



Department of Computer Science

THE NATIONAL UNIVERSITY OF SINGAPORE

CP5105 Computing Capstone Project

**AI-Powered Product and Service Review Summarizer and
Ranking System Application (Project ID: 425)**

Submitted By: Yap Wei Xuan (E0310021)

Student ID: A0183226H

Prepared for: Prof. Praneeth Shalinda Adikari A. A.

GitHub Link: <https://github.com/YapWeiXuan/CP5105-AI-Powered-Product-and-Service-Review-Summarizer-and-Ranking-System/tree/main>

This report is submitted in partial fulfilment for the requirement for the Master of Computing (General Track)

Acknowledgements

I would like to express my deepest appreciation to Professor Praneeth Shalinda Adikari A. A. for the support and guidance in not just the project requirements but also the mental support through encouragement. I'd also like to acknowledge Professor Prasanna Karthik VAIRAM for assisting with the final presentation judging.

Table of Contents

Acknowledgements	2
Table of Contents	3
1. Abstract	5
2. Introduction	5
3. Project Methodology.....	7
4. Software Architecture.....	7
4.1. FrontEnd Framework: Angular (TypeScript).....	8
4.2. BackEnd Framework: FastAPI (Python).....	9
4.3. Database: MongoDB (Non-Relational Database)	9
4.4. Web Scraping (Google Search Api & BeautifulSoup)	9
4.5. LLM Platform: Ollama.....	10
5. Literature Review	11
5.1. Research Paper: Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. (Authors: Banghao Chen, Zhaofeng Zhang, Nicolas Langren, Shengxin Zhu).....	11
5.2. Paper relevance to the project:	13
6. Datasets used for Evaluation and Optimization	14
6.1. Summarizer LLM Evaluation Dataset:.....	14
6.2. Ranking List LLM Evaluation Dataset:.....	14
7. Evaluation Metrics	15
7.1. Recall-Oriented Understudy for Gisting Evaluation (ROUGE).....	15
7.2. Bidirectional Encoder Representations from Transformers (BERT) Score	16
7.3. Total Ranking Scores and Ranking Scores Variance (For Ranking List Generation LLM) ..	17
8. Summarizer LLM Evaluation (Results and Discussions).....	19
8.1. Results	21
8.2. Discussion and Insights	22
9. Ranking List LLM Evaluation	22
9.1. Evaluation Part 1	23
9.1.1. Methodology:	23
9.1.2. Results.....	25
9.1.3. Discussion and Insights	27
9.2. Evaluation Part 2	28
9.2.1. Methodology:	28

9.2.2. Results.....	30
9.2.3. Discussion and Insights	33
10. Model Optimization	35
10.1. Methodology:.....	35
10.2. Sensitivity Analysis.....	36
10.2.1. Results.....	36
10.2.2. Discussion and Insights	37
10.3. Stepwise Regression	37
10.3.1. Results.....	37
10.3.2. Discussion and Insights	39
11. Web Application Design and Usage	39
12. Further Improvements	42
12.1. LLM Future Improvements	42
12.2. Web Application Future Improvements	43
13. Conclusion	44
14. References.....	45

1. Abstract

This project aims to build an AI-Powered Product and Service Review Summarizer and Ranking System Application that receives an input query from the user and generates a product/service ranking list with summarized descriptions for each product/service on the list. The research paper, “Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review” (Banghao Chen, 2023), will be studied and referenced for the various prompt engineering techniques that can be utilized to improve the performance of Large Language Models (LLMs). To evaluate a LLM response generation performance, evaluation metrics such as the ROUGE score, BERT score, Total Ranking Scores, Ranking Score Variance etc., will be used to judge the quality of the LLM’s generated response. Among the various open-source LLM, results show that the Llama 3.1 model have the best summarization capabilities and is used as the Summarizer LLM. For the Ranking List LLM, various prompt engineering techniques (such as Zero Shot , One Shot , Few Shot and Chain-of-Thought) and input variants were explored and measured against one another using the evaluation metrics to generate the most accurate and consistent Ranking List. Sensitivity Analysis and Stepwise Regression are then utilized to further optimize the Ranking List LLM parameters. From the results, Zero Shot prompting technique with the parameter configuration (Temperature = 0.8, Top P = 0.9, Top K = 36) allows the LLM to generate the most accurate and consistent Ranking List. Next, the web application is built using the Angular Framework (Typescript) for the frontend, FastApi (Python) for the backend and MySQL for the database to showcase the practical use of the optimized LLM as an AI-Powered Product and Service Review Summarizer and Ranking System. Finally, future improvements to the LLM and web application are discussed to improve upon the current project.

GitHub Link: <https://github.com/YapWeiXuan/CP5105-AI-Powered-Product-and-Service-Review-Summarizer-and-Ranking-System/tree/main>

2. Introduction

The creations of the first artificial neural network, the Mark 1 Perceptron, by Frank Rosenblatt in the 1958 and the first Natural Language Processing (NLP) program , ELIZA, by Joseph Weizenbaum in 1966 were some of the first few milestones in the development of neural networks and LLM (Foote, 2023). Unfortunately, the computational power back then was not

powerful enough for general practical usage of neural networks. However, following Moore's Law, in which the number of transistors on an integrated circuit (IC) micro-chip roughly doubles every 2 years (Tardi, 2024) , the general computational power has vastly increased in the 20th century. Fast forward to today, BERT (Bidirectional Encoder Representations from Transformers) was announced by Google in 2019, which is a two-directional, 340-million parameter model that can derive the text's context. OpenAI would subsequently announce the widely popular ChatGPT (which utilize GPT-3 with 175 billion parameters) in November 2022, setting the standard for LLMs while bringing global attention to it (The history, timeline, and future of LLMs, 2023).

In the modern day, the online shopping scene has been steadily increasing in popularity over the years with the number of U.S online shoppers projected to increase from 268 million in 2022 to 285 million in 2025 and the amount of global e-commerce sales is projected to reach \$58.74 trillion by 2028 (Taheer, 2025). Moreover, the number of online stores in 2021 is between 12 to 25 million with this number ever increasing with time (How Many Online Stores Are There? , 2021). Hence, with the increasing amount of people shopping online and the increasing number of goods/services being sold online, there is an increasingly overwhelming amount of product and service reviews for the average consumer to absorb before making his/her decision.

Thus, this project (Capstone Project AI-Powered Product and Service Review Summarizer and Ranking System) aims to address the aforementioned issue by developing an application that provides a ranking list based on the query inputted by the user. Instead of going through multiple online search engines and scouring many online websites to look for information regarding the product/service the user is looking for, the user can input his/her query into the application and the application will return a ranking list of the top product/services with their respective detail summaries based on the top web articles. As such, the project objectives are as follows:

1. Analyse and summarize product/service reviews from a range of sources into insightful and concise descriptions.
2. Compute the ranking order of the product/service based on various factors (number of items from the same brand, ranking given by the source etc.)

3. The application output should be in real-time whereby the moment the user queries a product/service, the application should start retrieving the necessary information, summarization of the retrieved information and return the ranking calculation promptly.
4. The application output should consider information from a wide range of sources.
5. The user interface of the application should be user-friendly, have a structured and intuitive layout.

3. Project Methodology

The design, experimentation, analysis and implementation processes towards the completion of the project deliverables are explained chronologically in the process flow below in Fig.1:

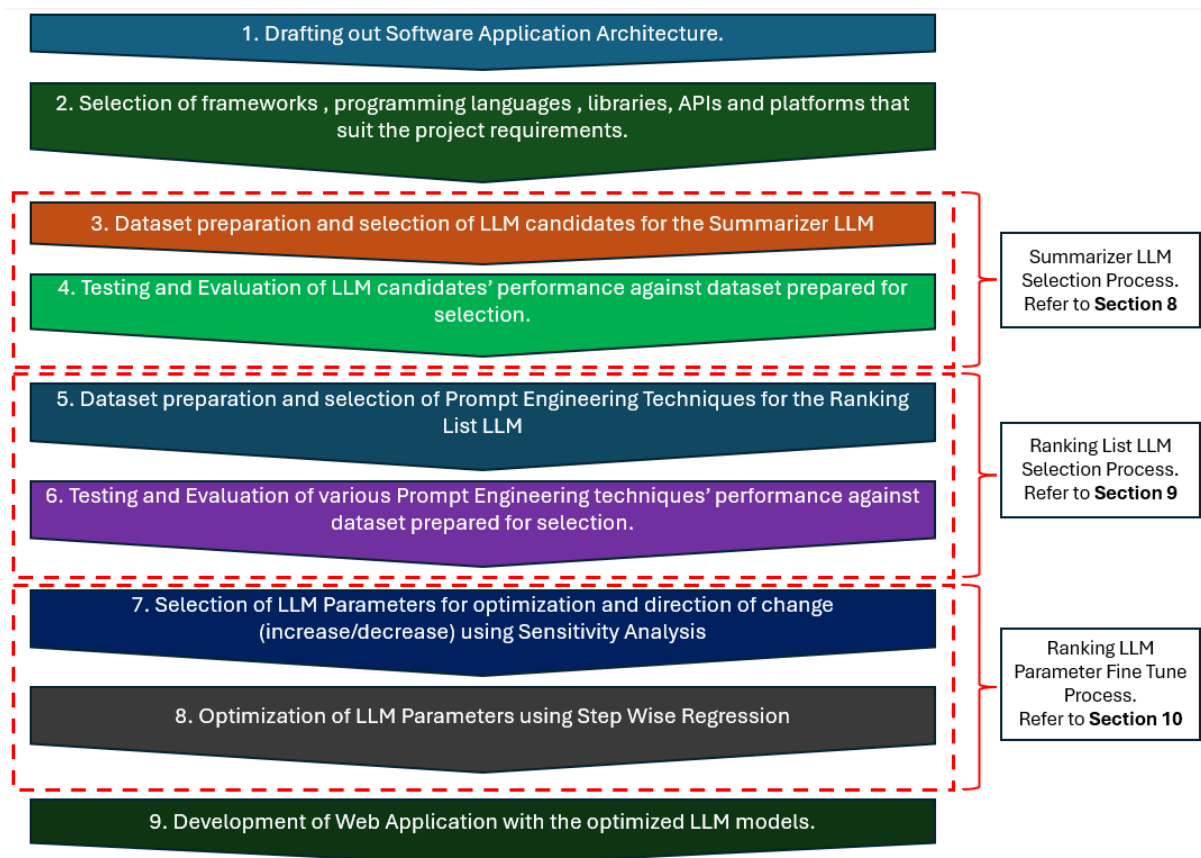


Fig.1: Project Methodology Process Flow

4. Software Architecture

The software architecture, choice of software/frameworks/database/programming language, response generation process and user's query processing steps are all illustrated in Fig.2.

Moreover, the reasons for the aforementioned choices are explained in the following sub-sections.

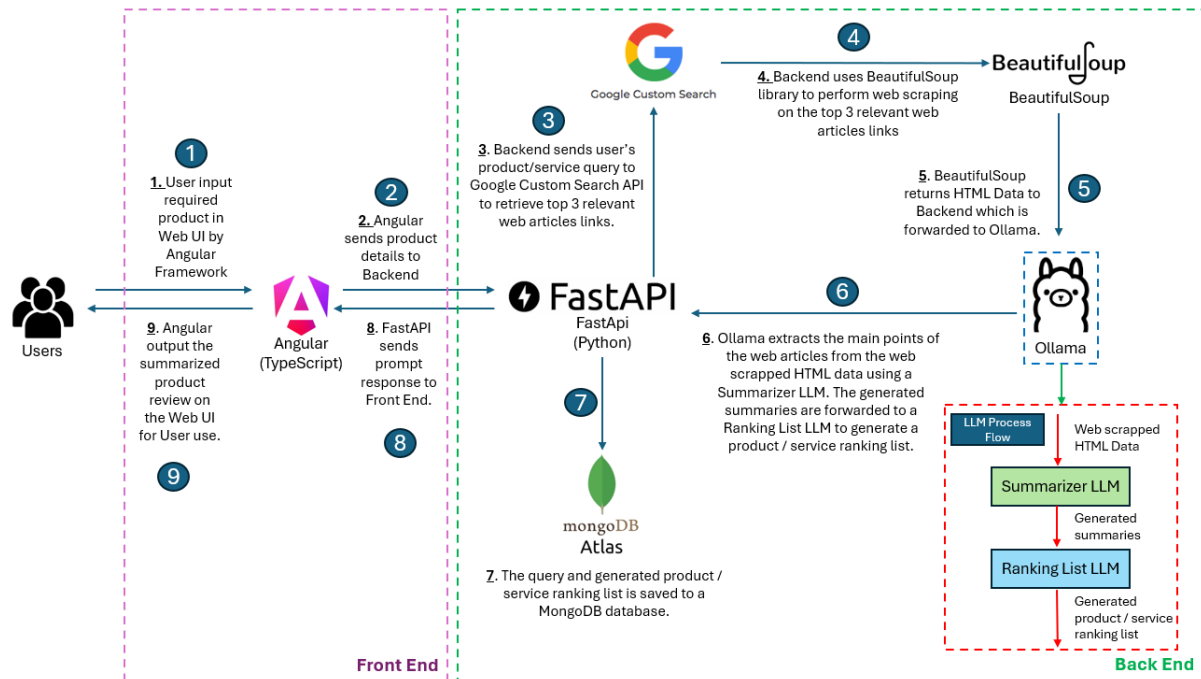


Fig.2: Project Web Application Software Architecture and Data Flow

4.1. FrontEnd Framework: Angular (TypeScript)

Angular is a component-based Model-View-Controller (MVC) and Model-View-View Model (MVVM) architecture framework that is built on TypeScript while being developed and maintained by Google. It is usually utilized to create Single-Page Applications (SPAs) and other complex dynamic web interfaces. Some of its key features include: 2-way data binding to reflect the latest model state instantaneously by automatically synchronizing the model and the view, components, directives to extend HTML's behaviour (Top 10 Angular features you should know, 2024).

Due to multiple relevant advantages Angular offers for this project, it is chosen over other frontend frameworks such as React. Unlike React, Angular has more functions already built into its library such as HTTP requests and testing, this eliminates the need to install 3rd party libraries to enable the required functions. Moreover, by automatically separating the rendering and data handling code, Angular provides better code clarity and readability unlike React, which have both data handling and rendering logic mixed into the HTML code. On top of that, as the project only requires a SPA (with a User Interface (UI) being a search bar for the User

to enter the query and a table to reflect the backend LLM's response), the aforementioned Angular's key features makes it the best framework for the project (Friesen, 2019).

4.2. BackEnd Framework: FastAPI (Python)

FastAPI is a high-performance and modern web framework to develop Python based REST APIs. It is one of the fastest python frameworks while reducing ~40% of induced bugs due to its coding style. Some of its key features include Asynchronous Server Gateway Interface (ASGI) to support asynchronous tasks, automatic API documentation with JSON schema

One of the main reasons why FastAPI is chosen over the other Python frameworks is due to its ASGI supporting asynchronous tasks (Python FastAPI vs Flask: A Detailed Comparison, n.d.). The project web application backend performs web scraping based on the user query and then passes these HTML text to the LLM models for summarization and ranking. These tasks typically require a significant amount of time that might lead to the web application handling multiple requests simultaneously and user frustration should they have to wait too long for the web application to produce the results. Thus, FastAPI's speed in handling asynchronous tasks is key in reducing user's frustration while waiting for the lengthy response generation by the LLMs in the backend, making it the choice backend framework for this project.

4.3. Database: MongoDB (Non-Relational Database)

Each query is stored as a document object which have the following data structure: (query: string, html_links: string[], html_text: string[], generated_response: string). Non-relational database is chosen over a relational database in this project's context due to easier horizontal scalability as each query document object is self-contained (Difference between RDBMS and MongoDB, 2025).

4.4. Web Scraping (Google Search Api & BeautifulSoup)

Google Search Api is chosen due to its prevalence and simple API usage. Similarly, BeautifulSoup is chosen as the project only requires a simple HTML scraping library for web articles. For the Google Search API, only the first page of search results is chosen as it consists of 10 of the most relevant links which is more than sufficient for this project's target number of links used to be 3. The number of links is 3 as any more would incur a risk of including web pages that have no relevance to the user's input query as shown in Fig.3. Moreover, having

more page links would lead to increased wait times for the user which could lead to their frustration as simply utilizing 3 web articles texts for the Ranking List generation requires ~4 mins for it be generated. Furthermore, it is possible for certain links with restrictions to return Errors such as “The read operation timed out” or “HTTP Error 403: Forbidden” and thus can’t be scrapped which is why there is a need for the number of links fetched to be more than the number of links utilized for the ranking list generation. This increased risk of the backend processing irrelevant links and taking a longer processing time is why out of the 10 web article links fetched, only 3 of which will be processed by the backend.

Google Search API results for Query (“What are the best sunscreen brands?”) :


Fetching 3 Links :

```
"query": "What are the best sunscreen brands?",
"links": [
  "https://www.ewg.org/sunscreen/",
  "https://www.goodhousekeeping.com/beauty/anti-aging/g1288/best-sunscreens/",
  "https://people.com/best-sunscreens-8636289"
],
```

Fetching 5 Links :

```
"query": "What are the best sunscreen brands?",
"links": [
  "https://www.ewg.org/sunscreen/",
  "https://www.goodhousekeeping.com/beauty/anti-aging/g1288/best-sunscreens/",
  "https://people.com/best-sunscreens-8636289",
  "https://nymag.com/strategist/article/best-sunscreen-for-face.html",
  "https://www.vacation.inc/"
],
```

Relevant Article



Irrelevant Article

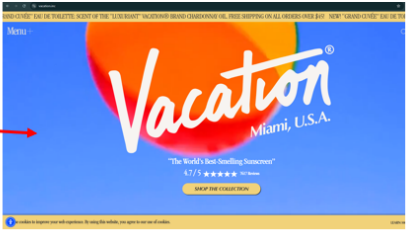


Diagram illustrating Google Search API results for the query “What are the best sunscreen brands?”. The results are categorized into two groups: Relevant Article and Irrelevant Article. The Relevant Article is a link to a People magazine article titled “The 13 Best Sunscreens from Neutrogena, Supergoop, and More, Tested on the Beach”. The Irrelevant Article is a link to a Vacation Miami, U.S.A. advertisement. The diagram shows the API results for fetching 3 links and 5 links, with red boxes highlighting the relevant and irrelevant links respectively.

Fig.3: Google Search API results comparisons for Query (“What are the best sunscreen brands?”)

4.5. LLM Platform: Ollama

Ollama is a user-friendly open-source platform that runs LLMs directly on a local machine environment, eliminating the dependency for cloud services. On top of that, Ollama’s diverse LLM library that’s constantly updating allows one to access many LLMs ranging from the recent popular “deepseek-r1” reasoning model to other widely popular models such as Meta’s Llama 3.1 the while providing the various Parameter Count versions for each model (Modeling Ollama, 2023) (What is Ollama? Understanding how it works, main features and models, 2024). For example, Ollama provides the 1.5/7/8/14/32/70/671 billion Parameter Count versions for “deepseek-r1” (Ollama, n.d.). All these factors make the development and

optimization of LLMs easier on a local machine, making it a choice LLM platform for this project.

5.Literature Review

5.1. Research Paper: Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. (Authors: Banghao Chen, Zhaofeng Zhang, Nicolas Langren, Shengxin Zhu)

The research review paper done by the personnel from the Guangdong Provincial Key Laboratory of Interdisciplinary Research and Application for Data Science, BNU-HKBU United International College in Guangdong, China explores the use of various prompt engineering techniques to fully capitalize on the potential of Large Language Models (LLM) (Banghao Chen, 2023). GPT-4 (developed by OpenAI) is utilized through the paper's research for testing, generating results and examples for comparison.

The paper first illustrates the proper method for writing good prompts for the LLM by adhering to the following guidelines:

1. **Be clear and precise:** Prompts should be unambiguous and specific
2. **Using role-prompting:** Include a role for the LLM to play in the prompt
3. **Use of triple quotes:** To separate different portions of the prompt and to sum up multi-line strings.
4. **One-shot/Few shot prompting:** To include different number of prompt/answers examples to provide more context and guidance for the LLM.
5. **Fine tuning LLMs common hyper parameters:** Temperature and Top P, whereby temperature refers to the randomness of the LLM output while Top P refers to the sampling method to add randomness to the LLM output.

Advanced prompting methods were also discussed, such examples include:

- **Chain-of-Thought:** The technique involves adding intermediate reasoning steps to the prompt to guide the LLM's output. This provides a clear structure for the LLM to follow for its reasoning process.
- **Self-Consistency:** This technique aims to resolve the model's limitations in showing consistent results for reasoning tasks. This is accomplished by generating a wide range

of differing reasoning paths and selecting the most consistent answer out of the final answer set.

The paper also explains the use of textual gradients to optimize prompts (by adapting the concept of gradient descent) and black-box prompt optimization to align human intent without model retraining. An emerging framework to reduce hallucinations in LLM's generated responses is the Retrieval Augmentation which involves including the latest external knowledge into the model's input as the model's training data might lack up-to-date knowledge. To access the efficacy of the various prompt methods, the paper also explains the following methods:

1. **Subjective evaluation:** This method involves human evaluators to assess a LLM's generated output. It is typically used when it is hard to represent the assessment of the model's output in datasets or abstract concepts such as summarization. In a research conducted by researchers from Princeton University and Google DeepMind mentioned in the paper, the researchers used human judgement to compare the generated response from the model using a "tree-of-thought" approach against outputs from other methods to judge its creative writing.
2. **Objective evaluation:** This involves using algorithms/metrics to gauge the LLM's output quality, conducting tests that quantitatively assess the various LLM's efficacy. Examples of such metrics include the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score and the BERTScore. Benchmarks, which are datasets containing instructions for the model to complete, are usually used in objective evaluation to provide a more holistic dataset for the LLM to work on. Such benchmarks include:
 - a. **Math Word Problems (MWP):** Hybrid Math Word Problems dataset (HMWP) and Simple Variations on Arithmetic Math word Problems (SVAMP) are benchmarks used to test a model's capability to resolve numerical-related questions.
 - b. **Question Answering (QA) tasks:** Massive Multitask Language Understanding (MMLU) and Question Answering with Long Input Text, Yes (QuALITY) are examples of benchmarks that assess a LLM's reasoning capability and basic intelligence ability.

The paper also explores the practical use of LLMs in real-world applications. In the field of education, LLMs have shown promise in the creation of personalized learning environments for young students of varying learning needs using prompt engineering and even help reduce teachers' workload by helping to assess students' answer scripts. In the field of mathematics, Artificial Intelligence Generated Content (AIGC) tools have shown good promise in resolving word-based math problems in the GSM8K dataset and their reasoning performance are further enhanced with the use of prompt engineering.

Lastly, adversarial attacks, which is the manipulation of input data to cause the model to be trained wrongly and make unreliable predictions, are assessed in the paper. This typically include data poisoning where the inaccurate data is injected into the model's training dataset to distort its learning, backdoor attacks where the model's training is manipulated such that when a specific prompt is used, the model will generate a response that poses a security risk. These attacks can be identified early by using adversarial example generation technique. This is done by creating input data with the intention to mislead the model, this allows the researchers to identify any potential security risks the model might have before deployment and resolve them early.

5.2. Paper relevance to the project:

The prompt engineering/model optimization techniques and the evaluation methods discussed in the report will be mainly used in the project. Prompt engineering techniques such as zero-shot/few-shot and Chain-Of-Thought (COT) will be evaluated during the model development to identify which technique will generate the best quality responses for text summarization and product/service ranking. Moreover, the Temperature, Top P, Top K hyperparameters will be further tuned to achieve better results with the evaluation metrics discussed in Section 7 and Section 10.1.

To evaluate the model performance while testing the aforementioned various prompt engineering techniques and hyperparameters, subject evaluation and objective evaluation will be deployed. As explained in the paper, ROUGE and BERTScore metrics will be used to quantitatively assess the quality of the model's generated summary and product/service ranking list against the model summary and model product/service ranking list. However, as it is possible for the model to generate good summaries of each product in the list, but the placement

of each product/service's ranking could be wrong, subjective evaluation will also be used to analyse the model's generated response for the product/service ranking list. This is done with a human's current real-world domain knowledge to judge whether the generated ranking list is truthful based on the information collected from web scraping.

6. Datasets used for Evaluation and Optimization

6.1. Summarizer LLM Evaluation Dataset:

- Name: Hugging Face: sujayC66/text_summarization_512_length_1_4000
- Details: It consists of ~3300 data rows with each row containing a text with ~1800 characters to be summarized and a model summary text with ~300 characters. Table 1 illustrates the dataset schema.

Table 1: Summarizer LLM Evaluation Dataset Schema

Input Text	Model Summary of the Input Text
Example: LONDON - Hunting PLC (LSE: HTG), a precision engineering group, ... 12% rise since the end of the third quarter of 2023 ... net debt position of approximately zero, consistent with the outlook provided in October 2023. (~1800 characters)	Example: Hunting PLC's 2023 financial performance aligns with expectations, with EBITDA estimated between \$96 million and \$100 million and revenue between \$925 million and \$930 million. The company's sales order book has increased by 12% since Q3 2023, and it expects to reach a net debt position of approximately zero by the end of 2023. (~300 characters)

6.2. Ranking List LLM Evaluation Dataset:

- Name: Semi Manually Created Ranking List Dataset
- Details: Due to the absence of a specific dataset that meets the specific requirements to evaluate the ranking list LLM, a dataset must be created that suits the evaluation requirements. The dataset consists of 15 rows, 5 of which are to be used as prompt examples while the remaining 10 are to be for testing purposes. Each data row contains a user query, a modified generated summary for 3 web articles with article labelling using Llama 3.1, a modified generated summary for 3 web articles WITHOUT article labelling using Llama 3.1, the model ranking list with the rankings determined manually (each item ranking is decided based on the frequency of the item's name appearance in the modified generated summary, reason for doing so is explained in Section 9) and an indicator on whether a data row is to be used as prompt example or for prompt testing as shown in Table 2.

Table 2: Ranking List LLM Evaluation Dataset Schema

Query	Modified 1 Generated Summary	Modified Multiple Generated Summaries	Model Ranking List Output	Prompt
- User text input Example: “What are the top car models in 2024?”	- 1 text string containing the concatenated 3 summarized web article - Summarization is done with selected LLM from Section 8 . Example: “The automotive industry is constantly evolving ... The auto industry has witnessed significant advancements ... The automotive industry is constantly evolving ...”	- 1 text string containing the concatenated 3 summarized web article with labelled articles . - Summarization is done with selected LLM from Section 8 . Example: “ Article #1: The automotive industry is constantly evolving ... Article #2: The auto industry has witnessed significant advancements ... Article #3: The automotive industry is constantly evolving ...”	- 1 text string containing the model ranking list with the summarized descriptions for each product/service. - Summarization is done with selected LLM from Section 8 . Example: “Here is the ranking list for the top car models in 2024 with their summaries from most to least good: **#1. Toyota Camry:** "The Toyota Camry is expected to be one of the top-selling car models in 2024 ... **#2. Honda Civic Type R:** "The Honda Civic Type R is a high-performance sports car ... **#3. Hyundai Ioniq 6:** "The Hyundai Ioniq 6 has gained recognition ...”	- An indicator of whether a data row is to be used as prompt example for Few Shot Prompting Technique or for LLM testing.

7. Evaluation Metrics

To investigate the performance difference between the various LLMs, prompting techniques and hyperparameters’ values on the datasets, a suite of evaluation metrics is deployed to obtain a holistic assessment of an LLM’s response generation performance.

7.1. Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

The ROUGE score is usually utilized in evaluating AI generated text summarization/translation against the model reference summary to investigate the LLM’s summarization/translation capability. For the summarization purposes of this project, it is used to calculate the number of overlapping n-grams (word sequences) that are found in both the model reference and the LLM-generated summary (An intro to ROUGE, and how to use it to evaluate summaries, 2017). It is further split into 3 scores as explained below (Santhosh, 2023) with the equations for each score is shown in Fig. 4:

1. ROUGE-1: Measures the precision, recall and F1-score based on the unigram overlap (overlap of single words). It's used to assess the fluency and grammatical correctness of the generated summary.
2. ROUGE-2: Measures the precision, recall and F1-score based on the bigram overlap (overlap of 2-word sequences). It's used to assess the fluency and grammatical correctness of the generated summary.
3. ROUGE-L: Measures the precision, recall and F1-score based on the longest common subsequence (LCS). It's used to assess the content coverage and semantic similarity of the generated summary.

$$\begin{aligned}
 RECALL &= \frac{\text{Overlapping number of } n\text{-grams}}{\text{Number of } n\text{-grams in the reference}} \\
 PRECISION &= \frac{\text{Overlapping number of } n\text{-grams}}{\text{Number of } n\text{-grams in the candidate}} \\
 F1 &= \frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})}
 \end{aligned}$$

Fig.4: Precision, recall and F1 score equations for each ROUGE score (Kızılırmak, 2023)

The higher the ROUGE scores, the better the LLM summarization performance. However, this metric has a few limitations as it only measures the n-gram overlaps, which may not be an accurate representation of a generated summary quality and may not be fully indicative of the semantic meaning/coherence quality of the generated summary (Kızılırmak, 2023). As such, other evaluation metrics such as the BERT Score are deployed to further assess the effectiveness of the LLM text summarization.

7.2. Bidirectional Encoder Representations from Transformers (BERT) Score

The BERTScore is another popular metric used to gauge the proficiency of a LLM in summarizing a given text. It aims to address the limitations of metrics that rely on n-gram overlap (such as the ROUGE score) as n-gram based metrics are not able to effectively capture long-range dependencies and incorrectly penalize semantically significant reordering (Sojasingarayar, 2024). Moreover, as the semantically accurate expression may be different from the reference model summary, n-gram based metrics may pair the wrong paraphrases between the generated and model summaries.

The BERTScore achieves its goals by the following procedure (Özolat, 2023):

1. The generated and model summaries are parsed into contextual embeddings using various embedding models such as BERT, Roberta.
2. The similarity between the generated and model summaries' contextual embeddings is measured using cosine similarity.
3. The recall, precision and F1 score are calculated by matching each token in the generated summary to the most similar token in the reference summary as shown in Fig.5. where \hat{x} is the generated response token and x is the model response token.

$$R_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j, \quad P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j, \quad F_{\text{BERT}} = 2 \frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}$$

Fig.5: Recall (R), Precision (P), F1 (F) scores equations of BERTScore (Özolat, 2023)

4. Inverse Document Frequency (IDF) is utilized to consider rare words into BERTScore equations. (Optional)
5. BERTScore values are then linearly rescaled for easier comparison. Fig.6 shows an example of the re-scaling equation for the Recall score of BERTScore, where \hat{R}_{BERT} is the rescaled Recall score and R_{BERT} is the un-rescaled Recall score.

$$\hat{R}_{\text{BERT}} = \frac{R_{\text{BERT}} - b}{1 - b}$$

Fig.6: Rescaled Recall (R) score equation of BERTScore (Özolat, 2023)

Similarly to the ROUGE Score, the higher the BERT score, the better the LLM summarization performance. Unfortunately, the BERTScore has a few limitations of its own which includes the biasness towards models which are like the BERT model and a sentence's syntactic structure is not accessed in the BERTScore calculations (Sojasingarayar, 2024).

7.3. Total Ranking Scores and Ranking Scores Variance (For Ranking List Generation LLM)

While ROUGE and BERT scores provide good quantitative measures of a LLM's summarization capability, they do not effectively measure a LLM's ability to provide the correct number ranking order of items based on the criteria set in the input prompts. As such,

the ranking score metric is used to access a LLM’s ability to provide an accurate and consistent ranking list of the goods and services based on the user’s query.

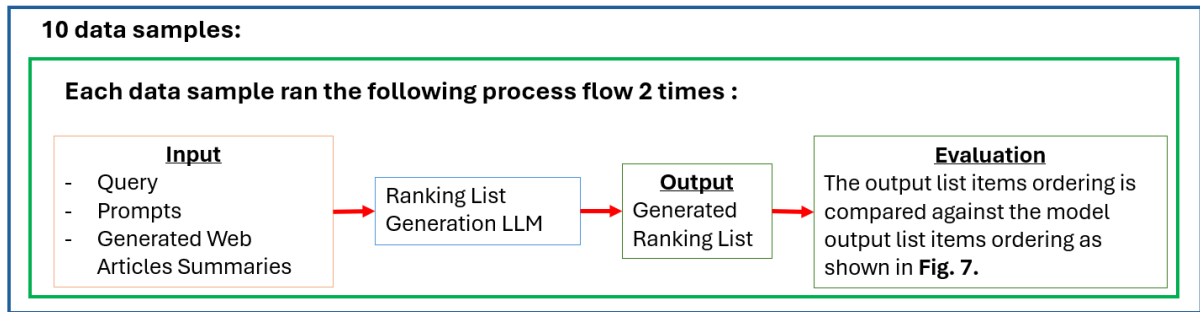


Fig.7: Ranking Scores Evaluation Process Flow

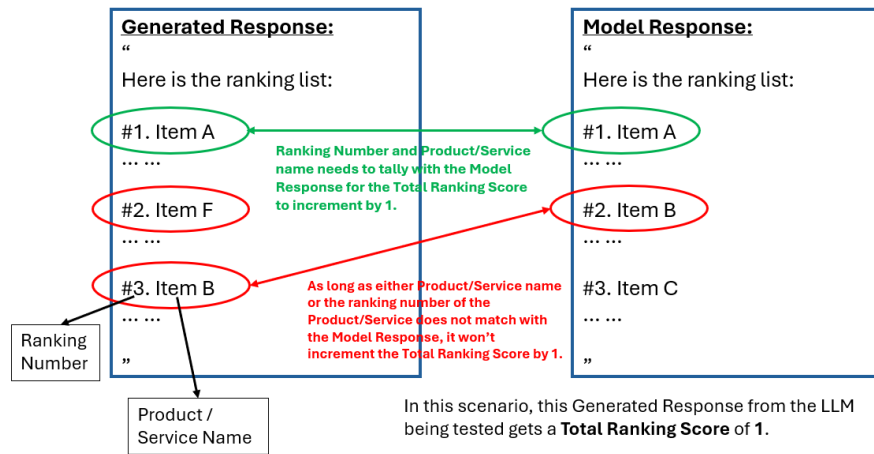


Fig.8: Total Ranking Scores Calculation

To obtain the ranking score, 10 manually created data samples (Containing sample query, prompts and generated summaries from web articles) are passed through the ranking list generation LLM to simulate the actual input the LLM would receive in a real-life setting. Each sample is ran a few times (2 for evaluation (Section 9: Ranking List LLM Evaluation)), 5 for optimization (Section 10: Model Optimization)) for each LLM test to capture the non-deterministic nature of LLMs and each generated output ranking list is compared against the model ranking list to check on the validity of the ranking as shown in Fig. 7. One point is given if the item name and ranking number in the generated output ranking list exactly matches the item name and ranking number in the model ranking list as shown in Fig. 8.

By using various evaluation metrics, a more holistic overview of a LLM’s performance can be achieved.

8. Summarizer LLM Evaluation (Results and Discussions)

The LLM Models for comparison (to achieve a fair comparison as much as possible, the Parameter Count version chosen for each model are kept as close to one another as possible (~7-8 billion)) :

1. Qwen2 (Qwen2, n.d.) (Ollama, n.d.)

- a. **Parameter Count:** 7 billion
- b. **Architecture:** Transformer architecture consisting of a mixture of sliding window/full attention, SwiGLU activation, group query attention, attention QKV bias etc. Context Length of 128000. The model pretrained on 7 trillion tokens and uses a Mixture of Expert (MoE) model. The model then goes through supervised fine-tuning, reinforcement learning from human feedback and rejection sampling (Qwen Team, n.d.).
- c. **Further Details:** Developed by Alibaba Group.

2. Llama 3.1 (meta-llama, n.d.)

- a. **Parameter Count:** 8 billion
- b. **Architecture:** Standard decoder-only transformer architecture with self-attention and feedforward network loops with SwiGLU activation function and group query attention. It uses minor adaptations instead of a Mixture of Expert (MoE) models to maximize training stability. It uses RoPE for positional embeddings and a 128000-vocabulary token size. Pretrained on 15 trillion tokens
- c. **Further Details:** Developed by Meta, its model was trained on high-quality data sets using the following strategies to achieve it: PII and safety filtering (to remove personally identifiable information and adult content, De-duplication: (to eliminate repeated data at various levels (URL, document, line)), Model-based quality filtering (using models like llama2, DistillRoberta, and fasttext to select high-quality tokens) . After the model is trained for text prediction using the aforementioned datasets, the model then undergo instruction fine-tuning, Rejection Sampling (RS) and Direct Preference Optimization (DPO) to generate more accurate, relevant and human-like responses. (Introducing Llama 3.1: Our most capable models to date, 2024)

3. Mistral (Deep-Dive-Into-AI-With-MLX-PyTorch, 2024)

- a. **Parameter Count:** 7 billion
- b. **Architecture:** Standard decoder-only transformer model. It uses Sliding Window Attention and Grouped-Query Attention (GQA) and has a Rolling buffer Cache. It also has 32 layers, 32 attention heads, and 8 key-value heads. The hidden size is 4096, and the intermediate size is 14336. The model's maximum position embedding is 32768, and the vocabulary size is 32000. Uses the Sigmoid Linear Unit (Silu) activation function. (Papers Explained 64: Mistral, 2023)
- c. **Further Details:** Developed by Mistral AI.

4. Deepseek-llm (DeepSeek-LLM, n.d.)

- a. **Parameter Count:** 7 billion
- b. **Architecture:** Auto-regressive decoder-only transformer model following the design of the Llama models. It has a mixture-of-experts layer, multi-head attention. It also consists of 30 layers, 32 heads and has a context length of 4096. During the model's pretraining, it uses Direct Performance Optimization instead of reinforcement learning from human feedback. Trained from 2 trillion tokens. (Chibueze, 2025) (DeepSeek LLM Scaling Open-Source Language Models with Longtermism, n.d.)
- c. **Further Details:** Developed by DeepSeek.

5. Phi3

- a. **Parameter Count:** 3 billion (phi3, n.d.)
- b. **Architecture and Training:** Dense decoder-only transformer model that follows closely to the llama 2 model. Vocabulary size of 32064. The model uses 3072 hidden dimension, 32 heads and 32 layers. It also uses grouped-query attention. Its pre training is done with datasets that included synthetic and filtered publicly available data totalling to 4.8 trillion tokens. Post training is done with supervised fine-tuning and direct preference optimization. It is a light weight model that can be used in mobile phone and can occupy only ~1.8GB of memory. (Ajay, 2024) (Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone, n.d.)

c. **Further Details:** Developed by Microsoft.

8.1. Results

Table 3. ROUGE Scores of the candidate Summarizer LLM (Max values highlighted in yellow)

LLM_Model	rouge-1-r	rouge-1-p	rouge-1-f	rouge-2-r	rouge-2-p	rouge-2-f	rouge-l-r	rouge-l-p	rouge-l-f
qwen2:latest	0.584	0.381	0.440	0.294	0.189	0.217	0.536	0.349	0.403
llama3.1:latest	0.468	0.545	0.488	0.253	0.311	0.269	0.430	0.501	0.448
mistral:latest	0.568	0.385	0.445	0.288	0.192	0.222	0.525	0.356	0.411
deepseek-llm:latest	0.398	0.450	0.402	0.182	0.213	0.185	0.361	0.408	0.364
phi3:latest	0.076	0.091	0.072	0.011	0.009	0.009	0.066	0.082	0.063

Table 4. BERT Scores of the candidate Summarizer LLM (Max values highlighted in yellow)

LLM_Model	Precision Mean	Precision Stddev	Recall Mean	Recall Stddev	F1 Score Mean	F1 Score Stddev	hashcode
qwen2:latest	0.890	0.0338	0.924	0.0253	0.907	0.0246	roberta-large_L17_no-idf_version=0.3.12
llama3.1:latest	0.926	0.0267	0.909	0.0288	0.917	0.0230	roberta-large_L17_no-idf_version=0.3.12
mistral:latest	0.891	0.0304	0.923	0.0226	0.907	0.0225	roberta-large_L17_no-idf_version=0.3.12
deepseek-llm:latest	0.907	0.0297	0.898	0.0295	0.902	0.0241	roberta-large_L17_no-idf_version=0.3.12
phi3:latest	0.777	0.0520	0.788	0.0510	0.782	0.0493	roberta-large_L17_no-idf_version=0.3.12

Table 5. Summarizer LLM Generated Summaries Word Count Statistics

LLM_Model	phi	deepseek	mistral	llama3.1	qwen2
count	3200	3200	3200	3200	3200
Mean Word Count	14	17	30	15	32
Word Count STDev	63	7	9	2	12
Min Word Count	0	3	10	7	0
25th Percentile	2	12	23	14	24
50th Percentile	7	15	28	15	30
75th Percentile	17	20	35	17	38
Max Word Count	3184	67	74	25	169

Observations:

For the Rouge Score Evaluation as shown in Table 3, the Llama 3.1 model have the highest F1 mean and precision mean scores for the Rouge-1, Rouge-2 and Rouge-L while the Qwen2 model have the highest Recall score. Overall, the Llama 3.1 model has the best Rouge Score performance compared to the other LLMs. For the Bert Score Evaluation as shown in Table 4, the Llama 3.1 model have the highest F1 Score mean and Precision mean scores and the lowest Precision standard deviation score. Mistral model has the lowest Recall and F1 Score standard deviation. Qwen 2 has the highest Recall Mean score. Overall, the Llama 3.1 model has the best Bert Score performance compared to the other LLMs. On top of that, Llama 3.1 model has the second-lowest mean word count despite achieving the best summarization performance as shown in Table 5.

8.2. Discussion and Insights

Llama 3.1's superior overall summarization performance over the other models is due to several reasons. The Llama 3.1 has an increased context length of 128 000 tokens compared to its previous iteration, ~1 billion more parameters compared to the other models, had better training stability using a standard decoder-only transformer model with minor adaptation instead of a mixture-of-experts model and on top of that was pretrained heavily by Meta AI using 15 trillion tokens from public sources which is further optimized using human-generated / synthetically generated examples. Compared to Mistral's 32000 context token length, Llama 3.1's 128000 context token length allows it to process a greater number of texts at one time than that of Mistral. Thus, Llama 3.1 can better understand the context of the input text to provide more accurate responses. Even though Qwen2 and DeepSeek-LLM have similar context length compared to Llama 3.1, they were not trained as heavily by their respective companies compared to that of Llama 3.1. Qwen 2 is pretrained on 7 trillion tokens while DeepSeek-LLM is pretrained on 2 trillion tokens however, Llama 3.1 is pretrained on 15 trillion tokens. Moreover, Llama 3.1 have ~5 billion more parameters than the Phi3 model, allowing it to capture more aspects of the inputs/prompts. The latest version of Llama features enhancements in both the quantity and quality of data for pre- and post-training, achieved through improved pre-processing, curation pipelines, stricter quality assurance, and advanced filtering methods. Llama 3.1's superior training process allows it to be able to better understand the system/user and assistant prompts as well as handle input texts of greater lengths to generate higher quality responses.

9. Ranking List LLM Evaluation

Due to the numerous prompting techniques and various input prompt formats, the Ranking List LLM Evaluation consists of 2 parts to explore these variations with the methodology sections (Section 9.1.1 and Section 9.2.1) for each evaluation part explains the purpose and evaluation process flow for their respective evaluation part.

The main criteria to determine a product/service ranking will be its appearance frequency throughout the input text as it is reasonable to assume that the product/service name that appears many times across the top few most relevant web articles fetched by Google Search API would be the most relevant to the query. For example, if the query is "What is the best

Sunscreen brand?” and “Neutrogena” is mentioned the greatest number of times across the top 3 most relevant web articles’ text from the Google Search API, then it would be reasonable to assume that “Neutrogena” is the best sunscreen brand. As such, the expected logic for the Ranking List LLM to determine the product/services rankings is shown in Fig. 9. The Ranking List LLM will have to first identify the relevant product/service to the query, then count the number of times the product/service appears in the input text and subsequently, the order the identified product/service in ascending order.

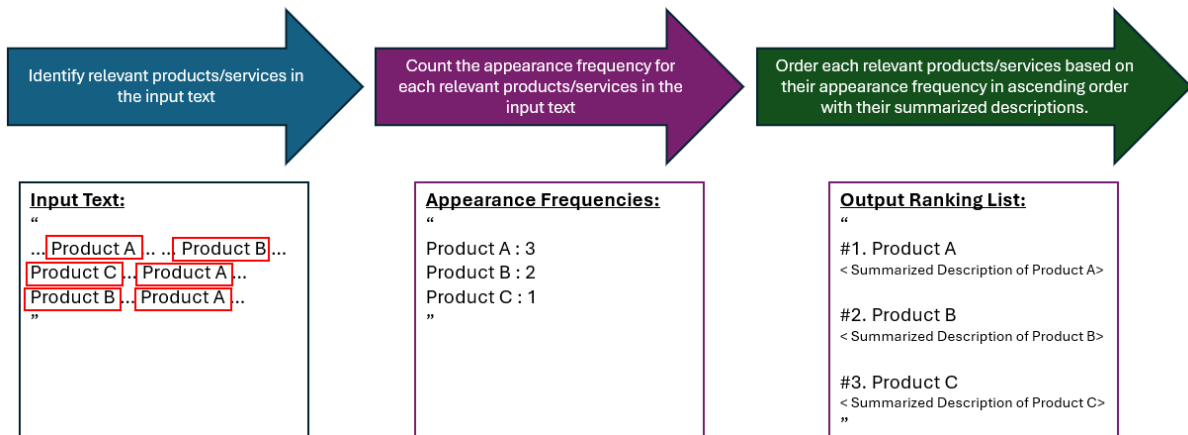


Fig. 9: Ranking List LLM's Process Logic in determining product/services ranking

****Note:** Few shot prompting techniques mentioned in the project report is referred as Few Shot (X) prompting, where X is the number of prompt examples provided in the prompt.

9.1. Evaluation Part 1

9.1.1. Methodology:

The Evaluation Part 1 study investigates the effect of the different text input types (single concatenated text (SCT) input vs a separated and labelled text (SLT) has on the performance of the ranking list response generation as illustrated in Fig.10 below. It also analyses the effect of the different prompt engineering techniques (Zero Shot vs Few (3) Shot) as illustrated in Fig. 11 (which shows Zero Shot vs One Shot Prompt as an example) below as well as the effect of placing the prompt examples in the System role vs User role.

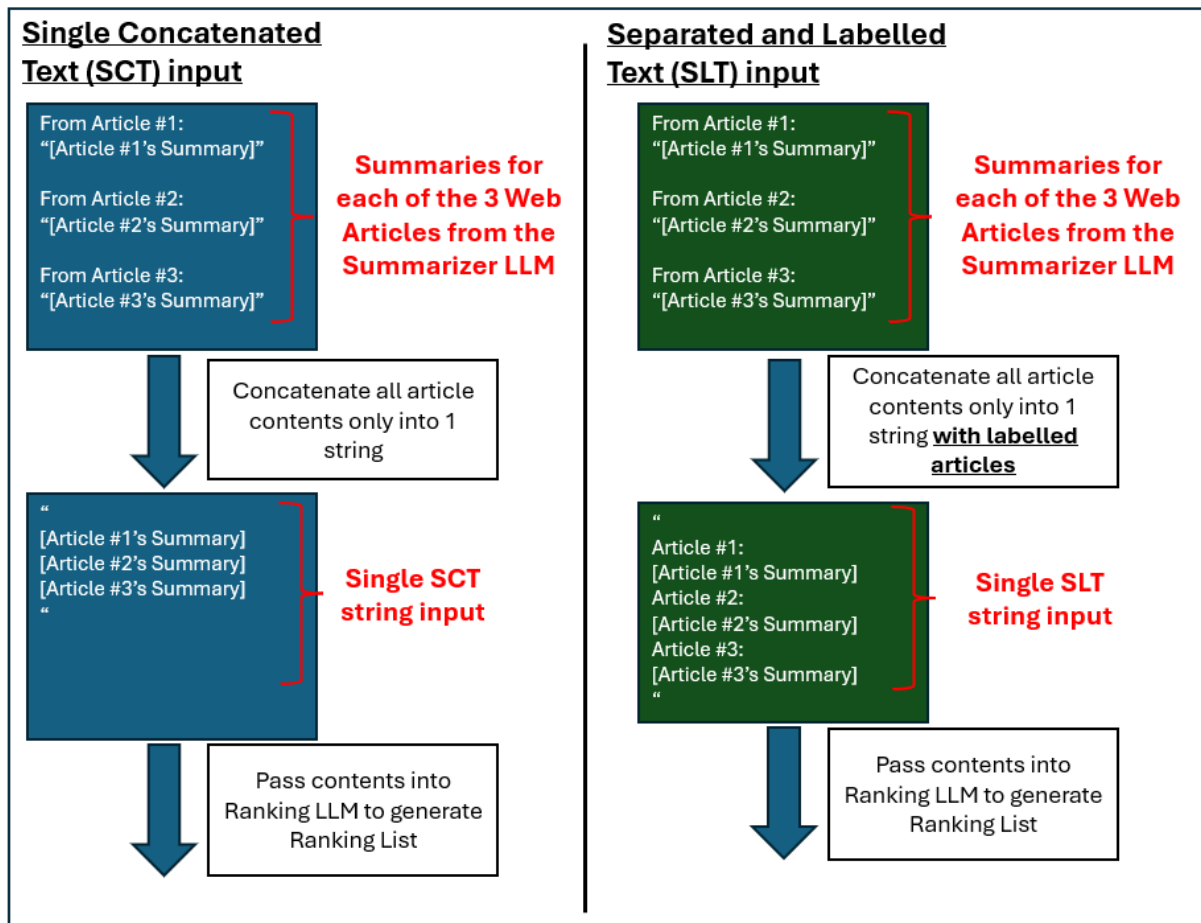


Fig. 10: Single Concatenated Text (SCT) input vs a Separated and Labelled Text (SLT) input

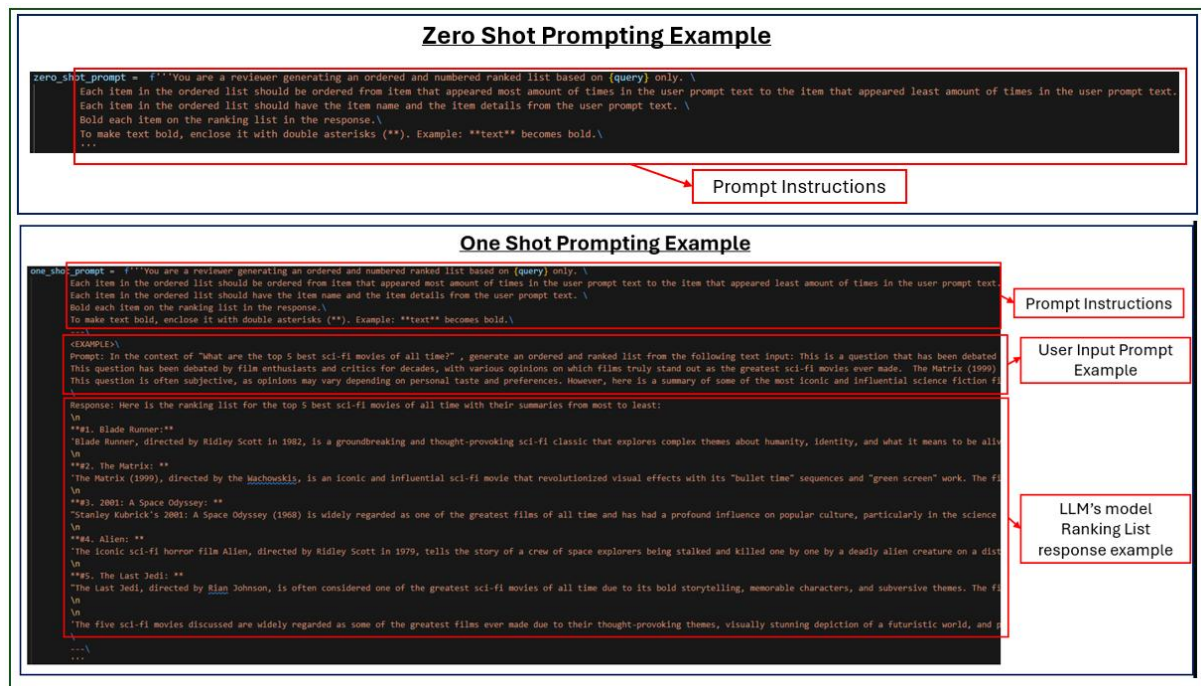


Fig. 11: Zero Shot vs One Shot Prompting example

To quantitatively measure the performance of the ranking list from each effect, each LLM/prompt variant being tested generates 2 ranking list responses for test query and each element in the generated ranking list is compared against the model ranking list whereby each model ranking list's element reflected in the generated ranking list (the product/service name and ranking number in the generated response needs to match with that of the model response) would increment the LLM/prompt variant Ranking Score by 1. Moreover, all the generated responses are parsed through the evaluation metric functions to obtain the ROUGE and BERT scores (which measures the summarization quality of the response). This performance measurement processes were explained in Section 7: Evaluation Metrics.

9.1.2. Results

Table 6. ROUGE Scores of the various Prompt Engineering techniques (Zero Shot vs Few(3) Shot) (Max values highlighted in yellow)

Prompt Engineering Technique	rouge-1-r	rouge-1-p	rouge-1-f	rouge-2-r	rouge-2-p	rouge-2-f	rouge-l-r	rouge-l-p	rouge-l-f
Few(3) Shot SCT Response	0.267	0.561	0.358	0.108	0.289	0.156	0.251	0.528	0.337
Few(3) Shot SLT Response	0.258	0.530	0.345	0.106	0.265	0.150	0.245	0.505	0.328
Zero Shot SCT Response	0.340	0.570	0.420	0.164	0.315	0.213	0.321	0.539	0.397
Zero Shot SLT Response	0.334	0.564	0.413	0.160	0.312	0.208	0.309	0.524	0.382

Table 7. BERT Scores of the various Prompt Engineering techniques (Zero Shot vs Few(3) Shot) (Max values highlighted in yellow)

Prompt Engineering Technique	Precision Mean	Precision Stddev	Recall Mean	Recall Stddev	F1 Score Mean	F1 Score Stddev	hashcode
Few(3) Shot SCT Response	0.880	0.0183	0.852	0.0073	0.866	0.0111	roberta-large_L17_no-idf_version=0.3.12
Few(3) Shot SLT Response	0.873	0.0223	0.850	0.0121	0.861	0.0164	roberta-large_L17_no-idf_version=0.3.12
Zero Shot SCT Response	0.883	0.0230	0.867	0.0202	0.875	0.0213	roberta-large_L17_no-idf_version=0.3.12
Zero Shot SLT Response	0.883	0.0200	0.864	0.0190	0.873	0.0186	roberta-large_L17_no-idf_version=0.3.12

Table 8. ROUGE Scores of the various Prompt Engineering techniques (Few(3) Shot Prompts in USER role) (Max values highlighted in yellow)

Prompt Engineering Technique	rouge-1-r	rouge-1-p	rouge-1-f	rouge-2-r	rouge-2-p	rouge-2-f	rouge-l-r	rouge-l-p	rouge-l-f
Few(3) Shot USER SCT Response	0.271	0.549	0.360	0.117	0.294	0.166	0.256	0.518	0.340
Few(3) Shot USER SLT Response	0.256	0.506	0.334	0.104	0.243	0.142	0.239	0.473	0.312

Table 9. BERT Scores of the various Prompt Engineering techniques (Few(3) Shot Prompts in USER role) (Max values highlighted in yellow)

Prompt Engineering Technique	Precision Mean	Precision Stddev	Recall Mean	Recall Stddev	F1 Score Mean	F1 Score Stddev	hashcode
Few(3) Shot USER SCT Response	0.874	0.0223	0.854	0.0109	0.864	0.0156	roberta-large_L17_no-idf_version=0.3.12
Few(3) Shot USER SLT Response	0.862	0.0238	0.846	0.0188	0.854	0.0205	roberta-large_L17_no-idf_version=0.3.12

Table 10. Total Ranking Scores of the various Prompt Engineering techniques (Max values highlighted in yellow)

Prompt Engineering Technique	Ranking Scores
Few(3) Shot SCT Response	40
Few(3) Shot SLT Response	37
Zero Shot SCT Response	38
Zero Shot SLT Response	48
Few(3) Shot USER SCT Response	35
Few(3) Shot USER SLT Response	35

Observations: Comparing the results of the responses that comes from a single concatenated text (SCT) input vs a separated and labelled text (SLT) input, the SCT input responses achieve higher Total Ranking Scores than SLT input responses for Few (3) Shot prompting but achieve lower Total Ranking Scores than SLT input responses for Zero Shot prompting as shown in Table 10. Comparing in terms of the ROUGE and Bert Scores, the SCT obtained higher overall ROUGE F1, precision, recall scores than that of the SLT (as shown in Tables 6, 7, 8, 9) and obtained higher overall BERT F1, precision, recall scores and lower F1 Score Standard Deviation than that of the SLT (as shown in Tables 6, 7, 8, 9). All in all, even though SCT have better quality summaries than that of SLT, it has mixed ranking accuracies results and its performance against SLT is inconclusive. Thus, the evaluation Part 2 in Section 9.2 will delve further into the comparison between SCT and SLT.

Comparing the results of the responses between putting the prompt examples in the System role vs putting the prompt examples in the User role in (Tables 6,7,8,9,10), the former attains higher overall Total Ranking Scores than the latter. Moreover, putting the prompt examples in the System role produces responses which have higher overall ROUGE F1, precision, recall scores than that of putting the prompt examples in the User role and have higher overall BERT F1, precision, recall scores than that of putting the prompt examples in the User role. Hence, putting the prompt examples in the System role achieve more accurate rankings and higher quality summaries than that of putting prompt examples in the User role. Thus, moving forward, the prompt examples would be placed in the System role for One/Few Shot prompting techniques.

Comparing the results of the responses that come from Zero Shot and Few (3) Shot prompting techniques in (Tables 6,7,8,9,10), the Zero Shot prompting technique achieved higher overall Total Ranking Scores compared to that of the Few (3) Shot prompting. On top of that, the Zero

Shot prompting obtained higher overall ROUGE F1, precision, recall scores than that of the Few (3) Shot prompting while achieving higher overall BERT F1, precision, recall scores and lower F1 Score Standard Deviation than that of the SLT. As this evaluation only compares between Zero Shot and Few (3) Shot prompting, the other variants of Few Shot promptings will be compared against the Zero Shot Prompting in Section 9.2: Evaluation Part 2.

9.1.3. Discussion and Insights

As the SCT's instruction to "find the appearance frequency of a name in the input text" is considerably simpler to understand compared to the SLT's instruction to "find the appearance frequency of a name across all ARTICLES", the LLM would find it easier to follow the former instruction. Thus, the LLM can produce ranking lists of higher accuracy with the SCT input as opposed to the SLT input. Moreover, the SCT input produces ranking lists with higher summarization quality compared to that of SLT. Hence, it is generally recommended to be straight to the point and to not include unnecessary descriptions / tasks for the LLM to process as shown in Fig. 12 from the OpenAI guide to writing prompts (Best practices for prompt engineering with the OpenAI API, 2024).

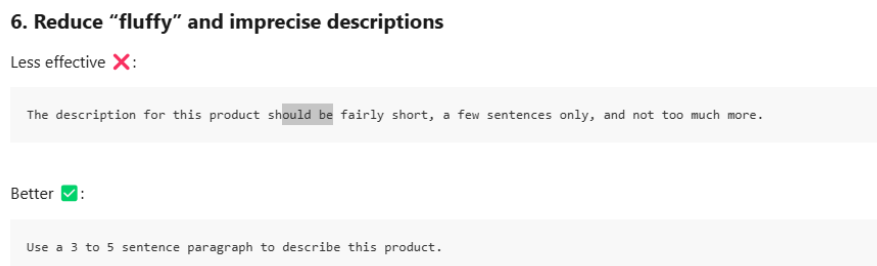


Fig. 12: Rule 6 of OpenAI's guide to writing effective prompts (Best practices for prompt engineering with the OpenAI API, 2024)

Inputs in the System role guide the LLM behaviour/role in the conversation with the user by specifying specific guidelines on how it should interact with the user while inputs in the User role are the inputs by the user to the chat for the LLM to generate a response to (Behl, 2024). Hence, putting the prompt examples in the system role provided an effective high-level context for the LLM to generate a more accurate response compared to that of putting the example prompts in the User role.

Interestingly, Zero Shot prompting achieves a better performance than Few Shot prompting when it is expected that providing more examples to the System Role will result in higher

accuracy in the products/services rankings. However, these results can be explained by the fact that the Few Shot prompting technique is usually used for nuanced, specific and complex tasks while the Zero Shot prompting technique is usually used for general queries (Stihec, 2024). As the project aims to deliver a Ranking List LLM that would be able to rank any product/service from any category based on the text frequency, the query scope is so broad that providing a few prompt examples and answers into the System Role is insufficient for the LLM to pick up on the required logic to produce accurate product/service rankings. Furthermore, adding additional prompt examples could potentially mislead the LLM to utilize the wrong logic in determining the products/services rankings. Thus, for the huge general scope that the project application requires, a Zero Shot prompting technique would result in better performance in generating a ranking list response. Nonetheless the 2nd part of the evaluation study will explore other prompting techniques to compare against Zero Shot.

9.2. Evaluation Part 2

9.2.1. Methodology:

Due to the inconclusive nature of the SCT vs SLT Evaluation results from the Part 1 evaluation, a second study is conducted again albeit a few changes to the prompt instructions to specifically instruct the LLM to produce an ordered ranking list following OpenAI's prompt creation guidelines (Best practices for prompt engineering with the OpenAI API, 2024) as shown in Fig. 13.

<p>Previous Prompt</p> <p>Prompt = f"" You are a reviewer generating an ordered list based on {query} only. \</p> <p>Each item in the ordered list should be ordered from item that appeared most amount of times in the user's text input to the item that appeared least amount of times in the user's text input. \</p> <p>Each item in the ordered list should have the item name and the item details from the user's text input. \</p> <p>Bold each item on the ranking list in the output. \</p> <p>To make text bold, enclose it with double asterisks (**). Example: **text** becomes bold. \</p> <p>""</p>
<p>New Prompt</p> <p>Prompt = f"" You are a reviewer generating an ordered and numbered ranked list based on {query} only. \</p> <p>Each item in the ordered list should be ordered from item that appeared most amount of times in the user prompt text to the item that appeared least amount of times in the user prompt text. \</p> <p>Each item in the ordered list should have the item name and the item details from the user prompt text. \</p> <p>Bold each item on the ranking list in the response. \</p> <p>To make text bold, enclose it with double asterisks (**). Example: **text** becomes bold. \</p> <p>""</p>

Fig. 13: Previous Prompt Instructions vs New Prompt Instructions (Changes highlighted in Yellow)

As mentioned in the Discussion and Insights section for the Evaluation Part 1 study, the Few Shot prompting techniques may prove to be ineffective for this project due to the project's broad general query scope. Thus, another advanced prompt engineering technique is explored in this Evaluation Part 2 study, namely the Chain-Of-Thought (COT) prompt engineering technique. COT is a prompting technique that allows for more complicated reasoning capabilities by utilizing intermediate reasoning steps (Chain-of-Thought Prompting, n.d.). Firstly, the evaluation will explore the performance of the various versions of COT as shown below in Fig.14 to discover the best COT variant in the context of this project. Secondly, the evaluation will explore whether placing the process logic in the System/User/Assistant roles would affect the performance of the ranking list generation.

As the Part 1 evaluation only compares the performance difference between the Zero Shot and Few (3) shot prompting techniques, the Part 2 evaluation aims to compare the response performance of the Zero-Shot Prompting technique against other prompting techniques (One-Shot Prompting, Few (5) Shot Prompting and Chain-Of-Thought (COT) Prompting). Following Section 9.1: Evaluation Part 1, the same Total Ranking Score measurement and ROUGE/BERT scores will be used to determine the ranking list accuracy and summarization quality respectively.

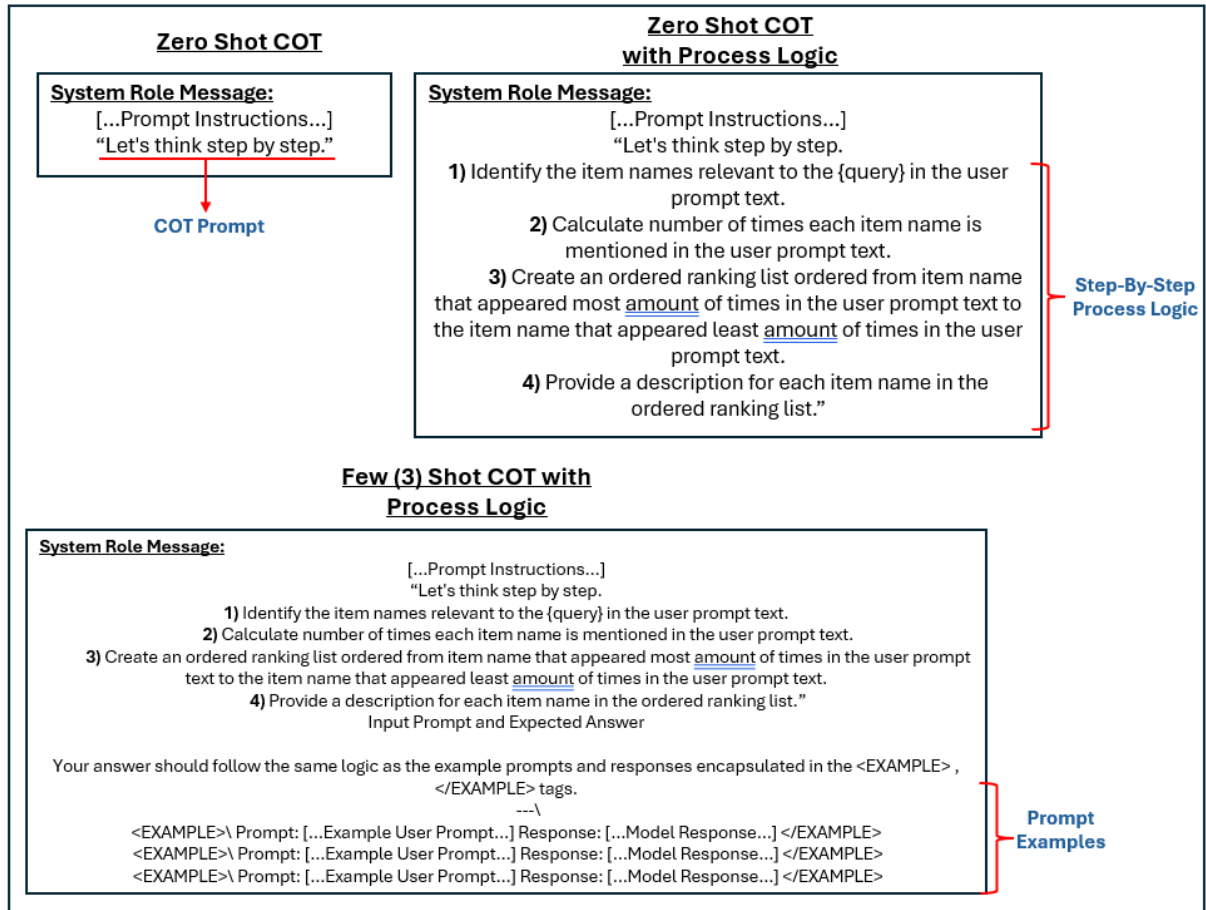


Fig.14: The different COT variants being tested in this project (Chain-of-Thought Prompting, n.d.)

9.2.2. Results

Table 11. ROUGE Scores of Zero Shot / One Shot / Few(5) Shot Prompt Engineering techniques (Max values highlighted in yellow)

Prompt Engineering Technique	rouge-1-r	rouge-1-p	rouge-1-f	rouge-2-r	rouge-2-p	rouge-2-f	rouge-l-r	rouge-l-p	rouge-l-f
Zero Shot Multi Response	0.304	0.573	0.393	0.141	0.312	0.192	0.286	0.539	0.370
Zero Shot Single Response	0.321	0.569	0.406	0.150	0.320	0.201	0.305	0.540	0.386
One Shot Multi Response	0.267	0.511	0.346	0.111	0.254	0.152	0.251	0.480	0.325
One Shot Single Response	0.290	0.570	0.381	0.126	0.303	0.175	0.272	0.535	0.357
Few(5) Shot Multi Response	0.262	0.508	0.340	0.100	0.237	0.138	0.242	0.472	0.315
Few(5) Shot Single Response	0.269	0.535	0.355	0.108	0.267	0.152	0.254	0.507	0.336

Table 12. BERT Scores of Zero Shot / One Shot / Few(5) Shot Prompt Engineering techniques (Max values highlighted in yellow)

Prompt Engineering Technique	Precision Mean	Precision Stddev	Recall Mean	Recall Stddev	F1 Score Mean	F1 Score Stddev	hashcode
Zero Shot Multi Response	0.881	0.0224	0.860	0.0158	0.870	0.0178	roberta-large_L17_no-idf_version=0.3.12
Zero Shot Single Response	0.883	0.0219	0.864	0.0161	0.874	0.0182	roberta-large_L17_no-idf_version=0.3.12
One Shot Multi Response	0.866	0.0242	0.855	0.0181	0.860	0.0206	roberta-large_L17_no-idf_version=0.3.12
One Shot Single Response	0.885	0.0224	0.860	0.0137	0.872	0.0169	roberta-large_L17_no-idf_version=0.3.12
Few(5) Shot Multi Response	0.865	0.0297	0.850	0.0198	0.857	0.0242	roberta-large_L17_no-idf_version=0.3.12
Few(5) Shot Single Response	0.878	0.0219	0.854	0.0167	0.866	0.0185	roberta-large_L17_no-idf_version=0.3.12

Table 13. ROUGE Scores of Zero Shot COT with Process Logic in System/User/Assistant roles (Max values highlighted in yellow)

Prompt Engineering Technique	rouge-1-r	rouge-1-p	rouge-1-f	rouge-2-r	rouge-2-p	rouge-2-f	rouge-l-r	rouge-l-p	rouge-l-f
Zero Shot COT with Process Logic ASST Single Response	0.284	0.509	0.355	0.127	0.253	0.165	0.272	0.485	0.339
Zero Shot COT with Process Logic SYS Single Response	0.312	0.537	0.387	0.143	0.286	0.186	0.294	0.506	0.365
Zero Shot COT with Process Logic USER Single Response	0.304	0.522	0.380	0.136	0.271	0.178	0.286	0.493	0.357

Table 14. BERT Scores of Zero Shot COT with Process Logic in System/User/Assistant roles (Max values highlighted in yellow)

Prompt Engineering Technique	Precision Mean	Precision Stddev	Recall Mean	Recall Stddev	F1 Score Mean	F1 Score Stddev	hashcode
Zero Shot COT with Process Logic ASST Single Response	0.857	0.0455	0.841	0.0423	0.849	0.0420	roberta-large_L17_no-idf_version=0.3.12
Zero Shot COT with Process Logic SYS Single Response	0.871	0.0255	0.858	0.0224	0.864	0.0227	roberta-large_L17_no-idf_version=0.3.12
Zero Shot COT with Process Logic USER Single Response	0.862	0.0274	0.855	0.0208	0.859	0.0233	roberta-large_L17_no-idf_version=0.3.12

Table 15. ROUGE Scores of Zero Shot COT vs Few (3) Shot COT with Process Logic (Max values highlighted in yellow)

Prompt Engineering Technique	rouge-1-r	rouge-1-p	rouge-1-f	rouge-2-r	rouge-2-p	rouge-2-f	rouge-l-r	rouge-l-p	rouge-l-f
Zero Shot COT SYS Single Response	0.334	0.593	0.421	0.160	0.338	0.214	0.314	0.559	0.397
Few (3) Shot COT with Process Logic SYS Single Response	0.269	0.544	0.354	0.111	0.268	0.155	0.251	0.508	0.331

Table 16. BERT Scores of Zero Shot COT vs Few (3) Shot COT with Process Logic (Max values highlighted in yellow)

Prompt Engineering Technique	Precision Mean	Precision Stddev	Recall Mean	Recall Stddev	F1 Score Mean	F1 Score Stddev	hashcode
Zero Shot COT SYS Single Response	0.890	0.0222	0.868	0.0186	0.879	0.0191	roberta-large_L17_no-idf_version=0.3.12
Few (3) Shot COT with Process Logic SYS Single Response	0.881	0.0212	0.853	0.0183	0.867	0.0184	roberta-large_L17_no-idf_version=0.3.12

Table 17. Total Ranking Scores of the various Prompt Engineering techniques (Max values highlighted in yellow)

Total Ranking Scores	
Prompt Engineering Technique	Ranking Scores
Zero Shot SLT Response	47
Zero Shot SCT Response	48
One Shot SLT Response	31
One Shot SCT Response	35
Few(5) Shot SLT Response	35
Few(5) Shot SCT Response	35
Zero Shot COT with Process Logic USER SCT Response	41
Zero Shot COT with Process Logic ASST SCT Response	38
Zero Shot COT with Process Logic SYS SCT Response	49
Zero Shot COT SYS SCT Response	38
Few (3) Shot COT with Process Logic SYS SCT Response	39

Observations:

Comparing the results of the responses that comes from a single concatenated text (SCT) input vs a separated and labelled text (SLT) input across the various prompting techniques as shown in (as shown in Tables 11,12,13,14,15,16,17), the SCT input responses achieve higher overall ranking scores than SLT input responses. Comparing in terms of the ROUGE and Bert Scores, the SCT obtained higher overall ROUGE F1, precision, recall scores than that of the SLT and obtained higher overall BERT F1, precision, recall scores and lower F1 Score Standard Deviation than that of the SLT. Hence, from this Part 2 evaluation, the SCT input results in an overall more accurate ranking and better-quality summaries response compared to that of the SLT input.

Comparing the results of the responses among the multiple COT variants in (as shown in Tables 13,14,15,16,17), it is observed that putting the COT process logic into the System role yields the highest Total Ranking Scores as well as the best ROUGE F1, precision, recall score and best BERT Score with the highest F1, Precision, Recall scores means and lowest F1 Score Standard Deviation. Thus, placing the process logic into the System Role is the recommended method in utilizing COT. On top of that, comparing the results of the responses among the multiple COT variants, it is observed that implementing Zero Shot COT with the process logic and without the prompt examples yields better Total Ranking Scores compared to the other COT variants, i.e. Zero Shot COT with no process logic/ prompt examples, Few Shot COT with process logic. Zero Shot COT with process logic achieves lower overall ROUGE scores compared to Zero Shot COT with no process logic and higher overall ROUGE scores compared to Few Shot COT with process logic. On top of this, the Zero Shot COT with process logic attains the lowest overall BERT Scores compared to the other COT variants. These results meant that the Zero Shot COT with process logic in System Role achieves the highest accuracy in product/service ranking compared to the other COT variants, but the lowest summarization quality compared to the other COT variants.

Comparing the various prompting techniques among Zero Shot, One Shot, Few (5) Shot and COT (Zero Shot COT with process logic) as shown in (Tables 11,12,13,14,17), Zero Shot prompting achieves the best overall ROUGE scores (F1, precision, recall) as well as the best overall BERT scores compared to the other specified prompting techniques and Zero Shot COT with process logic coming a close second. This means that the Zero Shot prompting have the best product/service summary quality compared to the other prompting techniques. However,

Zero Shot COT with process logic achieved the highest-Ranking Score (49) compared to the other specified prompting techniques with Zero Shot prompting coming a close second (48). Thus, Zero Shot COT with process logic produces the highest product/service ranking accuracy.

9.2.3. Discussion and Insights

Comparing the results between prompting techniques that used SCT text input vs SLT input, the prompting techniques that utilize SCT achieves the overall best performance as shown in as shown in Tables 11,12,13,14,15,16,17 and explained in the previous Section. The SCT's prompt instruction "find the appearance frequency of a name in the input text" is considerably simpler to understand compared to the SLT's instruction to "find the appearance frequency of a name across all ARTICLES", hence the LLM would find it easier to follow the former instruction. Thus, the SCT input is chosen to be input type for this project.

Comparing among the Zero Shot, One Shot, Few (5) Shot prompting techniques, the Zero Shot prompting technique has the best overall performance as shown in as shown in Tables 11,12 ,17 and explained in the previous section. It is clear that as the One Shot , Few (3) Shot , Few (5) Shot prompting techniques in Evaluation Part 1 and Part 2 studies all have worse performance compared to Zero Shot, adding prompt examples in this project context only serves to confuse the LLM as the prompt examples are insufficient to cover the project's broad scope of queries that could be asked due to the numerous product/service categories in the world or the prompt examples cause the LLM to behave too specifically that makes it underperform in a general context.

Comparing among the COT prompting variants, Zero Shot COT Process Logic in System Role (Zero Shot COT PLSR) achieves the best overall performance. This is aligned with the findings from Evaluation Part 1, whereby inputting instructions that dictate the system behaviour would be most effective when said instructions are in the system Role as opposed to the User/Assistant Roles as explained in Section 9.1.3 and as shown in Tables 13,14,15,16,17. Moreover, Zero Shot COT PLSR did better than Zero Shot COT without Process Logic (COT NoPL) and Few (3) Shot COT with Process Logic (Few(3) Shot COT PLSR). COT noPL lack of Process Logic along with the prompt instruction "Let's think step by step" only serves to confuse the LLM on how to think "step-by-step". On the other hand, Few(3) Shot COT PLSR's prompt examples

only serve to confuse the LLM as it's prompt examples could have mislead the LLM logic. Thus, Zero Shot COT PLSR achieves the best overall performance among the COT variants.

Comparing Zero Shot prompting and Zero Shot COT PLSR prompting, while the Zero Shot's Total Ranking Score (48) is slightly lower than that of Zero Shot COT PLSR's total Ranking Score (49) indicating that the Zero Shot is very slightly lower in accuracy compared to that of the COT, the Zero Shot's ROUGE and BERT Scores are greater than that of the Zero Shot COT PLSR. Unlike the Zero Shot, the Zero Shot COT PLSR's tendency to output the Process Logic in the Ranking List response as shown in Fig.15 causes the Zero Shot COT PLSR summarization quality to decrease as showcasing the process logic in the output is unnecessary. As the project aims to showcase the Ranking List to the public, it would only cause frustration and confusion for the user to look through the output to find the Ranking List.

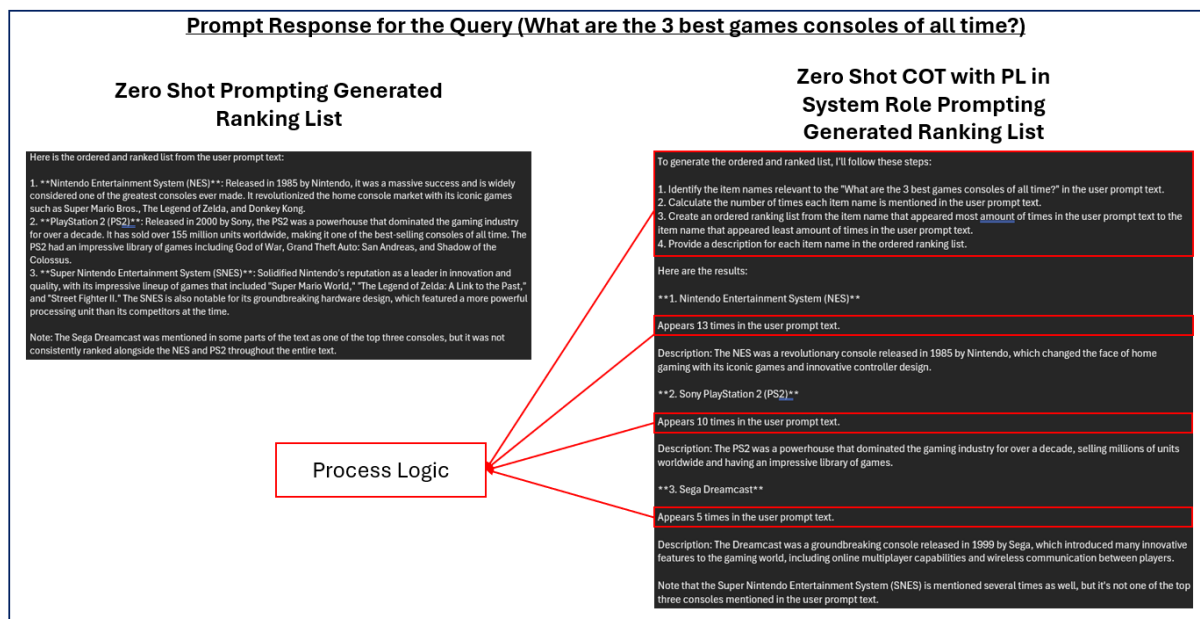


Fig.15: Zero Shot's Ranking List Response vs Zero Shot COT with Process Logic's Ranking List Response

All in all, as explained in the previous parts, the Ranking Model Prompting Technique Choice will be the **Zero Shot Prompting Technique** for this project's context.

10. Model Optimization

With the Ranking List LLM Prompting Technique chosen (Zero Shot prompting) as explained in the Section 9: Ranking List LLM Evaluation, this section focuses on the optimization of the Ranking List LLM with Zero Shot prompting based on the following LLM parameters: Temperature, Top P, Top K as explained below.

1. **Temperature:** This parameter controls the randomness and diversity of the LLM output. Lower temperature values (closer to 0) would cause the LLM to generate more precise and consistent responses while higher temperatures (greater than 1) would cause the LLM to generate more creative and variety in the responses. In general, lower temperature would be utilized when the user requires factual accuracy in the response while higher temperature would be utilized when the user requires variety such as brainstorming tasks.
2. **Top K:** The parameter controls the number of options (K) the LLM would consider when sampling. For example, if $K = 10$ is set, the LLM would only consider the top 10 word tokens when selecting the next word token in the response. This parameter helps to remove “low-quality” or less probable tokens from the list of possible tokens.
3. **Top P:** This parameter controls the probability threshold (P) on whether a word token is even considered in the possible token list. Lower P values would increase the amount of word tokens the LLM would consider as more “low-quality” or less probable tokens would be considered and vice versa for higher P values.

Generally, Temperature and Top P are not optimized together as the LLM would not be able to handle edge case scenarios if both Temperature and Top P parameters are too high. The Llama3.1’s default parameters are as follows (Temperature: 0.8, Top K: 40, Top P: 0.9) (Ollama modelfile.md, 2025).

10.1. Methodology:

The optimization process differs from the evaluation process by having a greater emphasis on the Total Ranking Scores of the model as well as the consistency of the Ranking Score as opposed to the summarization quality of the products/services described in the generated Ranking List response. Thus, for each test query, on top of calculating the Ranking Score for each generated response, each response is generated 5 times for each test set of LLM

parameters to have a more accurate representation of the ranking accuracy and variance for the model being tested. On top of that, the ranking score variance between each of the 5 generated response for the same query for each model being tested will be calculated to measure the consistency of the response generation of the model.

Sensitivity analysis is first utilized in the optimization process to investigate how each of the identified LLM parameters affect the accuracy and consistency of the generated response. Each of the identified LLM parameter is increased and decreased by 20% from the default value and the generated response performance is compared against to that of the default parameter values. The parameter change that yields better ranking performance will be considered in the Stepwise Regression with the parameter change that obtains the biggest ranking performance improvement the first and the parameter change that attains the smallest ranking performance improvement the last. Afterwards, Stepwise Regression is then used to slowly fine tune the LLM to the optimal point by increasing/decreasing step-by-step the appropriate parameter by 10% based on the results of the Sensitivity Analysis.

10.2. Sensitivity Analysis

10.2.1. Results

Table 18. Sensitivity Analysis Results

Sensitivity Analysis					
Prompt Engineering Technique	Total Ranking Response Score	5 Ranking Response Variance	Total Ranking Response Score % Change	Total Ranking Response Variance % Change	Change
Zero Shot Default (0.8 Temp , 0.9 Top P , 40 Top K)	111	1.19	0.00	0	-
Zero Shot (0.96 Temp)	109	0.62	-1.80	-47.90	INC Temp
Zero Shot (0.64 Temp)	105	0.51	-5.41	-57.14	DEC Temp
Zero Shot (0.72 Top P)	120	0.76	8.11	-36.13	DEC Top P
Zero Shot (1.08 Top P)	104	0.94	-6.31	-21.01	INC Top P
Zero Shot (32 Top K)	127	0.93	14.41	-21.85	DEC Top K
Zero Shot (48 Top K)	112	1.03	0.90	-13.45	INC Top K

From Table 18, it is observed that decreasing the Top P by 20% and decreasing the Top K by 20% improves the ranking response accuracy by 8.1% and 14.4% respectively and decreases the ranking response variance by 36.13% and 21.85% respectively. Hence, the Stepwise Regression will follow the following optimization process as shown in Fig.16 until the generated response performance deteriorates.

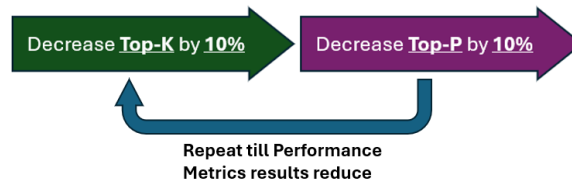


Fig. 16. Stepwise Regression Process Flow

10.2.2. Discussion and Insights

Interestingly, increasing/decreasing the Temperature parameter did not improve the LLM's performance. This could be an indication that the LLM's default parameter values are close to the optimal values.

10.3. Stepwise Regression

Following Fig.16, the Top K and Top P parameters decreased by 10% one at a time until the Ranking List LLM's performance drops.

The **Average Total Difference** and **Average Total Difference Variance** metrics introduced in this Section only serves to investigate the numerical difference between Ranking List LLM's **calculated** appearance frequency for each Ranking List element in the Generated Ranking List response vs the **actual** appearance frequency for each Ranking List element in the Ranking List LLM input i.e. the Generated Summaries from the Summarizer LLM.

10.3.1. Results

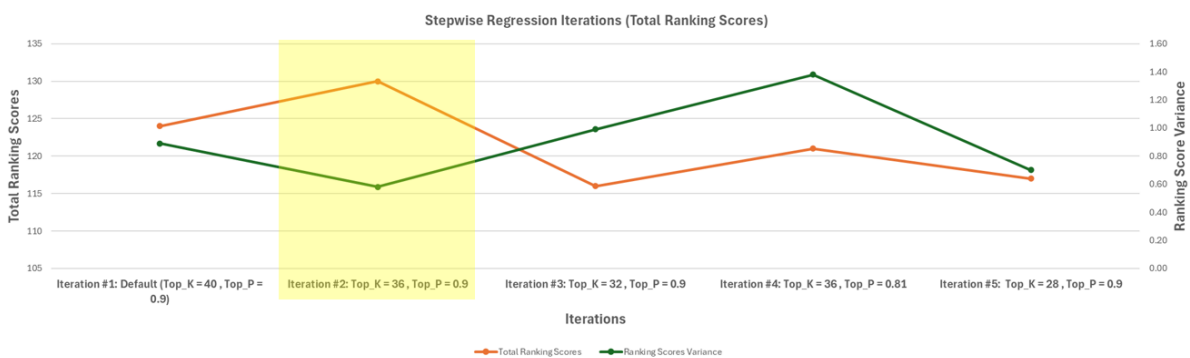


Fig. 17. Stepwise Regression Total Ranking Scores Trend

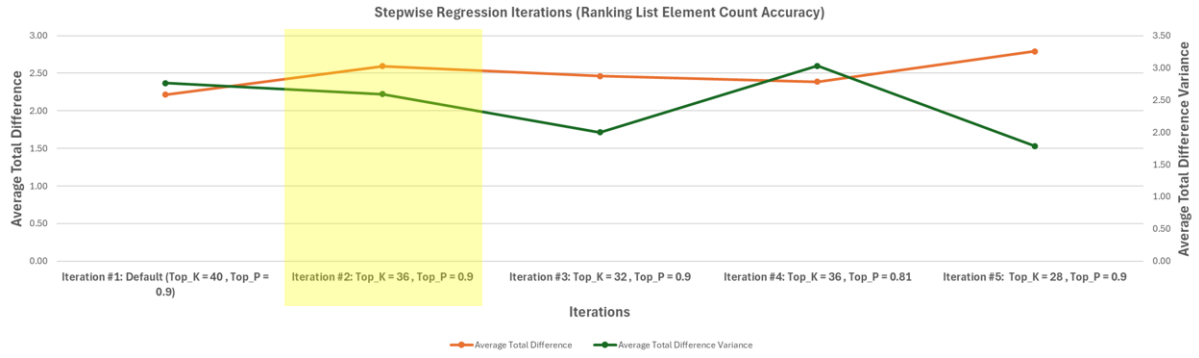


Fig. 18. Stepwise Regression Ranking List Element Count Accuracy

Observations:

From Fig. 17, it is observed that decreasing the Top K by 10% from 40 to 36 and maintaining the Top P value at its default value (0.9), the LLM generates the ranking list with the highest Total Ranking Scores (130) and lowest Ranking Score Variance (0.58). This means that Iteration #2 parameter configuration results in the most accurate and consistent Ranking List among the iterations. Further decreasing the Top K by 10% (as shown in Iteration #3) or decreasing the Top P value by 10% (as shown in Iteration #4), resulted in a decrease in the Total Ranking Score and increase in Ranking Score Variance of the generated ranking list. This means that further iterations of the stepwise regression resulted in the LLM ranking list generation performance to be less accurate and consistent. Further reduction in the Top K value as shown in Iteration #5 also resulted in a reduction in the LLM performance due to the reduction of the Total Ranking Score and increase in Ranking Score Variance.

From Fig. 18, Iteration #2 did not have the lowest Average Total Difference and Average Total Difference Variance. Instead, Iteration #1 have the lowest Average Total Difference and Iteration #5 have the lowest Average Total Difference Variance. However, these metrics, as explained in the earlier section, are not as important as the Total Ranking Scores and Ranking Score Variance metrics due to the fact that in the practical usage of the Ranking List LLM in the project web application, the appearance frequency of each element across the web articles summaries will not be displayed while the actual ranking number of the ranking list element will be displayed in the User Interface of the project web application and thus, much more important. As such, the Average Total Difference and Average Total Difference Variance metrics are purely for further investigation into the non-deterministic ranking mechanism behind the Ranking List LLM purposes only. This is further discussed in the Future Improvements Section 12.1. LLM Future Improvements.

From the aforementioned observations, even though Iteration #2 does not have the lowest Average Total Difference and lowest Average Total Difference Variance, it shows the best Total Ranking Score, lowest Ranking Score Variance. Thus, its parameter values are chosen as the Ranking List LLM parameter configuration.

10.3.2. Discussion and Insights

Decreasing the Top K/Increasing the Top P reduces variance of the LLM for the SAME query as shown in Fig. 18 but could increase the variance of the LLM responses BETWEEN different queries as shown in Fig. 17. When the query is the same, it is expected for repeated LLM response on the same query with decreased Top K/ increase Top P to be more consistent and thus lower overall Average Total Difference/Average Total Difference Variance for each query response.

However, BETWEEN the various queries, the Total Ranking Scores response variance could increase as shown in Fig. 17 when the Top K is further reduced, or the Top P is increased. This could be the strictness in its response generation causing the LLM to deviate more widely between its response for different queries. Moreover, the Ranking List LLM could get so strict in its response that it failed to consider lower probable word tokens that could lead to a more accurate overall response leading to worsening Total Ranking Scores. On the other hand, making the LLM more lenient, as shown in Iteration #3, Hence, further optimization from Iteration #2 causes further decreases in the Total Ranking Scores and could increase the Total Ranking Scores Variance.

As explained in the previous Section 10.3.1, as the Average Total difference/Average Total Difference Variance metrics not as important as the . For the context of this project, the Iteration #2 parameter values are chosen as the Ranking List LLM parameter configuration.

11. Web Application Design and Usage

The Web application initial state is as shown and explained below:

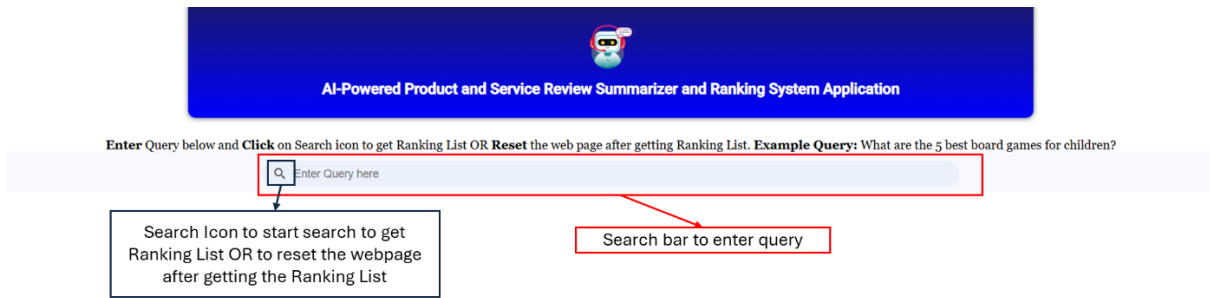


Fig. 19: Web application homepage.

Web application usage Process Flow:

1. To utilize the web application, users simply need to click on the Search bar to enter the query OR click on any of the User's past query inputs from the drop-down overlay.

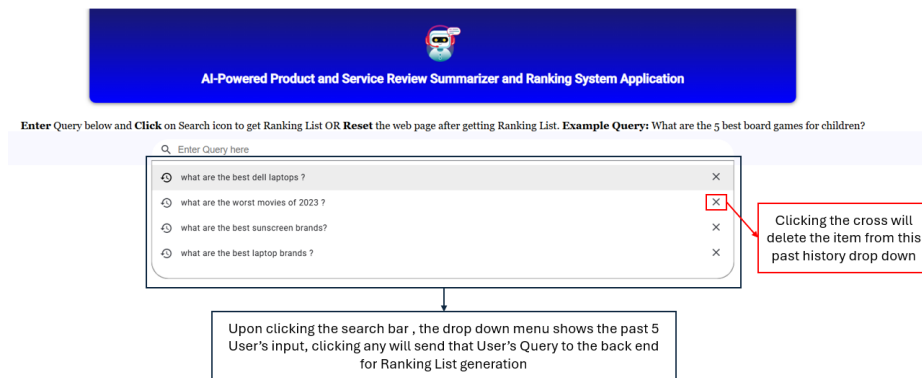


Fig. 20: Web application Search bar dropdown overlay.

If no input is given and User click on the Search icon, an error message will be prompted.

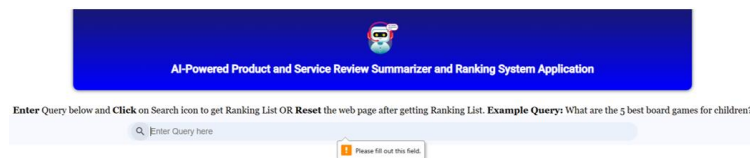


Fig. 21: Web Application no User Query Error

2. After entering Search bar input, the query will be sent to the backend for Ranking List generation.

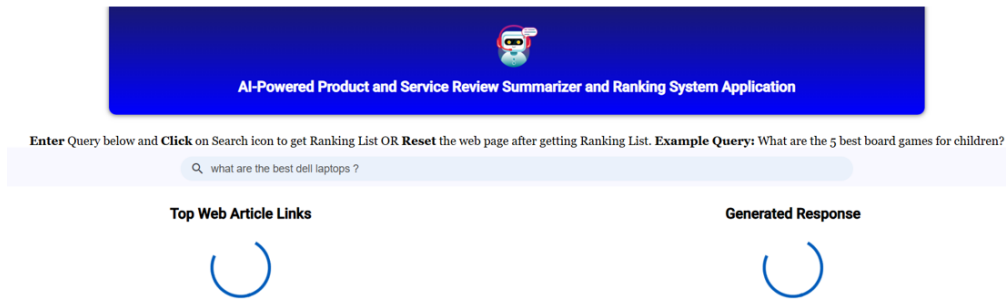


Fig. 22: Web Application loading screens

- Web articles used in the Ranking List LLM will be shown first as the Ranking List will take a longer time to be shown due to the processing time required by the web scraping, Summarizer LLM and Ranking List LLM. This allows users to right-click on the web articles link to view them before the Ranking List is generated.

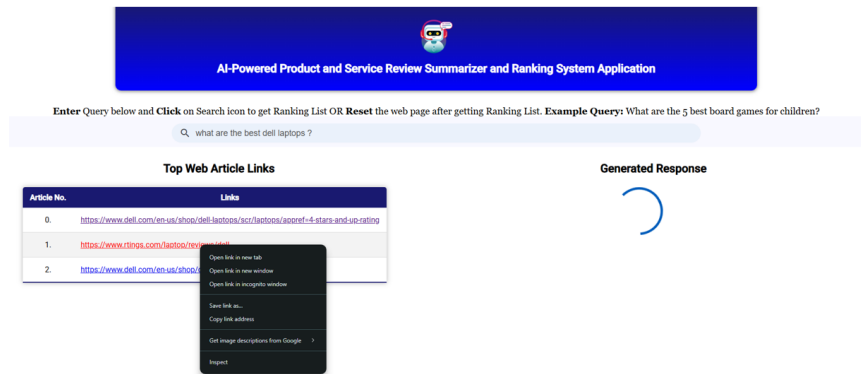


Fig. 23: Web Application loaded Web Article Links

- After Ranking List is shown, user can click on the Search icon to return to the homepage/initial state.

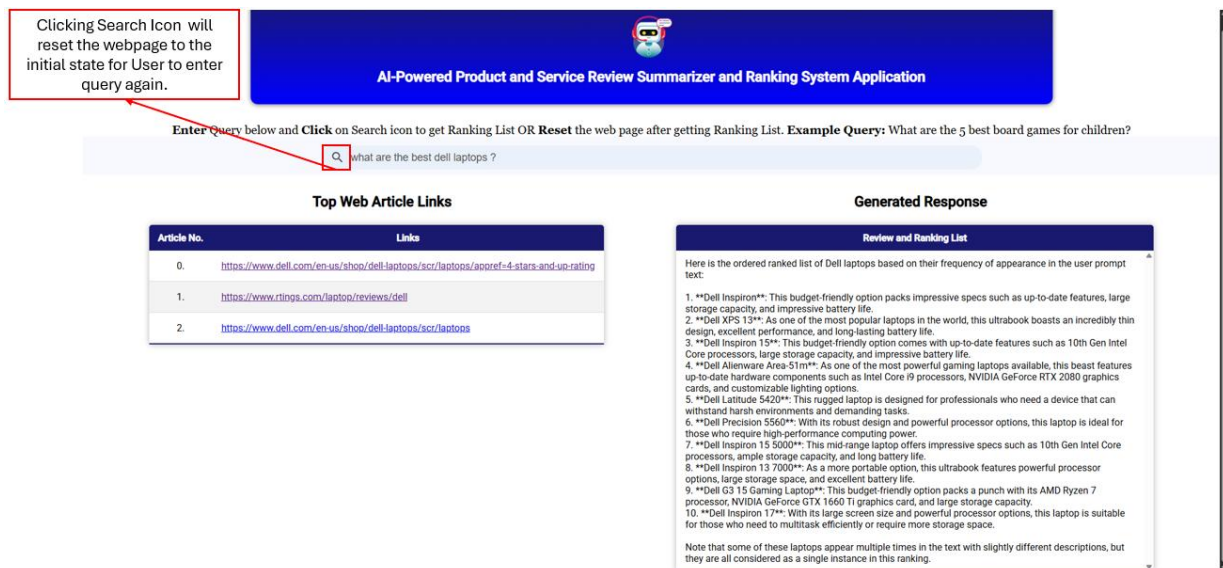


Fig. 24: Web Application loaded Generated Ranking List

12. Further Improvements

12.1. LLM Future Improvements

As shown in Sections 9 and 10 of the Ranking List LLM Evaluation and Optimization, prompt engineering may not be the best method in determining the product/service rankings via appearance frequencies due to the appearance frequencies inconsistencies in the Ranking List LLM as shown in Fig.18 and Fig.25 below. These inconsistencies are due to the LLM's non-deterministic nature which is controlled by the Temperature, Top P and Top K parameters. However, reducing the temperature/ increasing the top P/ decreasing the top K to make the LLM more deterministic might not result in higher Total Ranking Scores as shown in Section 10.3. Stepwise Regression. This is because making the LLM more deterministic is simply making the LLM take a 'greedier' approach in selecting the next most probable token in its response generation. Despite that, taking the next most probable word token during the ranking list generation does not equate to generating an overall correct Ranking List. (Dehtiarov, 2025). Moreover, running LLMs uses Graphic Processing Units (GPUs) which are concurrent processes, thus the process steps execution order may still differ from one response generation to another, even when using the same LLM model with the same input text (Fletcher, 2025).

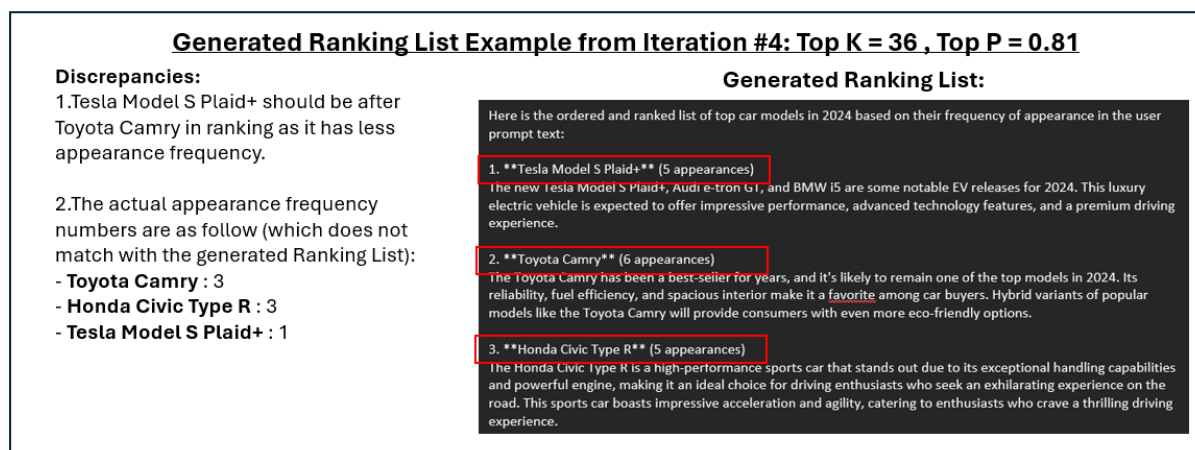


Fig. 25: Appearance Frequency discrepancy example

To mitigate these limitations of prompt engineering, a future improvement would be to utilize Natural Language Processing (NLP) techniques, namely Named Entity Recognition (NER). NER involves identifying certain entities (based on the model training) in a string input and classifying each entity into the correct categories. Hence, NER would enhance the Ranking List LLM's product/service recognition capability, increasing the accuracy of the appearance frequency calculated for each product/service detected, which would potentially lead to a more accurate and consistent Ranking List. (What is named entity recognition?, 2023)

12.2. Web Application Future Improvements

As the current project web application is currently deployed on a local machine, it would be difficult to allow the public to utilize it. Hence, future improvements to the web application include deploying the web application on a cloud platform such as Azure as well as have a user account feature that saves each user's past queries and responses. For deployment, deploying on a cloud platform would allow the web application to dynamically scale in accordance with the users, saving time in managing computing resources and speeds up the deployment process (Advantages and Disadvantages of Cloud Computing, n.d.).

Generating the Ranking List requires the web application backend to first utilize the Google Search API to retrieve the top relevant web articles, then parse the extracted HTML text into the and lastly parse the generated summaries into the Ranking List LLM to generate the required Ranking List for the user as shown in Fig. 2. The total time required for these processes involve several minutes. Hence, having an account for each user to be able to save their past respective queries and responses would be very efficient as the user could simply view their

past queries and responses instead of regenerating the same query responses. The storage account should also have an option for users to delete the responses for certain past queries should they feel that the generated response is too outdated.

13. Conclusion

All in all, the project has been largely insightful and shown potential benefits in utilizing the various prompt engineering techniques to not only summarize text responses but also to generate a ranking list based on the appearance frequency of each entity in the string input. Various prompting techniques (Zero Shot, One Shot, Few Shot and COT) and input variants were evaluated against one another to identify the best prompting technique. Afterwards the chosen LLM model and prompt is optimized using Sensitivity Analysis and Step Wise Regression to get the optimal set of LLM parameters (Temperature, Top K, Top P). Moreover, a web application is also built to showcase the practical usage of the LLMs. Lastly, future improvements were explored to enhance on the current findings/solutions of this project.

14. References

- (n.d.). Retrieved from Ollama: <https://ollama.com/library/deepseek-r1>
- (n.d.). Retrieved from Ollama: <https://ollama.com/library/qwen2>
- Advantages and Disadvantages of Cloud Computing*. (n.d.). Retrieved from Google Cloud: <https://cloud.google.com/learn/advantages-of-cloud-computing>
- Ajay. (13 November, 2024). *Phi 3 – Small Yet Powerful Models from Microsoft*. Retrieved from Analytics Vidhya: <https://www.analyticsvidhya.com/blog/2024/05/phi-3-small-yet-powerful-models-from-microsoft/#h-phi-3-the-next-iteration-of-phi-family>
- An intro to ROUGE, and how to use it to evaluate summaries*. (26 January, 2017). Retrieved from freeCodeCamp: <https://www.freecodecamp.org/news/what-is-rouge-and-how-it-works-for-evaluation-of-summaries-e059fb8ac840/>
- Banghao Chen, Z. Z. (2023). *Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review*. Guangdong, China.
- Behl, H. S. (7 July, 2024). *Understanding User, Assistant, and System Roles in ChatGPT*. Retrieved from Baeldung: <https://www.baeldung.com/cs/chatgpt-api-roles>
- Best practices for prompt engineering with the OpenAI API*. (October, 2024). Retrieved from OpenAI: <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api>
- Chain-of-Thought Prompting*. (n.d.). Retrieved from Prompt Engineering Guide: <https://www.promptingguide.ai/techniques/cot>
- Chibueze, P. (1 February, 2025). *Scaling the Future: How DeepSeek LLM is Redefining Open-Source Language Models*. Retrieved from Medium: <https://paulchibueze.medium.com/scaling-the-future-how-deepseek-llm-is-redefining-open-source-language-models-d48c32fdbd69>
- Deep-Dive-Into-AI-With-MLX-PyTorch*. (December, 2024). Retrieved from GitHub: <https://github.com/neobundy/Deep-Dive-Into-AI-With-MLX-PyTorch/blob/main/deep-dives/001-mistral-7b/README.md>
- DeepSeek LLM Scaling Open-Source Language Models with Longtermism*. (n.d.). Retrieved from arxiv: <https://arxiv.org/pdf/2401.02954>
- DeepSeek-LLM*. (n.d.). Retrieved from GitHub: <https://github.com/deepseek-ai/deepseek-LLM>
- Dehtiarov, V. (7 January, 2025). *How accurate is ChatGPT: long-context degradation and model settings*. Retrieved from sommo: <https://www.sommo.io/blog/how-accurate-is-chatgpt-long-context-degradation-and-model-settings>
- Difference between RDBMS and MongoDB*. (24 January, 2025). Retrieved from geeksforgeeks: <https://www.geeksforgeeks.org/difference-between-rdbms-and-mongodb/>

Fletcher, A. (28 March, 2025). *Non-Determinism in LLMs: Why Generation Isn't Exactly Reproducible*. Retrieved from LinkedIn: <https://www.linkedin.com/pulse/non-determinism-llms-why-generations-isnt-exactly-adam-6mhme>

Foote, K. D. (28 December, 2023). *A Brief History of Large Language Models*. Retrieved from Dataversity: <https://www.dataversity.net/a-brief-history-of-large-language-models/>

Friesen, J. (3 March, 2019). *Why We Chose Angular Over React*. Retrieved from Medium: <https://medium.com/@jacobfriesen/why-we-chose-angular-over-react-e633b9d5d155>

How Many Online Stores Are There? . (17 July, 2021). Retrieved from digitalintheround: <https://digitalintheround.com/how-many-online-stores-are-there/>

Introducing Llama 3.1: Our most capable models to date. (23 July, 2024). Retrieved from Meta: <https://ai.meta.com/blog/meta-llama-3-1/>

Kızılırmak, E. (2 August, 2023). *Text Summarization: How To Calculate Rouge Score*. Retrieved from Medium: <https://medium.com/@eren9677/text-summarization-387836c9e178>

meta-llama. (n.d.). Retrieved from GitHub: https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/MODEL_CARD.md

Modeling Ollama. (June, 2023). Retrieved from deepchecks: <https://www.deepchecks.com/llm-tools/ollama/>

Ollama modelfile.md. (February, 2025). Retrieved from GitHub: <https://github.com/ollama/ollama/blob/main/docs/modelfile.md>

Özbolat, H. (28 September, 2023). *Text Summarization: How to Calculate BertScore*. Retrieved from Medium: <https://haticeozbolat17.medium.com/text-summarization-how-to-calculate-bertscore-771a51022964>

Papers Explained 64: Mistral. (23 October, 2023). Retrieved from Medium: <https://medium.com/dair-ai/papers-explained-mistral-7b-b9632dedf580>

phi3. (n.d.). Retrieved from Ollama: <https://ollama.com/library/phi3>

Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. (n.d.). Retrieved from arxiv: <https://arxiv.org/pdf/2404.14219>

Python FastAPI vs Flask: A Detailed Comparison. (n.d.). Retrieved from Turing: <https://www.turing.com/kb/fastapi-vs-flask-a-detailed-comparison#what-is-fastapi?>

Qwen Team, A. G. (n.d.). *Qwen2 Technical Report*. Retrieved from arxiv: <https://arxiv.org/html/2407.10671v1>

Qwen2. (n.d.). Retrieved from Hugging Face: https://huggingface.co/docs/transformers/en/model_doc/qwen2

Santhosh, S. (16 April, 2023). *Understanding BLEU and ROUGE score for NLP evaluation*. Retrieved from Medium: <https://medium.com/@sthanikamsanthosh1994/understanding-bleu-and-rouge-score-for-nlp-evaluation-1ab334ecadcb>

- Sojasingarayar, A. (15 January, 2024). *BERTScore Explained in 5 minutes*. Retrieved from Medium: <https://medium.com/@abonia/bertscore-explained-in-5-minutes-0b98553bfb71>
- Stihec, J. (1 October, 2024). *Zero-Shot vs. Few-Shot Prompting: Comparison and Examples*. Retrieved from shelf: <https://shelf.io/blog/zero-shot-and-few-shot-prompting/>
- Taheer, F. (13 February, 2025). *Online Shopping Statistics: What to Expect in 2025?* Retrieved from OptimMonster: <https://optimmonster.com/online-shopping-statistics/>
- Tardi, C. (2 April, 2024). *What Is Moore's Law and Is It Still True?* Retrieved from Investopedia: <https://www.investopedia.com/terms/m/mooreslaw.asp>
- The history, timeline, and future of LLMs*. (26 July, 2023). Retrieved from Toloka: <https://toloka.ai/blog/history-of-llms/>
- Top 10 Angular features you should know*. (14 May, 2024). Retrieved from geeksforgeeks: <https://www.geeksforgeeks.org/top-10-angular-features-you-should-know/>
- What is named entity recognition?* (26 August, 2023). Retrieved from IBM: <https://www.ibm.com/think/topics/named-entity-recognition>
- What is Ollama? Understanding how it works, main features and models*. (25 November, 2024). Retrieved from Hostinger Tutorials: <https://www.hostinger.com/tutorials/what-is-ollama>