LLM Orchestration with the World Bank API

Text as Data Final Report, Spring 2024

Sharif Kazemi

2024-05-02

Introduction

Report outline

1. Introduction to the motivation, research question, and key concepts in this paper to contextualise the product.
2. A description of the LLM orchestration process driving this product.
3. Demonstration of the product using pre-set parameters.
4. Evaluation of the product's performance across readability, similarity, and accuracy.
5. Concluding thoughts.

Research question: Large-language models as policy tools

Large-language models (LLMs) are trained using deep learning techniques and very large datasets to understand and generate text. Through word embeddings and other inner-workings, they derive a relational understanding of language that can be used to predict responses to prompts from users (Mitts, Lecture 11, 2024). With the proliferation of AI tools and their infusion into daily lives, they will also become increasingly used in policymaking settings to inform decision-makers (Liao and Wortman Vaughan, 2024).

However, the initial training dataset of the LLM might not provide the most relevant information and LLMs are known to fabricate ('hallucinate') information when it is uncertain (Miller, 2023).

Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a process by which a machine learning model, such as an LLM, generates content with the aid of retrieved information (Lewis, et. al., 2020). Directly referencing sources is an added value of the RAG approach, whereas a simple AI chatbot only relies on the dataset it was previous trained on. This is one of the methods hypothesized to improve LLM functionality for applications in high-stakes settings such as policymaking.

This product's approach to answering the research question

This product utilizes an LLM (OpenAI's GPT 4.0) in summarizing the results from World Bank research by specifying the sector of interest, the years of interest, and the country. For example, education policies in Mexico between 2020 and 2024. These summaries offer a more accessible tool for background research and

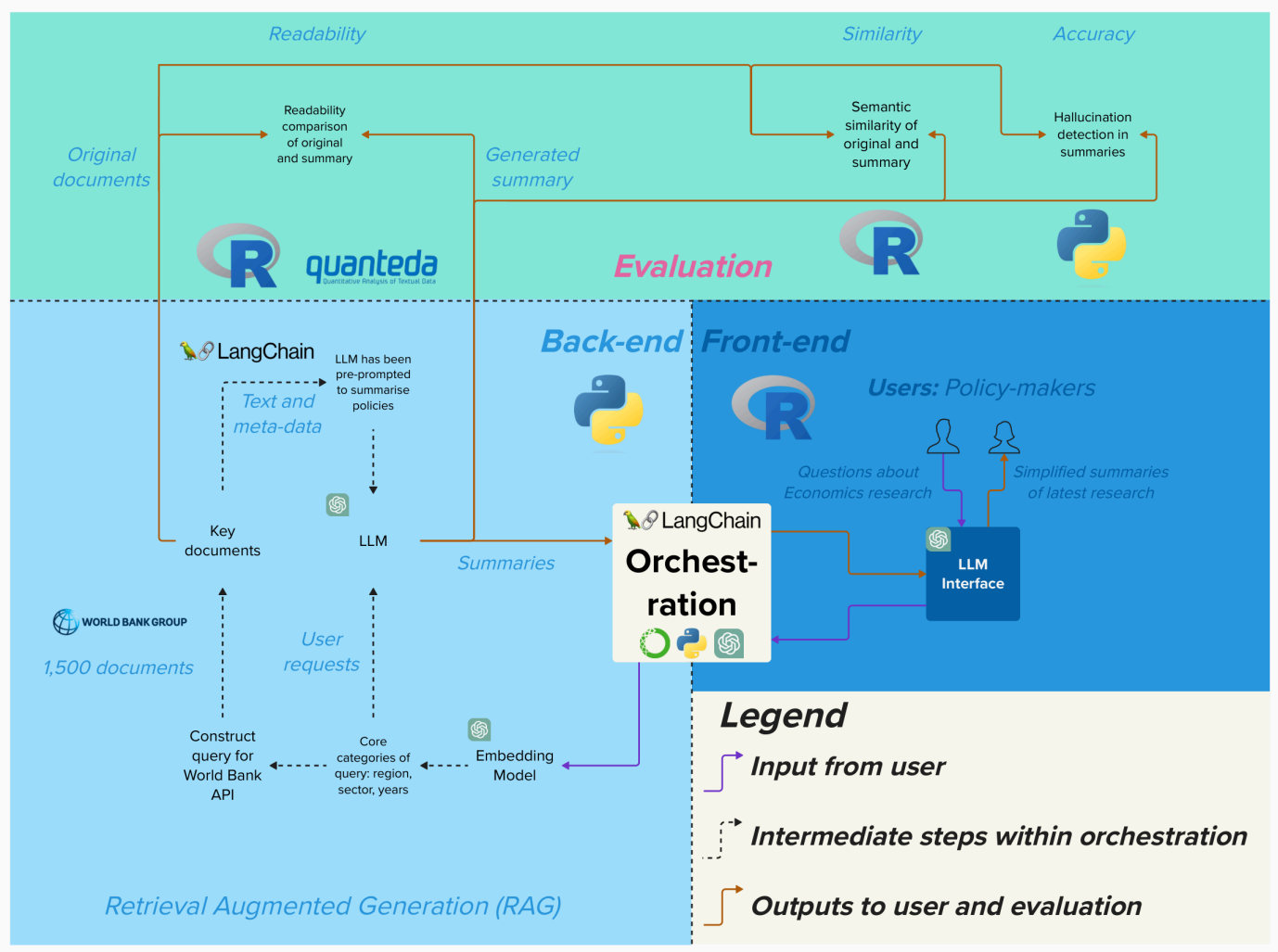
policy recommendations for policymakers around the world, whilst also citing specific sources which can then be used in tandem with expert discussions to inform policy making. The benefits of the LLM would be in answering simple questions regarding policy research for policymakers and other stakeholders who are not experts within that sector but need to conduct analysis. The use case is based on studies that have highlighted that ChatGPT and LLMs are ‘highly useful’ in summarizing and simplifying research (Miller, 2023).

The overall process here is LLM Orchestration. This is the process of linking multiple LLMs and other functions, such as APIs and document pre-processing, to achieve complex tasks that go beyond the capabilities of a single LLM. An overview of this orchestration is provided in the next section.

The product’s outputs are also evaluated across readability, similarity, and accuracy to understand if such an approach meets more desirable summaries that can be used by policymakers.

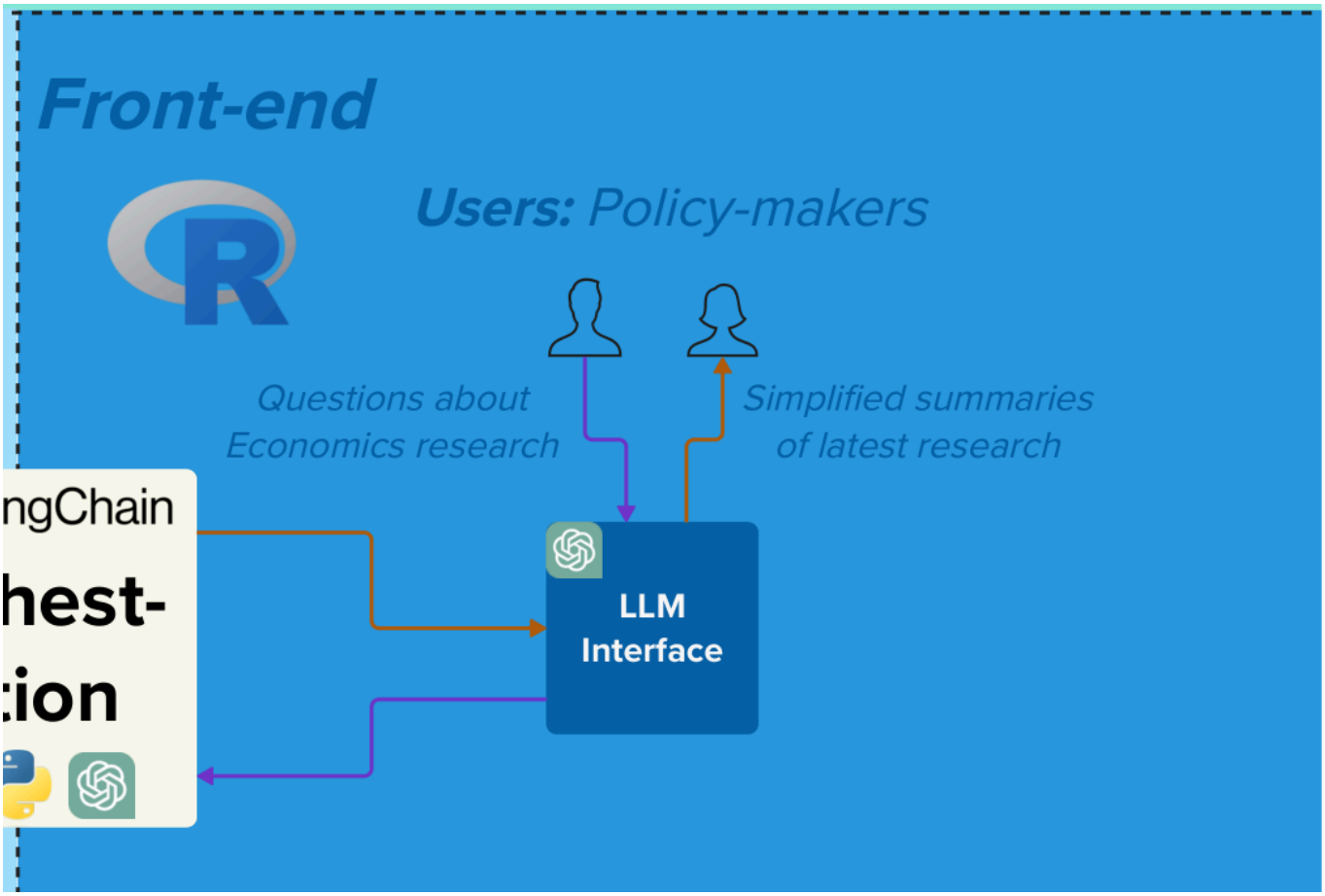
Large Language Model Orchestration

Description of operation



Here’s the overall architecture.

Front-end: User prompts query

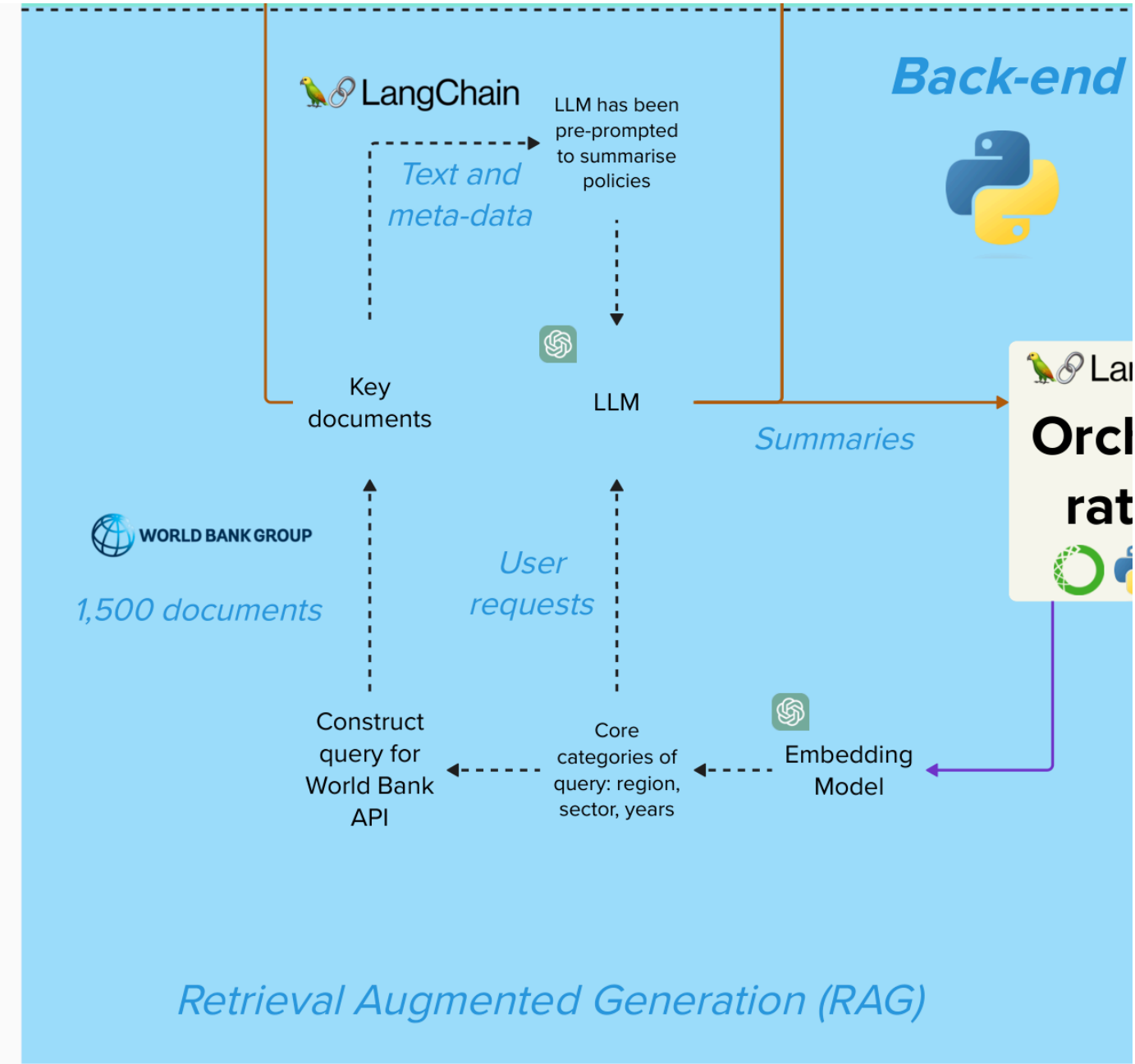


The front-end segment of the architecture.

The user is asked to input three variables into the model:

- user_prompt: *Please request your policy summary, specifying country, year, and sector of interest*
- documents_to_index: *Please request number of documents to index (5 is good for a short query)*
- page_limit_per_document: *Please request the page limit for indexed documents that will be sent to the OpenAI API (50 is good for a short query)*

Back-end: User prompt embedding



The back-end segment of the architecture.
The following steps are conducted for deciphering the user prompt:

- The user_prompt is tokenised and analysed to find the years of interest, country specified, and sectors.
- Sectors are derived from proper nouns in the user_prompt using nlp (spacy).
- The sector, country, and years, alongside the parameters set by the user, are used to construct a query to the World Bank API. From this query, the most relevant documents (as defined by the sector keyword) are retrieved in a .csv file.

Back-end: Document processing and collation

The indexed documents from the World Bank API, held in the .csv, are analysed sequentially to see if they meet the page and token length specifications (see below section on ‘Potential errors in operation, and their work-arounds’).

If they meet these parameters, then they are saved in two ways:

- Each document is saved separately and numbered.
- Appended to a single file with the document number and titles.

Back-end: Prompt-engineering for the LLM

Using the RAG process, a prompt has been refined which requests the LLM to:

1. summarise **policy recommendations** from the documents,
2. refer to the country, year, and sector variables derived in the embedding stage - specifically cite each document when giving the source's recommendations,
3. and then provide an overall summary of all documents.

All of the documents are first summarised separately using a loop function, and then the file with all appended documents is sent to the GPT for an overall summarisation.

Front-end: Policy recommendations

The final outputs are document-specific summaries and an overall summary of all documents. These summaries include the main policy recommendations that can be inferred from the documents.

Potential errors in operation, and their work-arounds

Issues from a narrow search which can cause a failure in operation have been anticipated. Functions are set-up to monitor these errors and give helpful feedback to the user for iterating their prompt parameters for a successful run.

- **Problem:** No relevant documents were found. **Recommended solution:** Broaden the years, sectors, and countries in the initial prompt.
- **Problem:** Document count after filtering is zero. **Recommended solution:** Recommend expanding number of documents to index and/or pages per document, and then re-running the script.
- **Problem:** No relevant documents were found. **Recommended solution:** Broaden the years, sectors, and countries in the initial prompt.

Other issues that can break the operation have also been anticipated with fail-safes incorporated. These design choices limit the scale of the product's operation but were necessary with the limited funds that the author has for using the OpenAI API.

- **Problem:** NLP approaches from spacy mistakenly identify sectors and countries from the prompt. **Implemented solutions:** Although this can never be fully avoided, the extracted sectors and countries are printed for the user to audit. Moreover, the NLP function is coded to exclude 'policy' and 'policies' from the sector, and the variable 'country' which was defined earlier will be ignored for the sector as well.
- **Problem:** Token length for documents is too long, hitting the limit of what the OpenAI API allows for. **Implemented solutions:**
    - Added a page length for documents that the user can adjust.
    - Documents are only added to the final list for GPT if the cumulative token limit does not exceed 16,385 for GPT 3.5 or 128,000 for GPT 4.0.
    - References, tables, charts, figures, and appendices are removed from the documents before they are passed to GPT.

Demonstration

Calling Python script

```
suppressMessages(library(reticulate)) # used for Python
suppressMessages(library(quanteda))
suppressMessages(library(quanteda.textstats))
suppressMessages(library(textTinyR))
suppressMessages(library(stm))
suppressMessages(library(topicmodels))
suppressMessages(library(tidytext))
suppressMessages(library(tidyverse))
suppressMessages(library(ldatuning))
suppressMessages(library(ggplot2))
suppressMessages(library(dotwhisker))
suppressMessages(library(readr))
```

For demonstration purposes, we will load a pre-built Python query that consistently returns two summarized documents.

```
# user_prompt = "I want to know more about the education policies of India
# between 2010 and 2020"
# documents_to_index = 5
# page_limit_per_document = 30

# NB. For knitted Rmd files, the input function from the Python file does not
# transfer over, therefore, different Python files that are pre-loaded
# with the parameters above are called to create a knitted file.

use_condaenv("C:/Users/mmsha/Anaconda3")
# Initializing the environment for Python.

py_run_file("C:/Users/mmsha/OneDrive/Desktop/The_Ghosts_in_the_Machine/R/Text_as_Data/Final_p
roject/Anaconda_nexus/llms_orchestration_v14_port2r_sk_final.py")
```

Inspecting the output

```
# Summary of all documents
response_all_documents <- readLines("response_all_documents.txt")
```

```
## Warning in readLines("response_all_documents.txt"): incomplete final line found
## on 'response_all_documents.txt'
```

```
# Creating a more readable line separator
cat(paste(response_all_documents, collapse = "\n"), "\n")
```

```
## **Summary of Education Policies from the Technical Education Quality Improvement Project I
II, World Bank**
##
## **Document 1: The World Bank Technical Education Quality Improvement Project III (P154523)
Report**
##
## - **Key Policy Insights:**
##   1. **Enhancement of Quality and Equity:** The project aims to improve the quality and eq
uity of engineering education in focus states by enhancing the capabilities of faculty and in
frastructure in educational institutions.
##   2. **Disbursement-Linked Indicators (DLIs):** Financial disbursements are linked to the
achievement of specific educational outcomes such as accreditation rates, faculty developmen
t, and student performance.
##   3. **Focus on Disadvantaged Groups:** There is a specific emphasis on increasing the enr
ollment and support for students from traditionally disadvantaged groups (Scheduled Castes/Sc
heduled Tribes and women).
##   4. **Digital and Pedagogical Reforms:** The project supports digital infrastructure impr
ovements and pedagogical training to ensure that institutions can offer high-quality educatio
n that meets current technological standards.
##   5. **Covid-19 Adjustments:** Due to disruptions caused by the Covid-19 pandemic, the pro
ject timelines and some goals were adjusted to accommodate delays in implementation.
##
## **Overall Summary:**
## The World Bank's Technical Education Quality Improvement Project III for India focuses on
enhancing the quality and equity of engineering education in selected states. It includes fin
ancial incentives linked to specific performance indicators like student accreditation rates
and faculty training. The project also places a strong emphasis on supporting disadvantaged g
roups and integrating digital tools and teaching methods into the curriculum. Adjustments hav
e been made to account for disruptions caused by the Covid-19 pandemic, ensuring that the pro
ject's goals can still be met within new timelines. This initiative reflects a comprehensive
approach to improving technical education through targeted reforms and support.
```

The summary has succeeded in dividing the policy recommendations between the two documents, and then combining them for a final summary. The LLM has also recognised that both documents are discussing the same policy initiative.

The raw text can also be examined, although only the top 20 lines should be viewed as otherwise the whole document would be too long.

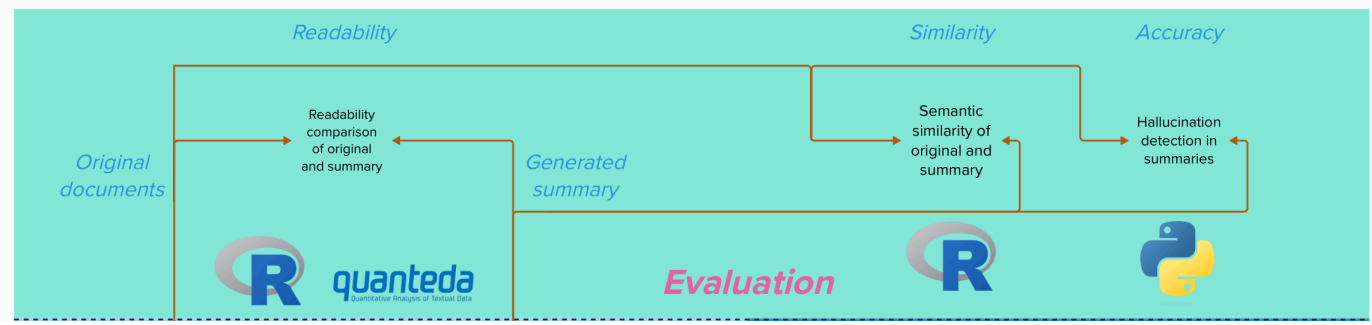
```
# Full text of all documents
raw_all_documents <- readLines("exported_text.txt")

# Top ten lines only
raw_all_documents_topten <- readLines("exported_text.txt", n = 20)

# Pasting the top ten lines
cat(paste(raw_all_documents_topten, collapse = "\n"), "\n")
```

```
## Document 1:
## The World Bank
## Technical Education Quality Improvement Project III (P154523)
## REPORT NO.: RES43311
## RESTRUCTURING PAPER
## ON A
## PROPOSED PROJECT RESTRUCTURING
## OF
## TECHNICAL EDUCATION QUALITY IMPROVEMENT PROJECT III
## APPROVED ON JUNE 24, 2016
## TO
## INDIA
## EDUCATION
## SOUTH ASIA
## Regional Vice President: Hartwig Schafer
## Country Director: Junaid Kamal Ahmad
## Regional Director: Lynne D. Sherburne-Benz
## Practice Manager/Manager: Mario Cristian Aedo Inostroza
## Task Team Leader(s): Namrata Raman Tognatta
## Public Disclosure Authorized Public Disclosure Authorized Public Disclosure Authorized Pub
lic Disclosure Authorized The World Bank
```

Evaluation



The evaluation segment of the architecture.

Readability

Using the Flesch–Kincaid readability test, we can see if the LLM was successful at summarising the key takeaways from the papers in more accessible language.

```
raw_all_documents <- paste(raw_all_documents, collapse = " ")

response_all_documents <- paste(response_all_documents, collapse = " ")

response_all_documents_corpus <- corpus(response_all_documents)

raw_all_documents_corpus <- corpus(raw_all_documents)

Flesch_reading_ease_table <- data.frame(
  Score = c("100.00–90.00", "90.0–80.0", "80.0–70.0", "70.0–60.0", "60.0–50.0", "50.0–30.0",
"30.0–10.0", "10.0–0.0"),
  `School level (US)` = c("5th grade", "6th grade", "7th grade",
                          "8th & 9th grade", "10th to 12th grade",
                          "College", "College graduate", "Professional"),
  Notes = c(
    "Very easy to read. Easily understood by an average 11-year-old student.",
    "Easy to read. Conversational English for consumers.",
    "Fairly easy to read.",
    "Plain English. Easily understood by 13- to 15-year-old students.",
    "Fairly difficult to read.",
    "Difficult to read.",
    "Very difficult to read. Best understood by university graduates.",
    "Extremely difficult to read. Best understood by university graduates."
  )
)

categorize_score <- function(score) {
  if (score >= 90.0) {
    return("5th grade")
  } else if (score >= 80.0) {
    return("6th grade")
  } else if (score >= 70.0) {
    return("7th grade")
  } else if (score >= 60.0) {
    return("8th & 9th grade")
  } else if (score >= 50.0) {
    return("10th to 12th grade")
  } else if (score >= 30.0) {
    return("College")
  } else if (score >= 10.0) {
    return("College graduate")
  } else {
    return("Professional")
  }
}
```

```r
# Summary response score of all documents
readability_stats_response <- textstat_readability(
  response_all_documents_corpus, measure = "Flesch.Kincaid")

flesch_score_all_summary <- readability_stats_response$Flesch.Kincaid

cat("Flesch-Kincaid Grade Level score:", flesch_score_all_summary)

school_level_response <- categorize_score(flesch_score_all_summary)


# Text score of all documents
readability_stats <- textstat_readability(
  raw_all_documents_corpus, measure = "Flesch.Kincaid")

flesch_score_raw_text <- readability_stats$Flesch.Kincaid

cat("Flesch-Kincaid Grade Level score:", flesch_score_raw_text)

school_level_raw <- categorize_score(flesch_score_raw_text)




## Flesch scores compared ----

compare_Flesch_scores_function <- function(flesch_score_all_summary,
                                           flesch_score_raw_text) {
  level1 <- categorize_score(flesch_score_all_summary)
  level2 <- categorize_score(flesch_score_raw_text)

  if (level1 == level2) {
    return("The scores are in the same category.")
  } else {
    return(paste("Score for the summaries is", flesch_score_all_summary,
                 "and is categorized as", level1,
                 "and score", flesch_score_raw_text,
                 "is categorized as", level2))
  }
}

compare_Flesch_scores_result <- compare_Flesch_scores_function(
  flesch_score_all_summary, flesch_score_raw_text)
```

```r
print(paste("Score for summary response is", flesch_score_all_summary,
            ", categorized as", school_level_response))
```

```
## [1] "Score for summary response is 15.6699047619048 , categorized as College graduate"
```

```r
print(paste("Score for raw text is", flesch_score_raw_text, ", categorized as", school_level_
raw))
```

```
## [1] "Score for raw text is 22.6948045165843 , categorized as College graduate"
```

```
print(compare_Flesch_scores_result)
```

```
## [1] "The scores are in the same category."
```

While the readability score for the summaries is slightly higher than the raw text, both are still categorized at 'college graduate' level. This highlights the need for further prompt-engineering to deliver a more accessible summary.

Similarity

To understand whether our summary is using the same language as the original document, we can apply a cosine similarity approach to derive the cosine of the angles between the vectors (Mitts, Lecture 4, 2024). For this evaluation measure, we need to only capture the most relevant words / tokens. Therefore, we can apply a number of pre-processing steps including removing punctuation, stop words, numbers, and lower-casing all letters (Mitts, Lecture 2, 2024).

```
# Text is pre-processed to only reflect the most relevant words
response_all_documents_tokens <- tokens(response_all_documents,
                                        remove_numbers = TRUE,
                                        remove_punct = TRUE,
                                        remove_url = TRUE,
                                        remove_symbols = TRUE)

raw_all_documents_tokens <- tokens(raw_all_documents,
                                   remove_numbers = TRUE,
                                   remove_punct = TRUE,
                                   remove_url = TRUE,
                                   remove_symbols = TRUE)

response_all_documents_dfm <- dfm(response_all_documents_tokens,
                                  tolower = TRUE)

raw_all_documents_dfm <- dfm(raw_all_documents_tokens,
                             tolower = TRUE)

response_all_documents_dfm <- dfm_remove(response_all_documents_dfm,
                                         stopwords("english"))

raw_all_documents_dfm <- dfm_remove(raw_all_documents_dfm,
                                         stopwords("english"))


cosine_similarities = textstat_simil(x = response_all_documents_dfm,
                                     y = raw_all_documents_dfm,
                                     method = "cosine",
                                     margin = "documents")
```

```
print(cosine_similarities)
```

```
## textstat_simil object; method = "cosine"
##       text1
## text1 0.361
```

```
# -1 = perfect dissimilarity
# 1 = perfect similarity
```

The value is greater than 0, meaning that it is more similar than dissimilar, but this is still a low similarity rating and it is likely driven by the different number of top features that are used in the summary versus the response. We can investigate this further using the topfeatures() function.

```
top_ten_features_raw <- topfeatures(raw_all_documents_dfm, n=10)
top_ten_features_response <- topfeatures(response_all_documents_dfm, n=10)

print(top_ten_features_raw)
```

```
##        states         focus       project participating    institutes
##           885           285           246           184           172
##            st    accredited     education       quality         least
##           160           143           129           111           111
```

```
print(top_ten_features_response)
```

```
##   education     project     quality   technical improvement         iii
##           8           7           6           4           3           3
##       world      equity     faculty    specific
##           3           3           3           3
```

This confirms our earlier suspicion. A summary is naturally going to focus on the words identified as the most important for policy recommendations, particularly in relation to the sector (education in this case). Further examinations of similarity can be explored.

Latent Dirichlet Allocation (LDA) assumes a document has multiple topics, with each topic having a mixture of words written about it with each word having a higher probability of being associated with it. Building up from words, each document then has a different proportion of topics. The motivation is that there's a 'hidden' structure of topics reflected in the 'observed' documents and words - reversing the generative process (Mitts, Lecture 7, 2024; Griffiths and Steyvers, M., 2004).

Further similarity exercises were conducted outside of this Rmd through LDA, but the small length of the LLM summaries (as indicated by the frequency of the topfeatures above) were not conducive to this method. Moreover, and unfortunately, topic models and other clustering exercises can only give a sense of the similarity in language between the summaries and raw text. They do not offer a deeper understanding of whether the language was summarised accurately. To investigate this, we can turn to packages designed to investigate the hallucination level of LLM outputs based on text they have summarised.

Accuracy

The Hughes Hallucination Evaluation Model (HHEM) is an open source model, created by Vectara, for detecting hallucinations in LLMs. Their GitHub notes that it is particularly useful in the context of building RAG applications such as this product (Hughes and Vectara, 2024).

This model was trained using SentenceTransformers Cross-Encoder class (a Python framework for text embeddings that is particularly useful for semantic textual similarity). The model outputs a probabilitity from 0 to 1, 0 being a hallucination and 1 being factually consistent.

For this product, each document's individual summary is compared against that document's raw text to understand the level of factual consistency in the former.

Unfortunately, this model does not run on R as of yet. Therefore, another Python file with the code for this model needs to be executed with reference to the outputted text and summaries from the previous steps.

```
use_condaenv("C:/Users/mmsha/Anaconda3")
# Initializing the environment for Python.

py_run_file("C:/Users/mmsha/OneDrive/Desktop/The_Ghosts_in_the_Machine/R/Text_as_Data/Final_p
roject/Anaconda_nexus/llms_orchestration_v14_hallucination_evaluation_sk.py")
```

The Python file saves the outputted scores in a 'pickle' file so that we can then import the list of values into R.

```
# Loading the pickle module from Python
pickle <- import("pickle")

# Reading the binary content of the pickle file into readable format for R
binary_content <- readBin("scores_summaries_list.pkl", "raw", file.info("scores_summaries_lis
t.pkl")$size)

# Creating a list from the binary content
scores_summaries_list <- pickle$loads(binary_content)

# Loop function for deriving accuracy result for however many documents
# have been fed through the LLM orchestration
for (i in seq_along(scores_summaries_list)) {
  cat("Accuracy result for document", i, ":", scores_summaries_list[[i]], "\n")
}
```

```
## Accuracy result for document 1 : 0.7550119
## Accuracy result for document 2 : 0.9622746
```

The accuracy results for both of our documents are fairly high, particularly for document 2. This is positive news that the inferences made by the LLM being factually consistent, and are an indication of the potential of the RAG approach for future applications in policymaking.

Conclusion

Key takeaways

Drawing from the evaluations above, and noting the limitations of drawing from a single exercise, we can make a few inferences as to the strengths and weaknesses of this product.

- Strengths
    - Rapid collation and summarisation of multiple documents.
    - Relatively strong performances in both similarity and accuracy.

    - The product successfully references sources through RAG, improving the reliability of approaches such as these for policymaking tools.
- Weaknesses
    - Token limit in the OpenAI API reduces the scale of operation. This can be ameliorated with further funding or utilizing a localized LLM such as Llama 3 with powerful GPUs for conducting inference.
    - The summaries may still be too general to be of direct use for the policymaker. Additional methods to retrain the LLM through fine-tuning may be necessary to improve precision (see below).
    - The readability improvements are quite low and this also points towards the need for further fine-tuning or prompt engineering.

As LLM becomes more widely used through all aspects of life, their applications as tools for policymaking will also become more wide-spread. Methods such as RAG may alleviate some of the weaknesses of narrow products such as this, but more robust adaptations will be needed to make them fit for purpose.

These potential adaptations are outlined in the next section.

Potential improvements

In future iterations of this product, several additional functionalities can be scoped to improve the accessibility and quality of output:

- Adding a user interface that is outside of R. This can be hosted on a webpage using Flask or similar packages.
- Adding the capacity for non-English languages to be processed by the model. The World Bank API is capable of handling non-English languages with many such documents also being stored in their database. To achieve this, the NLP packages and methodologies must incorporate non-English datasets whilst the LLM segment is already capable of multilingual processing.
- Additional prompt-engineering, or employing other methods, for improving the readability scores of the summaries as compared to the raw text.
- Exploring ways of conducting topic model analysis to understand the similarity in topics between the document summaries and their raw versions.
- Fine-tuning, alongside RAG, is one of the proposed approaches for adapting LLMs to be more reliable. This process relies on retraining the model to a limited extent, ‘fine-tuning’ it, so that the underlying word embeddings are more aligned to the specific context (Miller, 2023). This is possible with the OpenAI API but requires additional financing.

Acknowledgements

Original LLM Orchestration code written in Python within group project. Thanks go to team-members for contributing sections of the initial code: Geetika Malhotra (MPA ’24), Jinyu Qi (MPA ’24), Shumpei Watanbe (MPA ’24)

This project was further-developed thanks to methodologies taught by Professor Tamar Mitts in Text as Data (INAF U6514) at Columbia University’s School of International and Public Affairs. Core knowledge about text as data structure, management, and evaluation methods through the Quanteda package, in particular, were learned through that class.

Thanks also go to Wei Lu, Lead Software Engineer at the World Bank’s Development Impact (DIME) department, who offered valuable feedback on improving the functionalities of this product.

Bibliography

- Griffiths, T. L., & Steyvers, M. 2004. Finding scientific topics. Proceedings of the National Academy of Sciences of the United States of America, 101 Suppl 1(Suppl 1), 5228–5235.

https://doi.org/10.1073/pnas.0307752101 (https://doi.org/10.1073/pnas.0307752101)
- Hughes, Simon Mark, and Vectara. 2024. “Hughes Hallucination Evaluation Model”. *Hugging Face*. https://huggingface.co/vectara/hallucination_evaluation_model (https://huggingface.co/vectara/hallucination_evaluation_model)
- Lewis, P., Oguz, B., Rinott, R., Riedel, S., & Schwenk, H. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020). https://arxiv.org/abs/2005.11401 (https://arxiv.org/abs/2005.11401)
- Liao, Q. V., & Wortman Vaughan, J. (2024). AI transparency in the age of LLMs: A human-centered research roadmap. Harvard Data Science Review. https://hdsr.mitpress.mit.edu/pub/aelql9qy (https://hdsr.mitpress.mit.edu/pub/aelql9qy)
- Miller, Katharine. October 2023. “LLMs Aren’t Ready for Prime Time. Fixing Them Will Be Hard.” *Stanford University’s Human-Centerd Artificial Intelligence*. https://hai.stanford.edu/news/llms-arent-ready-prime-time-fixing-them-will-be-hard (https://hai.stanford.edu/news/llms-arent-ready-prime-time-fixing-them-will-be-hard)
- Mitts, Tamar. 2024. “Lectures on Text as Data”. Columbia University’s School of International and Public Affairs