values. They also recognize how GDP is composed of consumer spending, investment, government spending, and net exports based on the expenditure approach, which is equivalent to the production approach that sums the gross value added of all industries.

The model is expected to not only behave as if it understands these economic relationships but also generate Python code for more complex data analysis. This generated code can then be automatically executed to extract and analyse economic insights from large datasets. Fine-tuning also helps the model handle intricacies with time series data, such as computing percentage changes first before filtering the series for a specific time period when calculating year-over-year changes for the selected period. Reversing this order can yield null results at the beginning of the filtered series and lead to different inferences, especially for shorter time series. When prompts include requests for data trends, the model will generate a chart to better provide a visualization to accompany the text. Fine-tuning ensures the model performs these tasks consistently, thereby mimicking the typical workflow of an economist when undertaking data queries, transformations, and visualizations.

Conceptual Framework

A. Initial Approach Using General-Purpose LLMs

The initial approach involves using a general-purpose GPT API to process the questions on economic data. This method entails directly supplying both the question and the tabular data in the prompt to the general-purpose model. Nevertheless, supplying the entire tabular data within the prompt can lead to context dilution and increased complexity, which can result in less relevant or coherent responses.

B. Enhanced Approach with Fine-Tuned LLMs

The general-purpose model is trained using sample data together with specific structure of the intended tabular datasets, which enabled the model to develop an understanding of the data's organization and presentation. Consequently, during the inference phase, we do not need to supply the entire tabular dataset to the model. Instead, the model generates Python code designed to ingest and process the data. This method ensures that data handling is performed by the generated code and mitigates the risks of context dilution and complexity, thereby optimizing the model's efficiency and accuracy in delivering relevant insights.

The enhanced approach also incorporates fine-tuning of the general-purpose GPT model using domain-specific data, including questions, economic data, Python codes, economic relationships and accounting identities. This fine-tuning process tailors the model to better handle economic terminologies and contexts, leading to more consistent and accurate insights. The model generates Python code based on the data structure and question, which is then automatically executed to produce the desired insights.

Such inclusion of expert-driven fine-tuning is a critical aspect of creating an effective, domain-specific LLM . This necessitates close collaboration with economists to guide the model's responses in a way that mirrors their thought processes and analytical framework. In our implementation, we incorporated an economist's perspective by directly including relevant data and information to define the "system role" to give the model extra context for the conversation and shape the model's approach to economic questions.

For example, the model is primed to utilize a demand-side approach to GDP calculation using the following equation:

$$GDP = c + i_{priv} + i_{pub} + g + (x - m) + inventory$$

where:

- c Private consumption
- i_{priv} Private investment
- i_{pub} Public investment
- g Government spending
- x Exports
- m Imports
- $inventory$ Changes in inventory levels

This expert-driven framework is embedded within the fine-tuned model, which effectively guides it to respond to prompts in a way that aligns with the contextual understanding used by economists. Such guidance is crucial to ensure that the model not only interprets data accurately but also reflects the economic relationships and accounting identities that economists apply to interpret drivers of macroeconomic indicators.

As an example, in Sample 9 in the Results section, the fine-tuned model shows its capability of calculating each sector's percentage contribution to GDP growth. Unlike a generalized LLM model that might use simple averages, the fine-tuned model gives a precise breakdown in percentage points (ppt), accurately showing each component's impact on GDP growth for 2023. This demonstrates how a model fine-tuned with input from domain experts can provide responses that align closely with the expected output of an economic analysis

To further enhance accuracy, we incorporated a protocol to prioritize certain official sources for global economic descriptions. When asked for a global economic overview, the model is prompted to source from:

- Bank for International Settlements' Annual Reports
- International Monetary Fund's World Economic Outlook (WEO)
- Organisation for Economic Co-operation and Development's Economic Outlook
- World Bank's World Development Report

This prioritization ensures the model provides descriptions tailored specifically for an economist audience. Instead of generalized explanations, it offers detailed insights that meet the depth and precision economists expect.

Process Flow

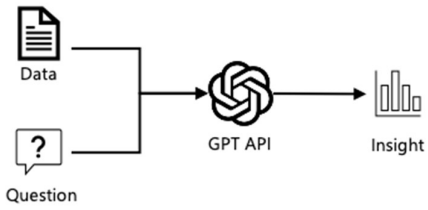
The process flow, as illustrated in the provided diagram, encompasses several key steps:

1. **Training Data Collection:** A comprehensive training dataset was gathered comprising economic data, Python code snippets, and relevant insights. This training data was meticulously curated to cover various aspects of the workflow of economic analysis.
2. **Model Fine-Tuning:** The collected data was used to fine-tune the GPT-4o model and define the system role. This involved adjusting the model's parameters to better understand and generate responses specific to economic contexts. The fine-tuning process ensures that the model can interpret economic data accurately and provide contextually relevant insights.
3. **Question and Data Input:** During the inference phase, users input their questions into the system. The fine-tuned GPT API processes these questions, leveraging its training to understand the context and generate appropriate Python code.
4. **Code Execution and Data Processing:** The generated Python code is executed to process the relevant data. This step involves ingesting the necessary data, performing calculations, and generating visualizations or summaries as needed.

5. **Insight Generation:** The results from the code execution are then compiled into insights, which are presented to the user in a clear and concise format. This allows economists to quickly obtain insights, reducing the need for manual data processing.

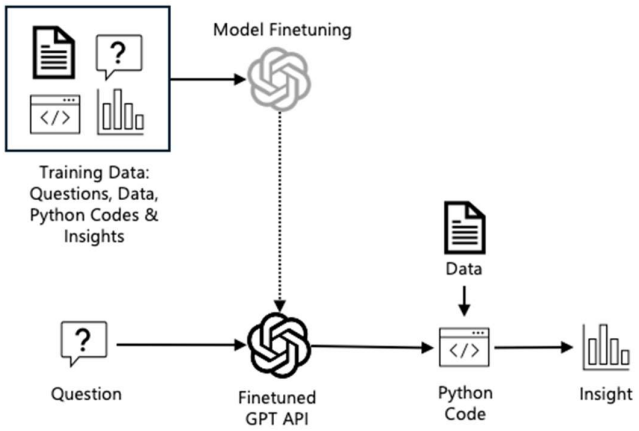
A. Initial Approach Using General-Purpose LLM

This method involves directly providing both the question and the tabular data in the prompt to the model.



B. Enhanced Approach with Fine-Tuned LLM

The model was trained with sample data to understand the dataset structure, fine-tuned, and has its "system" role defined, allowing it to generate Python code for data processing during inference, thus improving efficiency and accuracy.



3. Results

The performance of our fine-tuned, contextual LLM is evaluated using a series of benchmark tests. These tests included question-answering tasks, both extracting direct numbers from the table to more advanced questions like finding the trends.

The model can also answer some questions by generating Python codes for charting. For example, when asked about data trends, the model answers by generating code to plot a line chart based on the tabular data.

The fine-tuned LLM demonstrated superior performance compared to the general-purpose LLM in tasks requiring specific economic knowledge and data processing. It provided more accurate and contextually-relevant answers to economic queries and generated more relevant codes for data processing. In data extraction tasks, the custom LLM was able to identify and present relevant data points with higher precision.

For the purpose of this performance evaluation, we took a dataset of Malaysia's GDP time series data from an open data portal www.data.gov.my. The dataset consists of seasonally adjusted GDP level for every quarter since 2015 Q1 until 2024 Q1. The dataset also includes the components of GDP both from the expenditure and production approaches.

In straightforward cases of data extraction (Sample 1), both the general-purpose model and enhanced approach yielded correct results. However, when addressing more complex queries, such as computing year-over-year growth rates and the percentage contributions of GDP components to overall growth, the enhanced approach performed significantly better (Samples 2–9).

Sample 1

What is the GDP level for the agriculture sector in Q1 2022?

General Model Output:

RM 24,833 million

Enhanced Approach Output (Python Code):

```
# Filter the data for Q1 2022
q1_2022_data = data[(data["date"] == "01/01/2022") & (data["chart_type"] == "real_sa")]
# Extract the GDP level for the agriculture sector in Q1 2022
gdp_agri_q1_2022 = q1_2022_data["supply_agri"]

gdp_agri_q1_2022 : 24833000000
```

Both approaches are able to produce the correct answer, as it only requires data extraction.

Sample 2

What is the highest YoY growth in the manufacturing sector in 2021?

General Model Output:

12.35%

Enhanced Approach Output (Python Code):

```
# Calculate year-on-year (YoY) growth for the manufacturing sector
data["supply_manufacturing_yoy"] = data["supply_manufacturing"].pct_change(periods=4) *
100
# Filter the data for the year 2021
data_2021 = data[data["date"].dt.year == 2021]
# Find the highest YoY growth in the manufacturing sector in 2021
max_yoy_growth_2021 = data_2021["supply_manufacturing_yoy"].max()

# max_yoy_growth_2021 = 27.2345795387164
```

The general model is unable to provide the correct answer because calculating YoY growth involves specific computational steps that it cannot execute directly.

Sample 3

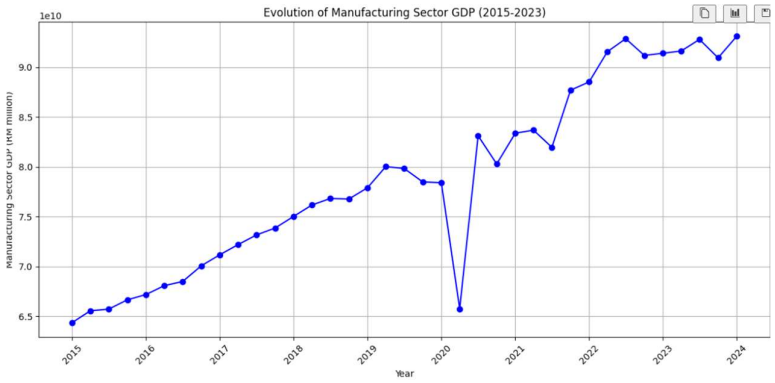
How has the manufacturing sector's GDP evolved from 2015 to 2023?

General Model Output:

Increased from 158,444,000,000 in 2015 to 236,823,000,000 in 2023.

Enhanced Approach Output (Python Code):

```
# Extract the data for the manufacturing sector GDP from 2015 to 2023
manufacturing_gdp = data[["date", "supply_manufacturing"]]
# Convert the date to datetime format for better visualization
manufacturing_gdp["date"] = pd.to_datetime(manufacturing_gdp["date"], format="%d/%m/%Y")
# Plot the evolution of the manufacturing sector GDP from 2015 to 2023
plt.figure(figsize=(12, 6))
plt.plot(manufacturing_gdp["date"], manufacturing_gdp["supply_manufacturing"], marker="o",
linestyle="-", color="b")
```



The general-purpose model provides a simple numerical answer, while the enhanced approach generates code for a line chart, offering a more comprehensive and insightful representation of the data. This enhanced approach provides a more relevant response, illustrating the trend over time.

Sample 4

Since 2020, how many quarters has the overall GDP growth been below 0%?

General Model Output:

01/04/2020:— This quarter's GDP is below 0%.
01/07/2020:— This quarter's GDP is above 0%.
01/10/2020:— This quarter's GDP is above 0%.
01/01/2021:— This quarter's GDP is above 0%.
01/04/2021:— This quarter's GDP is above 0%.
01/07/2021:— This quarter's GDP is below 0%.

Enhanced Approach Output (Python Code):

```
# Filter the data for the years 2015 to 2021
data_2015_2021 = data[(data["date"].dt.year >= 2015) & (data["date"].dt.year <= 2021)]

# Calculate the year-on-year (yoy) GDP growth
data_2015_2021["yoy_gdp_growth"] = data_2015_2021["overall"].pct_change(periods=4) * 100

# Extract the quarters with negative yoy GDP growth since 2020
negative_yoy_gdp_quarters_2020_onwards = data_2015_2021[(data_2015_2021["date"].dt.year
>= 2020) & (data_2015_2021["yoy_gdp_growth"] < 0)]

# Extract the quarters with negative yoy GDP growth prior to 2020
negative_yoy_gdp_quarters_prior_to_2020 = data_2015_2021[(data_2015_2021["date"].dt.year <
2020) & (data_2015_2021["yoy_gdp_growth"] < 0)]

quarters_negative_yoy_gdp_2020_onwards = negative_yoy_gdp_quarters_2020_onwards[["date",
"yoy_gdp_growth"]]

2020-04-01    -16.785837
2020-07-01     -2.036582
2020-10-01     -2.977868
2021-01-01     -0.395378
2021-07-01     -4.400473
```

The general-purpose model provides an incorrect answer. In contrast, the Enhanced Approach, through a few analytical steps (filters and compute growth), accurately identifies the quarters with negative GDP growth using detailed data analysis.

Sample 5

When was the most recent period of overall GDP growth exceeding 5%? Also, provide the growth rates of the GDP components during that period.

General Model Output:

01/01/2024: 4.00739E+11— This appears to be a significant number. However, without the previous year's data or a percentage change provided directly, I cannot compute whether this represents over 5% growth without more context or data.

Enhanced Approach Output (Python Code):

```
# Calculate the year-on-year (YoY) growth rates for the overall GDP and its components
data["overall_yoy_growth_rate"] = data["overall"].pct_change(periods=4) * 100
data["supply_agri_yoy_growth_rate"] = data["supply_agri"].pct_change(periods=4) * 100
data["supply_construction_yoy_growth_rate"] =
 --- Code truncated for brevity ---
data["demand_m_yoy_growth_rate"] = data["demand_m"].pct_change(periods=4) * 100
data["demand_nx_yoy_growth_rate"] = data["demand_nx"].pct_change(periods=4) * 100
data["demand_inventory_yoy_growth_rate"] = data["demand_inventory"].pct_change(periods=4)
* 100

# Find the most recent period with overall GDP YoY growth exceeding 5%
most_recent_period = data[(data["overall_yoy_growth_rate"] > 5) &
(data["overall_yoy_growth_rate"].notnull())].tail(1)
```

date	01/01/2023
overall_yoy_growth_rate	5.357358
supply_agri_yoy_growth_rate	1.687271
supply_construction_yoy_growth_rate	7.074106
supply_import_duties_yoy_growth_rate	13.383838
supply_mining_yoy_growth_rate	1.338334
supply_services_yoy_growth_rate	6.858594
supply_manufacturing_yoy_growth_rate	3.251243
demand_c_yoy_growth_rate	6.124202
demand_g_yoy_growth_rate	-2.213296
demand_i_yoy_growth_rate	4.921175
demand_x_yoy_growth_rate	-2.772273
demand_m_yoy_growth_rate	-6.798617
demand_nx_yoy_growth_rate	80.474955
demand_inventory_yoy_growth_rate	-51.611018

The output from general-purpose model indicates a need for additional context and data to determine if the GDP growth exceeds 5%. On the other hand, the Enhanced Approach successfully identifies the most recent period with an overall GDP growth exceeding 5% on the first quarter of, 2023, with a rate of 5.36%. It also provides detailed growth rates for GDP components.

Sample 6

What were the percentage contributions of the various sectors to GDP growth in 2023?

General Model Output:

To calculate the percentage contributions of various sectors to GDP growth for 2023, you would typically use the formula for the percentage contribution of a sector.

This formula is:

$$\text{Percentage Contribution} = \left(\frac{\text{Sector Growth}}{\text{Total GDP Growth}}\right) \times 100$$

Enhanced Approach Output (Python Code):

```
# Extract the data for the year 2023
data_2023 = data[data["date"].dt.year == 2023]

# Calculate the average GDP for each sector for the year 2023
data_2023_total = data_2023[["supply_agri", "supply_construction", "supply_import_duties",
"supply_mining", "supply_manufacturing", "supply_services"]].sum()

# Calculate the percentage contribution of each sector to the overall GDP
data_2023_percentage = (data_2023_total / data_2023_total.sum()) * 100

# Print the results
print(data_2023_percentage)
supply_agri              6.422817
supply_construction      3.617378
supply_import_duties     1.172254
supply_mining            6.218398
supply_manufacturing    23.383169
supply_services         59.185983
```

The General Model Output requires manual calculations for sector contributions to GDP, providing only the formula. In contrast, the Enhanced Approach automatically generates the correct code and delivers the sector contributions for 2023 directly and accurately.

Sample 7

Since 2020, how many quarters has the overall GDP growth been below 0%? Prior to 2020, which specific years also experienced negative GDP growth?

General Model Output:

Since 2020, there have been 7 quarters where the overall GDP growth was below 0%. Prior to 2020, the only year with negative GDP growth was 2019.

Enhanced Approach Output (Python Code):

```
data['year'] = data['date'].dt.year  # Ensure 'year' column is present in case it was dropped
# calculate year-on-year growth rates
data['yoy_gdp_growth'] = data['overall'].pct_change(periods=4) * 100

# Filter for quarters from 2020 onwards with year-on-year GDP growth below 0%
yoy_negative_2020_onwards = data[(data['date'] >= '2020-01-01') & (data['yoy_gdp_growth'] <
0)]
count_yoy_negative_growth_2020_onwards = yoy_negative_2020_onwards.shape[0]

annual_gdp = data.groupby('year')['overall'].sum().reset_index(name='annual_gdp')

# Calculate year-to-year growth rate based on annual GDP
annual_gdp['yoy_annual_growth'] = annual_gdp['annual_gdp'].pct_change() * 100

# Compile the result
{
   "count_yoy_negative_growth_2020_onwards": count_yoy_negative_growth_2020_onwards,
   "negative_yoy_growth_years = annual_gdp[annual_gdp['yoy_annual_growth'] <
0]['year'].tolist()


}

count_yoy_negative_growth_2020_onwards: 5
negative_yoy_growth_years : 2020
```

The General Model gives an incorrect answer, likely due to using the wrong data field or method. It reports 7 quarters of negative growth since 2020 and lists 2019 as having negative GDP growth, which is inaccurate. In contrast, the Enhanced Approach gives the correct result, showing 5 quarters of negative growth since 2020 and correctly identifying 2020 as the only year with negative year-to-year GDP growth.

Sample 8

Compare the volatility/standard deviation in overall quarterly GDP growth for 2015 - 2019 and 2020 onwards?

General Model Output:

The standard deviation of quarterly GDP growth rates, a measure of volatility, was low in both periods. From 2015 to 2019, it was about 0.0083, and from 2020 onward, it rose to 0.061. Although higher in recent years, both numbers are close to zero.

Enhanced Approach Output (Python Code):

```
# Recalculate year-on-year quarterly GDP growth if not already present in the data
if 'yoy_gdp_growth' not in data.columns:
    data = data.sort_values('date').reset_index(drop=True)
    data['yoy_gdp_growth'] = data['overall'].pct_change(periods=4) * 100

# Extract the year from the date if not already present
if 'year' not in data.columns:
    data['year'] = data['date'].dt.year

# Split the data into the periods 2015-2019 and 2020 onwards
pre_2020_data = data[(data['year'] >= 2015) & (data['year'] <= 2019)]
post_2020_data = data[data['year'] >= 2020]

# Calculate the standard deviation of quarterly GDP growth for each period
volatility_pre_2020 = pre_2020_data['yoy_gdp_growth'].std()
volatility_post_2020 = post_2020_data['yoy_gdp_growth'].std()

# Compile the results
volatility_comparison = {
    '2015-2019 GDP Growth Volatility': volatility_pre_2020,
    '2020 onwards GDP Growth Volatility': volatility_post_2020
}


volatility_comparison
{'2015-2019 GDP Growth Volatility': 0.8593957861240162,

 '2020 onwards GDP Growth Volatility': 7.339528596907005}
```

The General Model computes Quarter-on-Quarter (QoQ) growth, but it produced incorrect standard deviation values. The standard deviation for these QoQ growths are 3.1 for 2015-2019 and 7.5 for 2020 onwards. In contrast, the Enhanced Approach calculates Year-over-Year (YoY) growth, a more commonly used transformation, and yields accurate standard deviation values .

Sample 9

Which components contributed to the downturn in GDP growth during the most recent economic trough? Additionally, describe the global economic conditions during that period based on your existing knowledge base.

General Model Output:

The year 2020 marked the most recent economic trough, with a GDP growth decline of approximately -5.46%. The downturn in 2020, indicated by a significant GDP decline, largely resulted from the global economic impact of the COVID-19 pandemic.
The mean values of the economic components for this year are as follows:

- Supply-Side Components:
- Agriculture: 24.78 billion
- Construction: 13.40 billion
- Manufacturing: 76.90 billion
- Mining: 23.22 billion
- Services: 194.42 billion
- Demand-Side Components:
- Consumption (demand_c): 200.69 billion
- Government Spending (demand_g): 44.93 billion
- Investment (demand_i): 70.29 billion
- Imports and Duties: 3.84 billion
- Imports (demand_m): 185.77 billion
- Net Exports (demand_nx): 21.77 billion

Enhanced Approach Output (Python Code):

```
# Filter out the latest year if it doesn't contain all 4 quarters
latest_year = data['year'].max()
quarters_in_latest_year = data[data['year'] == latest_year]['date'].dt.quarter.nunique()
if quarters_in_latest_year < 4:
    data_filtered = data[data['year'] < latest_year]
else:
    data_filtered = data

# Calculate annual sums for GDP components
yearly_data_sum = data_filtered.groupby('year').agg({
    'overall': 'sum',
    'supply_agri': 'sum',
    'supply_mining': 'sum',
    'supply_manufacturing': 'sum',
    'supply_construction': 'sum',
    'supply_services': 'sum',
    'supply_import_duties': 'sum',
    'demand_c': 'sum',
    'demand_g': 'sum',
    'demand_i': 'sum',
    'demand_x': 'sum',
    'demand_m': 'sum',
    'demand_inventory': 'sum'
}).reset_index()
```

```
# Calculate year-on-year GDP growth for overall GDP
yearly_data_sum['yoy_yearly_growth'] = yearly_data_sum['overall'].pct_change() * 100
trough_year = yearly_data_sum.loc[yearly_data_sum['yoy_yearly_growth'].idxmin(), 'year']
overall_growth_rate = yearly_data_sum.loc[yearly_data_sum['year'] == trough_year,
'yoy_yearly_growth'].values[0]

# Percentage point contributions by expenditure
trough_values = yearly_data_sum[yearly_data_sum['year'] == trough_year].iloc[0]
prev_values = yearly_data_sum[yearly_data_sum['year'] == (trough_year - 1)].iloc[0]
ppt_contributions_expenditure = {
    'Private Consumption (C)': (trough_values['demand_c'] - prev_values['demand_c']) /
prev_values['overall'] * 100,
    'Investment (I)': (trough_values['demand_i'] - prev_values['demand_i']) / prev_values['overall'] *
100,
    'Public Consumption (G)': (trough_values['demand_g'] - prev_values['demand_g']) /
prev_values['overall'] * 100,
    'Exports (X)': (trough_values['demand_x'] - prev_values['demand_x']) / prev_values['overall'] *
100,
    'Imports (M)': -(trough_values['demand_m'] - prev_values['demand_m']) / prev_values['overall']
* 100,
    'Change in Inventories': (trough_values['demand_inventory'] -
prev_values['demand_inventory']) / prev_values['overall'] * 100,
    'Overall GDP Growth Rate': overall_growth_rate
}

# Percentage point contributions by sector
ppt_contributions_sectors = {
    'Agriculture': (trough_values['supply_agri'] - prev_values['supply_agri']) / prev_values['overall'] *
100,
    'Mining': (trough_values['supply_mining'] - prev_values['supply_mining']) /
prev_values['overall'] * 100,
    'Manufacturing': (trough_values['supply_manufacturing'] -
prev_values['supply_manufacturing']) / prev_values['overall'] * 100,
    'Construction': (trough_values['supply_construction'] - prev_values['supply_construction']) /
prev_values['overall'] * 100,
    'Services': (trough_values['supply_services'] - prev_values['supply_services']) /
prev_values['overall'] * 100,
    'Import Duties': (trough_values['supply_import_duties'] - prev_values['supply_import_duties']) /
prev_values['overall'] * 100,
    'Overall GDP Growth Rate': overall_growth_rate
}


Trough year: 2020

According to reports from the IMF World Economic Outlook and OECD Economic Outlook, the
2020 global GDP downturn stemmed from the COVID-19 pandemic's unprecedented impact on
public health and economic activity. Lockdowns and supply chain disruptions curtailed
production and investment, while reduced consumer spending and international trade hampered
demand.
```

Expenditure Contributions (ppt):

- Private Consumption (C): -2.32
- Investment (I): -3.33
- Public Consumption (G): +0.50
- Exports (X): -5.46
- Imports (M): +4.48
- Change in Inventories: +0.66
- Overall GDP Growth Rate: -5.46

Sector Contributions (ppt):

- Agriculture: -0.17
- Mining: -0.70
- Manufacturing: -0.61
- Construction: -0.90
- Services: -3.01
- Import Duties: -0.06
- Overall GDP Growth Rate: -5.46

Both the General Model and Enhanced Approach successfully identify 2020 as the trough year. However, the General Model uses simple mean values to describe the contributions of GDP components, which gives a basic overview of each sector's size rather than its specific impact on growth.

In contrast, the Enhanced Approach provides contributions in percentage points (ppt), which more accurately reflects the influence of each component on GDP growth.

The General Model provides a broad explanation, attributing the 2020 GDP downturn to the global economic impact of the COVID-19 pandemic. While this gives a basic understanding, it lacks specific details on the underlying factors and supporting sources.

In contrast, the Enhanced Approach offers a more comprehensive explanation, detailing the key drivers of the downturn, such as lockdowns, supply chain disruptions, and reduced consumer spending, which affected both production and demand. It also references reliable sources like the IMF and OECD, adding credibility and depth to the explanation.

4. Discussion

Advantages of Custom LLMs

The primary advantage of a custom-trained LLM is its ability to understand and interpret domain-specific instructions. For economists, this means more intuitive and efficient data analysis. The model's capacity to generate codes and relevant content allows for seamless interaction for quick insights.

The custom LLM's performance in specific economic tasks in Section 3 highlights the importance of domain-specific training. While general-purpose LLMs offer broad capabilities, their effectiveness can be significantly enhanced through targeted fine-tuning.

Additionally, using a code generation approach eliminates the need to include the entire table within the prompt. Instead, the model generates Python code that processes the data, which ensures that the data handling is done efficiently and accurately. This method reduces the risk of context dilution and complexity, allowing the model to focus on generating precise and contextually relevant responses.

Challenges and Limitations

Despite the promising results, there are challenges associated with using the enhanced approach with fine-tuned LLMs:

- **Reduced Generalization:** Fine-tuning the model to handle very specific economic contexts can limit its generalizability. As a result, the model might respond with "unable to answer" when posed with questions outside its specialized training, or worse, hallucinate and draw incorrect inferences. To mitigate this, it is crucial to continually expand the training dataset to handle common or emerging tasks in an economist's workflow.
- **Code Generation Errors:** The model generates Python code based on the data structure and the question asked. However, this generated code might contain errors or inconsistencies, or call for libraries that are not installed within a given environment, leading to execution failures and unanswered queries. Therefore, thorough validation and robust error-handling mechanisms are essential to ensure the reliability of the generated code, particularly in a production setting.

5. Conclusion

This paper demonstrates the potential of custom-trained LLMs to facilitate economic analysis through question answering and quick information retrieval in natural language. By fine-tuning a general-purpose LLM with economic data and context, we developed a model that is more efficient and accurate in extracting economic insights, thereby extending the application of LLMs in the economics domain.

For future work, we plan to extend this approach to open-source LLM models, such as LLaMA, to address issues related to data governance that restricts the transfer of sensitive data to external APIs. Using open-source models will allow us to create a closed framework where all operations can be hosted on-premises, providing better control over data privacy and security, ensuring compliance with data governance requirements.

Moreover, efforts are underway to develop a table retrieval mechanism that can be incorporated to enhance the model's capabilities. Considering the extensive nature of economic datasets, this feature would enable the model to automatically select the most relevant datasets based on users' query, which are then processed by the custom-trained LLMs to extract and compute relevant insights, thereby improving both efficiency and accuracy. This ongoing study aims to further strengthen the model's capability to provide precise, contextually relevant economic insights. (Seah et al., 2024)

References

Araujo, Douglas Kiarelly Godoy, Doerr, Sebastian, Gambacorta, Leonardo, and Tissot, Bruno (2024). "Artificial Intelligence in Central Banking". *BIS Bulletin*, No. 84.

Chen, Zhiyu, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa et al. "FinQA: A dataset of numerical reasoning over financial data." *arXiv preprint arXiv:2109.00122* (2021).

Herzig, Jonathan, Nowak, Pawel Krzysztof, Müller, Thomas, Piccinno, Francesco, and Eisenschlos, Julian Martin (2020). "TAPAS: Weakly Supervised Table Parsing via Pre-training". *arXiv preprint* arXiv:2004.02349

OpenAI (n.d.). "Text Generation Guide." Retrieved 7th November 2024 from platform.openai.com.

Povio.com (n.d.). "GPT-4 Turbo Preview: Exploring the 128k Context Window". Retrieved 31st July 2024 from povio.com.

Seah, B. K., Chong, E., & Chew, M. (2024). Using RAG and LangChain to drive chatbots for interactive economic data analysis and insight (Forthcoming)

Sohangir, Sahar, Dingding Wang, Anna Pomeranets, and Taghi M. Khoshgoftaar. "Big Data: Deep Learning for Financial Sentiment Analysis." *Journal of Big Data* 5, no. 1 (2018): 1-25.

TheMind.io (n.d.). "Navigating the Limits of Long Context Windows in GPT-4". Retrieved 31st July 2024 from themind.io.

Wu, Shijie, 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 preprint* arXiv:2303.17564.

Yin, Pengcheng, Graham Neubig, Wen-tau Yih, and Sebastian Riedel (2020). "TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data". *arXiv preprint* arXiv:2005.08314.

Leveraging Custom Large Language Models for Economic Insights

Nik Ahmad Akram & Eilyn Chong

BANK NEGARA MALAYSIA

Motivation

Data-Rich World: Why We Need New Methods for Extracting Meaningful Insights

- In today's **data-rich environment,** economists have access to more data than ever before, often stored in accessible formats. However, they typically deal with complex **tabular data** that comes **in various schemas**, making it **challenging to extract** meaningful **insights** efficiently.

- The challenge is **to find an easier way** to **get insights** from these large and complex datasets**. Current methods** often require a lot of effort and **specialized skills in data analysis**. This study aims to **make information retrieval more straightforward** through natural language queries, potentially making the process easier for economists and decision-makers.

Global Datasphere

The global datasphere expected to double by 2026 relative to the end of 2022

2022: 101 Zettabytes
2023: 123 Zettabytes

2026: 221 Zettabytes

Sources: AVANT by Avison Young, IDC Global DataSphere Forecast (2022-2026)

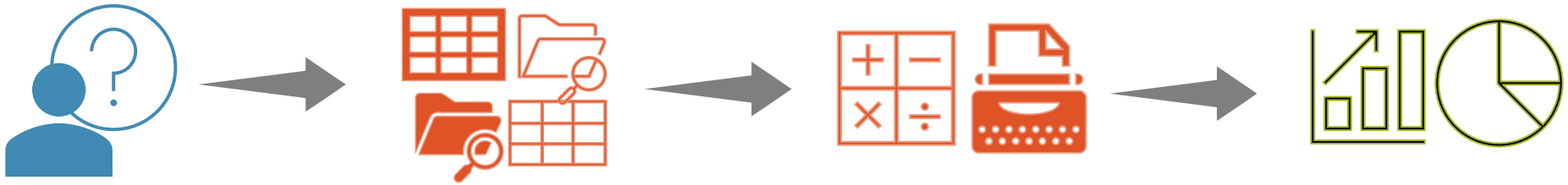
Addressing Dataset Navigation Challenges

- In our organization**, we have curated dataset specifically for economists** and presented it through a **data dashboard**. However, users still **face challenges navigating** this dashboard due to the **vast datasets** and the **varied questions different users have**, each requiring **specific data manipulation** to provide accurate insights.

- With the rise of AI, there's growing interest in implementing **Large Language Models (LLMs) to address these challenges**. However, **general LLMs** are too broad and not suited for the nuanced, specific questions economists ask. This situation calls for **a specialized LLM framework** designed to meet the unique needs of our users

Navigating Complex Data to Drive Informed Decisions

Streamlining Economic Analysis: From Questions to Insights in a Data-Rich World



Question formulation.

Relevant datasets are identified and processed.

Datasets are processed.

The processed data is analyzed to derive insights.

Challenges in Handling Economic Data

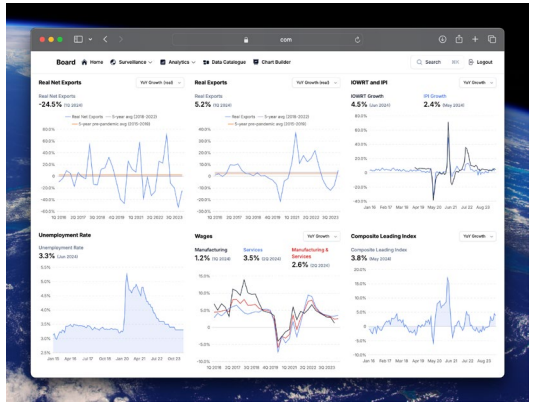
Dataset Formats: Economic datasets are often presented in tabular forms, requiring specific expertise and tools to manage effectively.

Structured Workflow: Economists adhere to a systematic process that necessitates precise data manipulation to generate insights.

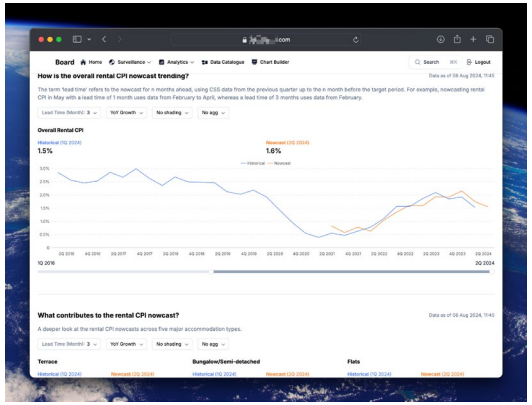
Context Dilution: Directly inputting large datasets into a general LLM can dilute context, resulting in less accurate outputs.

Dashboard: Curated Dataset for Economists' Analysis

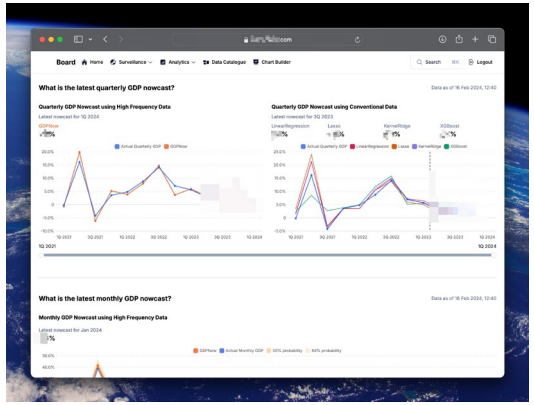
Economics Indicator Dashboard in BNM



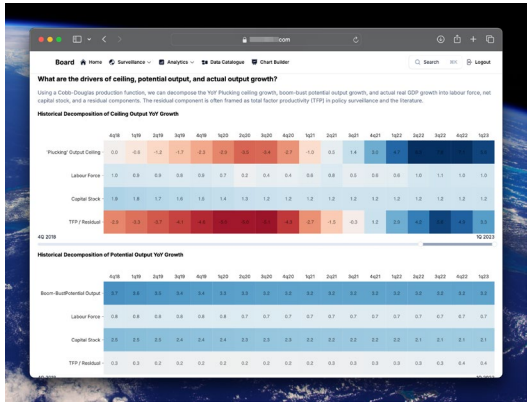
Comprehensive Overview: Multiple Economic Indicators on a Single Dashboard Page



Nowcasting Rental CPI: Analysing Current Trends in the Rental Market



GDP Forecast: Leveraging High-Frequency Data for Enhanced Accuracy



Heatmap Visualizations

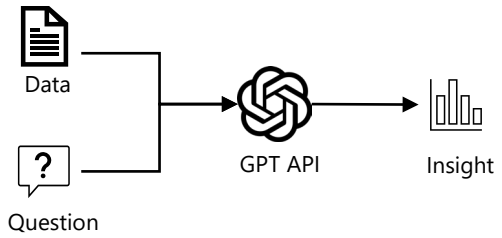
Highlights

- **100+ Charts:** Visualizations to track key economic indicators.
- **50+ Datasets:** Comprehensive data coverage for detailed analysis.
- **Daily Updates:** Ensures the latest information is always available.
- **Web-Based Access:** Optimized for both laptop and tablet use.
- **API Access:** Allows users to directly extract data for custom analysis.
- **Interactive:** Users can easily filter data by date and other parameters for customized insights.

Conceptual Framework & Proposed Solutions

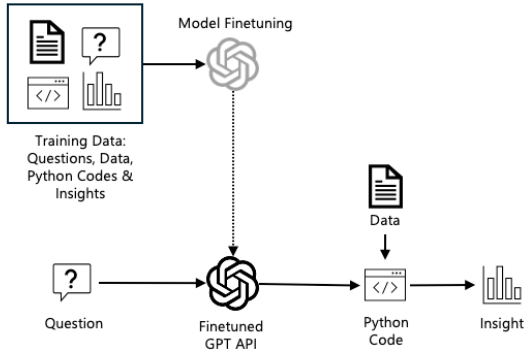
Initial Approach Using General-Purpose LLM

This method involves directly providing both the question and the tabular data in the prompt to the model.



Enhanced Approach with Fine-Tuned LLM

The model was trained with sample data to understand the dataset structure, allowing it to generate Python code for data processing during inference, thus improving efficiency and accuracy.



Proposed Solutions:

- **Model Fine-Tuning**: The LLM is fine-tuned using training data that includes typical questions, data structures, Python code, and the format in which economists usually present answers. This fine-tuning is done initially and updated periodically as new types of questions emerge.
- **Python Code**: When a user asks a question, the model generates the appropriate Python code tailored to the query.

Data itself is not fed into the LLM, which helps prevent context dilution that could occur if the model had to handle large datasets directly from the query

This process ensures that the model provides correct responses to specific, nuanced questions posed by economists.

Sample Results #1 : What is the GDP level for the agriculture sector in Q1 2022?

- For the purpose of this performance evaluation, we took a dataset of Malaysia's GDP time series data from an open data portal www.data.gov.my.
- The dataset consists of GDP level for every quarter since 2015 Q1 until 2024 Q1. The dataset also includes the components of GDP both from the expenditure and production approaches.

General Model Output:

Answer: RM 24,833 million

Enhanced Approach Output (Python Code)

Answer: 24833000000

```
# Filter the data for Q1 2022
q1_2022_data = data[(data["date"] == "01/01/2022") &
(data["chart_type"] == "real_sa")]
# Extract the GDP level for the agriculture sector in Q1 2022
gdp_agri_q1_2022 = q1_2022_data["supply_agri"]

gdp_agri_q1_2022 : 24833000000
```

Both approaches are able to produce the correct answer, as it only requires data extraction.

Sample Results #2 : What is the highest YoY growth in the manufacturing sector in 2021?

General Model Output:

Answer: 12.35% 

Enhanced Approach Output (Python Code)

Answer: 27.2345795387164 

```
# Calculate year-on-year (YoY) growth for the manufacturing sector
data["supply_manufacturing_yoy"] =
data["supply_manufacturing"].pct_change(periods=4) * 100
# Filter the data for the year 2021
data_2021 = data[data["date"].dt.year == 2021]
# Find the highest YoY growth in the manufacturing sector in 2021
max_yoy_growth_2021 = data_2021["supply_manufacturing_yoy"].max()

# max_yoy_growth_2021 = 27.2345795387164
```

The general model is unable to provide the correct answer because calculating YoY growth involves specific computational steps that it cannot execute directly.

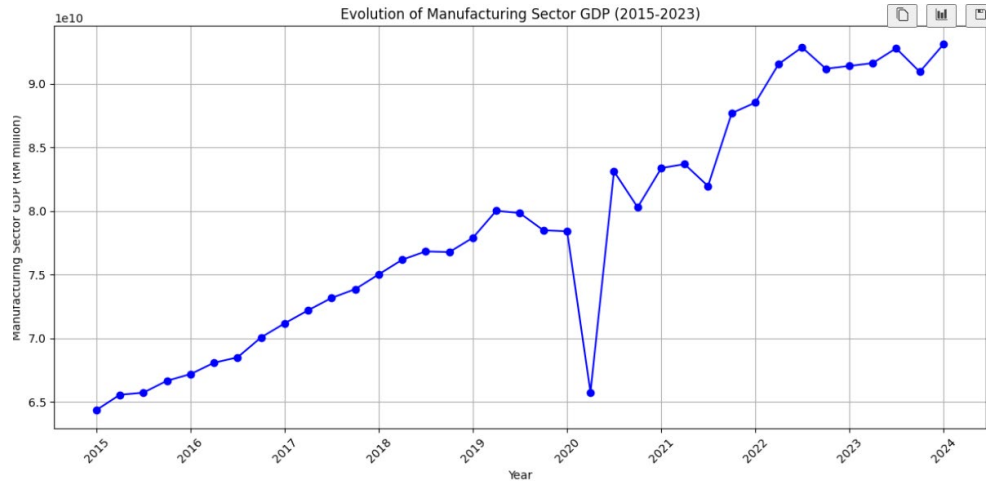
Sample Results #3 :How has the manufacturing sector's GDP evolved from 2015 to 2023?

General Model Output:

Answer: Increased from 158,444,000,000 in 2015 to 236,823,000,000 in 2023.

Enhanced Approach Output (Python Code)

```
# Extract the data for the manufacturing sector GDP from 2015 to 2023
manufacturing_gdp = data[["date", "supply_manufacturing"]]
# Convert the date to datetime format for better visualization
manufacturing_gdp["date"] = pd.to_datetime(manufacturing_gdp["date"], format="%d/%m/%Y")
# Plot the evolution of the manufacturing sector GDP from 2015 to 2023
plt.figure(figsize=(12, 6))
plt.plot(manufacturing_gdp["date"], manufacturing_gdp["supply_manufacturing"], marker="o", linestyle="-", color="b")
```



The general-purpose model provides a simple numerical answer, while the enhanced approach generates code for a line chart, offering a more comprehensive and insightful representation of the data. This enhanced approach provides a more relevant response, illustrating the trend over time.

Discussion & Conclusion

- **Domain Specific**: A custom-trained LLM excels at understanding and interpreting domain-specific language, making data analysis more intuitive and efficient for economists.
- **Context Dilution**: The model generates Python code instead of directly handling large datasets, reducing context dilution and ensuring accurate data processing.
- **Data Governance**: For future work, we plan to incorporate open-source LLM models like LLaMA to create a secure, on-premises framework that better protects data privacy.
- **Enhanced Efficiency**: We aim to add a table retrieval mechanism to automatically select the most relevant table from large datasets, further improving efficiency and accuracy.

Challenges:

- **Reduced Generalization**: Specialization may limit versatility, leading to errors with unfamiliar queries. Expanding the training dataset is essential to cover a broader range of tasks.
- **Code Errors:** Generated code might fail due to errors or missing libraries. Robust validation and error-handling are essential.

thank you ~!

email: ahmadakram@bnm.gov.my & eilyn@bnm.gov.my