National College of Ireland

Leveraging AI for Agile Backlog Management Using LLMs: A Comprehensive Approach

MSc Research Project
Ai For Business

Najam Ul Hassan KHan
Student ID: 23164816

School of Computing
National College of Ireland

Supervisor: Dr. Devanshu Anand

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Najam Ul Hassan Khan
Student ID:	23164816
Programme:	Ai For Business
Year:	2024
Module:	MSc Research Project
Supervisor:	Dr. Devanshu Anand
Submission Due Date:	12/08/2024
Project Title:	Leveraging AI for Agile Backlog Management Using LLMs: A Comprehensive Approach
Word Count:	6421
Page Count:	32

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Leveraging AI for Agile Backlog Management Using LLMs: A Comprehensive Approach

Najam Ul Hassan Khan
23164816

Abstract

Agile project management is mostly preferred due to its flexibility, rapid advancement, and ability to align with stakeholders. Methodologies such as Scrum, Kanban, Lean, Extreme Programming (XP), and Adaptive Project Framework (APF) facilitate the process of incremental improvement. Nevertheless, the task of overseeing the product backlog, which is a constantly changing inventory of innovations, bug fixes, and enhancements, can require a significant amount of effort. LLMs like Llama, T5, ChatGPT, Mistral, and Gemini have emerged as promising tools for automating task management and improving productivity. The study aims to assess the usage of a multi-pipeline of LLMs in handling backlogs, namely by improving productivity and controlling timeframes.The results will aid in enhancing agile processes and incorporating AI into project management.

Keywords: Agile, Back Log management, LLMs, Similarity Scores, Latent Similarity

1 Introduction

Agile methodology Srivastava et al. (2017) in project management is a planned approach to managing the project in a more flexible and coherent manner by dividing the work into smaller and more manageable phases with equal stress on continuous improvement. It is a system that also goes around in cycles, consisting of planning, implementing, and evaluating. Analyzing the outcomes of utilizing Agile is preferred in the management of projects because of the versatility associated with it, ability to accommodate new changes, and emphasis it places on the customer. Many teams use this methodology to obtain the subsequent benefits of Agile:

- **Swift advancement:** By efficiently shortening the duration required to finish different phases of a project, teams may promptly obtain input.
- **Ensuring congruence between customers and stakeholders:** By prioritising customer needs and incorporating stakeholder feedback, the Agile team is effectively positioned to provide outcomes.
- **Iterative enhancement:** The Agile project management methodology enables teams to incrementally work on assignments.

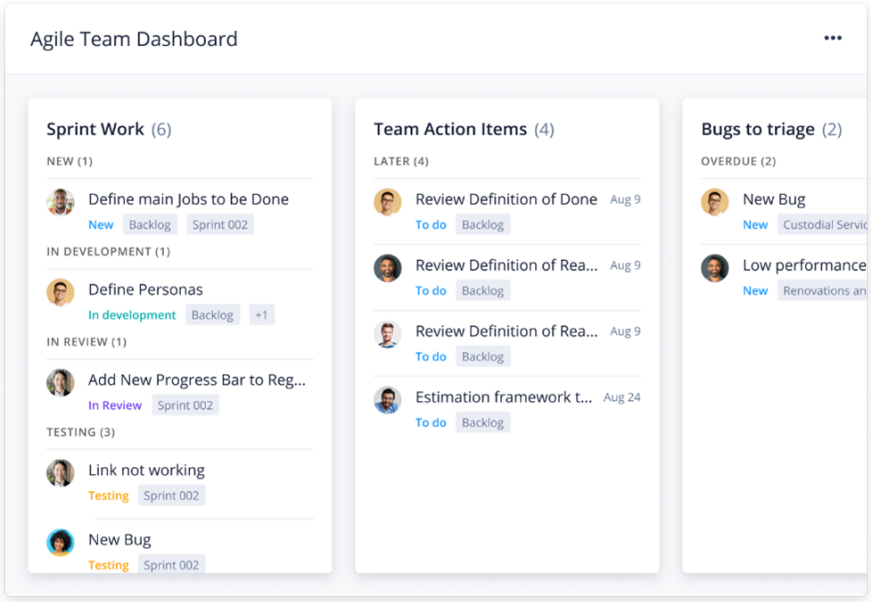

Figure 1: Agile Methodology framework (**Source: Attlassian.com**)

Agile project management constitutes various techniques including Scrum, Kanban, the Extreme Programming (XP), and the Adaptive Project Framework (APF). (**Atlassian.com**)

1.1 Research Motivation

In Scrum and other Agile techniques Kumar and Bhatia (2012), the product backlog is a meticulously ordered inventory of all potential components that can be included in the final product. This encompasses several elements like features, bug patches, functional and non-functional needs, and process enhancements. The list is dynamic and adapts to the evolving requirements and desires of users, consumers, and stakeholders.

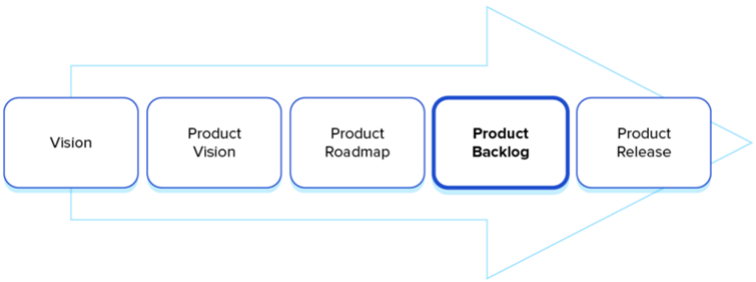

Figure 2: Product Backlog in the product management (**Source: Rapidr.io**)

The purpose of backlog management is to offer a concise and comprehensive understanding of the tasks required to develop or improve the product. The development team chooses tasks for each sprint and defines a time frame to finish it.

ToDo List

ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		**30**	

Figure 3: Backlog in a simplified manner (**Source: International Scrum Institute**)

The main challenge in managing a backlog is that it takes huge efforts and time from a manager when managing multiple teams and projects. With the rise of LLMs like ChatGPT, Gemini, Llama, etc., information generation is fast.

1.2 Research Problem

An improperly organised backlog looks like a lengthy, continuously expanding record that archives all the ideas and opinions of your team members on the project. This extensive inventory hinders your ability to prioritize, and you face difficulty choosing the appropriate tasks for long-term development cycles or short-term sprints.

1.3 Research Objective

Project Manager uses Microsoft Teams AI to recommend task durations and efforts in work plans, using NLP and Azure Open AI. The system will provide the schedule of the project with the analysis of the project name and description. Project Input is made, and in the case of a detailed project, the project manager should make a revision task and can optimize the task as necessary. Our objective is to make a system that could be integrated into any CRM and perform this functionality.

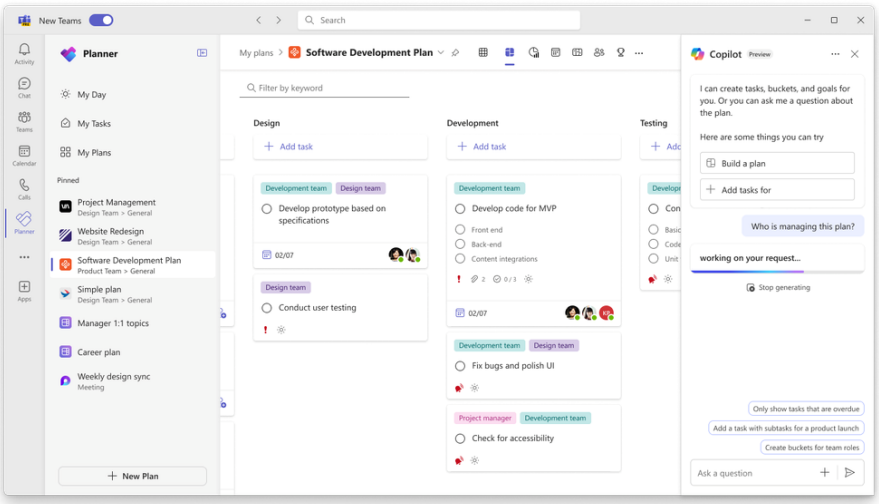

Figure 4: Microsoft Teams, automatic task summariser and manager for the Agile process (**Source: Microsoft Teams**)

1.4 Research Questions

Through this research, we would be investigating the following:

RQ1: How to make a generic copilot that will help in making a summary and the severity of the summary tasks based on all the details in the Jira?

RQ2: Can any LLM be used that will not show one client/users/different project details in a summary task? In other words, can we make a data sovereignty task summariser and scheduler for independent users?

1.5 Research Outcomes

In this research, we will make a LLM-Task scheduler that can be plugged into any agile methodology's product, and this can help in making a better task summariser. This task summariser can not only enhance productivity but also manage timelines effectively. The performance of the deployed model framework will be evaluated and discussed properly.

2 Related Work

OpenAI's GPT and Google's Gemini, which are substantial language models, present possibilities for facilitating software engineering procedures. Utilising a LLM, software engineering may be enhanced by providing developers with conversational assistance and expert knowledge across the whole software development process. The current applications encompass a wide range of tasks, including extracting requirements, resolving ambiguities, generating code and test cases, doing code reviews and translations, as well as verifying and repairing software vulnerabilities Belzner et al. (2023). In the context of agile software development, the task of preserving user stories is of utmost importance, although it presents difficulties. This research investigates the utilisation of LLMs

to enhance the calibre of user stories in agile teamsZhang et al. (2024). Effective team communication is crucial to the functioning of any software development company, but it may be difficult to coordinate and oversee. Frequently, meetings fail to adhere to norms and encounter problems that hinder their effectiveness and productivity or prolong the process of reaching decisions Cabrero-Daniel et al. (2024). Occasionally, the guidelines impose a heavy load on the development teams and necessitate adaptation. The Post-Rolling Refinement Model (PRIME), a Scaled Agile paradigm developed by the Austrian Post Niessl et al. (2023), seeks to alleviate this challenge.

2.1 Research Towards Generative AI for Backlog Management in Agile Process

Flexible and customer-centric, agile is an iterative method of project management. The Agile Manifesto Fowler and Highsmith (2001), published in 2001, is the foundation of agile methodology, which promotes adaptive planning, evolutionary development, early delivery, and continuous improvement while also supporting quick and adaptable responses to change. Splitting the project into smaller, more manageable units called sprints is what the Agile methodology does, as opposed to the standard waterfall process.

This study Nasiri and Lahmer (2024) proposes a technique for refining backlogs in Agile operations. It entails categorising user stories and identifying their shared characteristics. Subsequently, the k-means algorithm is employed to group these stories into clusters, while the SBERT model is used to ascertain the semantic commonalities inside each cluster. In addition, specific criteria were created to discover pairs of user stories with contrasting meanings by leveraging the WordNet API and the typed dependencies generated by NLP technologies. Comparative studies have revealed that the SBERT model, namely "paraphrase-monet-base-v2," regularly outperformed other models in terms of precision, recall, and F1 scores for detecting comparable user tales.

This research Sun and Shao (2023) presents a system that makes backlog management more efficient and easier to use by automating crucial activities. Starting with multimodal summarization driven by artificial intelligence, it gathers important project details from many data sources, including video calls and instant messages, to paint a complete picture of the project's progress. MAPM can improve productivity and enable real-time decision-making, especially in hybrid work environments. The paper Daniella et al. (2023) conducted experiments to determine the efficacy of an Artificial Neural Network (ANN) for user story classification. Three layers make up the ANN model: input, hidden, and output. The input and hidden layers used ReLU activation, the output layer used Softmax, and the Adam optimizer used a categorical cross-entropy loss function. Results from multiple experiments demonstrated that the model was overfit Dhruva et al. (2024), with training accuracy reaching 90% and validation accuracy falling as low as 30%. The study found that the model's performance was hindered by the unrestricted language of the dataset and the imbalance in sample sizes. If we want user narrative datasets to be better at classification, we need to work on these issues.

2.2 Emergence of LLMs and SLMs on task management in the Agile process

In research by Sami et al. (2024) concerned with how Generative AI, particularly LLMs such as GPT incorporated in software engineering (SE) processes primarily within the Agile task management system. With reference to the definitions of intelligent systems, it must be mentioned that LLMs may contribute to automation of tasks such as error identification, generation of code snippets, and preparation of documentation, which, in turn, contribute to the enhancement of efficiency and quality of software development.

This paper Tikayat Ray et al. (2023) focuses on the technique aimed at automating and normalizing the process of requirements analysis with the help of two LLMs, aeroBERT-NER and aeroBERT-Classifier, introduced into the Agile processes. In the paper Cabrero-Daniel et al. (2024), a procedure for converting customer requirements into a format understandable by machines was introduced. Thus, to make the concept of machine-generated templates usable under Agile methodologies in a real-life environment, it points out the need for domain knowledge to enhance LLM outcomes. Agility stands to benefit a great deal from LLMs in solution integration since it increases interactivity, eliminates manual expenses, and enhances the quality of requirement translations.

This paper Kim et al. (2024) shall focus on realising the use of LLMs in contexts of Agile, focusing on how the introduction of these LLMs affects the interpersonal processes of collaboration and scheduling. AI is explained as essential for enhancing the organizational efficiency of meetings, delivering prompt analytics, and supporting other Agile principles such as time division and collaboration expectations. Among the technological problems that emerged and were corrected with the progression of the interventions were data accuracy and real-time data integration. Due to these and similar problems, the architecture of AI has been updated iteratively Tikayat Ray et al. (2023). Although the participants displayed skepticism regarding the use of AI in the first place, the majority were convinced that the system test analyses may enhance team chemistry and decision making. To sum up the article, some recommendations concerning the intersection of AI assistants and agile approaches are given.

2.3 Improvement in the decision-making process of the response generated using LLMs

This research Kim et al. (2024) assists doctors in decision-making through the construction of MDAgents that employ LLMs. From the capacity of MDA, agents determine the level of difficulty of the questions and, in the process, define the operationalization of the LLMs. For cases that are less complex and more straightforward, a single PCP LLM suffices, and it is referred to as the Relationship GP LLM.

Thus, this paper Lakkaraju et al. (2023) compares the bias and performance of three chatbots: SafeFinance, ChatGPT (OpenAI), and Bard (Google). Two distinct situations are used to conduct the tests: One is known as linked product discovery (LPD), and it entails provision of the source of the response. The other is called no-link product discovery (NLPD), and it does not. The chatbot is checked as to its correctness in the answers provided through the use of Jaccard distance. Both ChatGPT and Bard Feng et al. (2024) present considerable favoritism towards the users based on the name provided, yet the results achieved by both systems are significantly different and far from

the predictions through the source in the case of the LPD scenario. Thus, although making replies accurately and impartially, SafeFinance can be less flexible compared to LLM-based systems. Among the potential developments of the discussed approach for increasing the efficiency of chatbots in various financial applications, in further studies, it is possible to mention the combination of the advantages of both techniques.

This study Yang et al. (2024) analyses the differences in voting behaviours of people and a LLM based on a study of information gathered from a simulated Participatory Budgeting experiment in Zurich. The human voters included 180 college students, and the students voted for 24 projects to fund with CHF 60,000. When LLMs reacted to the same instructions and prompts, the authors were able to mimic voting as a human being, especially LLaMA-2 and GPT-4 Turbo.

Throughout the assessment of the ethical impacts of the LLM He et al. (2024) prejudices and stereotypes in the framework of democratic procedures, cautious adaptation was recommended. In an effort to move the LLM outputs closer to human patterns, persona-based simulations were performed, but it was stated that there are still prejudices. Due to the subjectivity of explaining AI's decision-making capabilities, co-text (CoT) reasoning was applied, and it was noted to have a minimal impact on the percentage of votes.

2.4 Research Summary

GPT and Google's Bard are some of the LLMs that are transforming software engineering and agile activities like requirement extraction, code generation, and backlog handling. Research shows that the introduction of LLMs improves the quality of the user stories, management efficiency, and cohesion between the teams. For instance, AI in summarization and job priority ensures that the workflow is made more flexible in cases where its necessary. Furthermore, MDAgents also enhance the medical prognosis concerning the nature of queries and the proper LLMs to apply.

2.5 Research Niche

In this research work, we are going to explore different usages of LLMs like GPT, Mistral, Llama, etc. for different applications that include summarization, Q&A, search, sentiment analysis, and text generation. Unfortunately, there are relatively few insights into how exactly such models work in different scenarios, let alone places and contexts. In addition, comparisons such as cost of servers, storage, and computation play a rather vital role in the decision to choose LLMs. From the analysis that has been conducted in this paper, most LLMs have the potential to improve SE and agile relative to task automation and decision-making.

3 Methodology

Advancements in multimodal AI allow teams to create content across several forms of media, such as text, images, and video.

3.1 Research Resources – API's from Hugging Face

Hugging Face Transformers Shen et al. (2024) is a Python library that is open-source and grants access to several pre-trained Transformers models for various multi-modal tasks such as NLP, computer vision, audio tasks, and more.

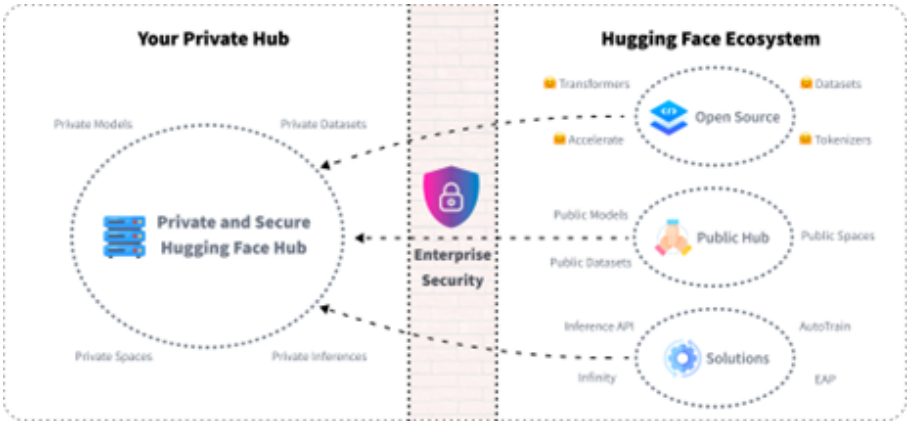

Figure 5: An example of how to use and integrate the HF Api's (Source: Hugging Face)

3.2 Research Resources – Modelling with Mistral, ChatGPT, Llama, Gemini, one more

3.2.1 Gemini Gemini AI's Team et al. (2023) modular design addresses various tasks, including language understanding, decision-making, and planning. All these components work together seamlessly, utilizing complex transformer networks to understand the context, subtleties, and complexities of human language.

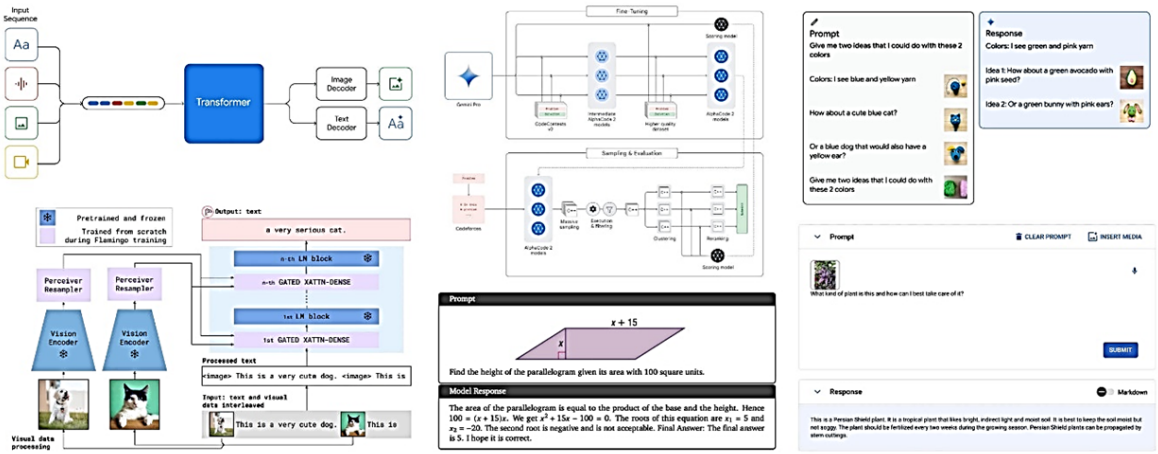

Figure 6: Gemini by Google **(Source: Team, G., 2023)**

Training Gemini AI involves massive datasets covering diverse languages, contexts, and

problem-solving scenarios, including text corpora, interactive simulations, and real-world problem scenarios. Reinforcement learning (RL) is a crucial part of this training, teaching the system to make decisions based on reinforcement for good results and penalties for bad ones.

3.2.2 Mistral Mistral Jiang et al. (2023) improves upon the successes of previous models such as GPT-2 and GPT-3 by employing a transformer-based architecture similar to those of its predecessors with enhanced dense and self-connected layers.

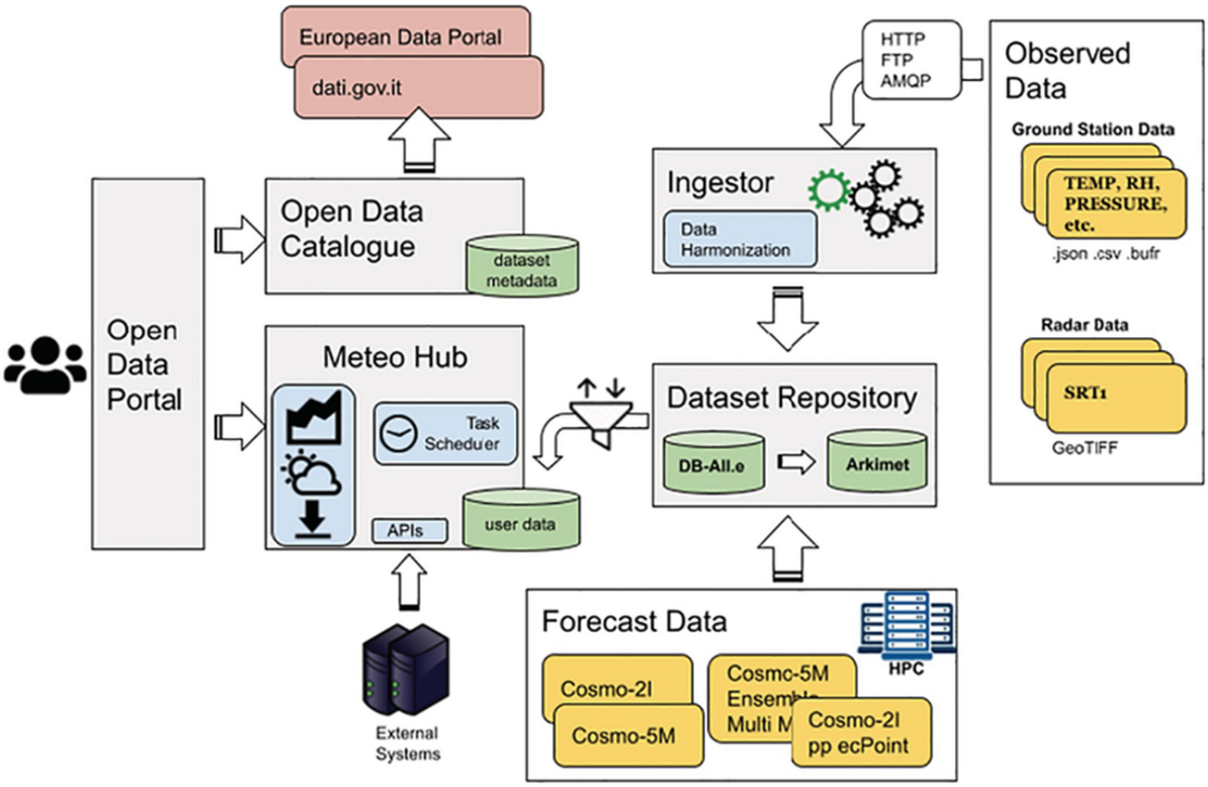

Figure 7: High Level representation of Mistral **(Jiang, A.Q., et. al., 2023)**

Mistral's role as a reliable and morally upright artificial intelligence system is reinforced by the agency's commitment to thorough testing and continuous monitoring, which serves to prevent the creation of harmful or misleading information.

3.2.3 Google T5 T5 (Google Text to Text Transformer) Roberts et al. (2023) employs a technique that involves treating problems that include text conversion. T5 has been performing tasks such as translation, summarization, question-answering, and text generation efficiently with vector transformations.

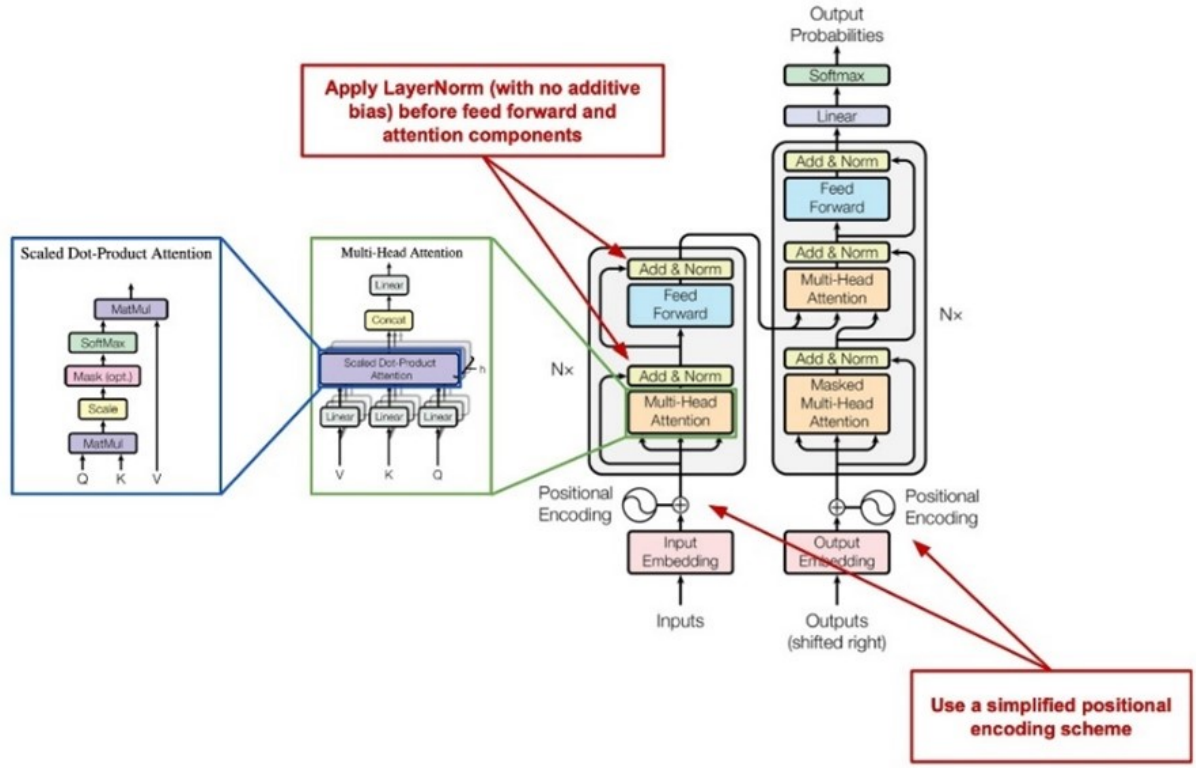

Figure 8: Network architecture of Google T5 **(Roberts, A., et. al., 2023)**

The efficacy of T5 can be attributed to its extensive training on diverse textual material, encompassing books, journals, and web content. This is equipped with a comprehensive comprehension of the complexities and nuances from understanding so much of huge corpus.

3.3 Research Resources – Similarity Scoring Models

The Sentence Transformer model 'all-MiniLM-L6-v2' analyses the semntic similarity among textual vectors. It involves the generation of high-dimensional vector embeddings for sentences. The vectors are subsequently compared using the cosine similarity technique to determine their similarity scores. This methodology ensures that the similarity scores accurately reflect the semantic relationship between texts.

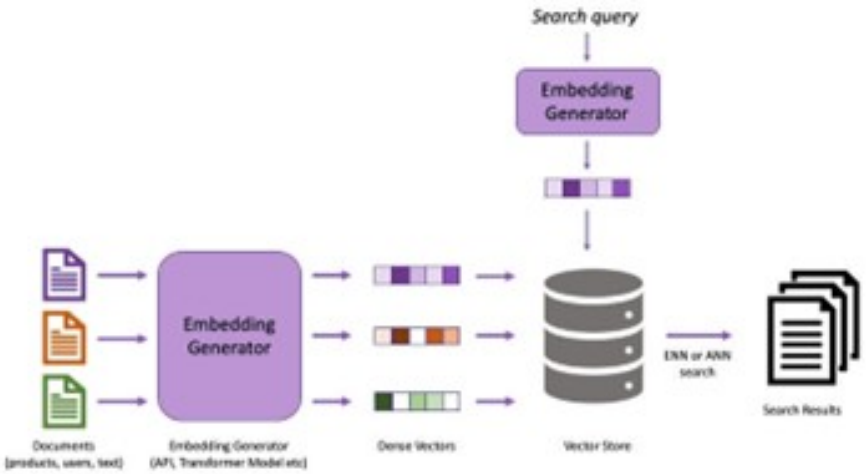

Figure 9: Network architecture of Sentence Transformers

3.3.1 Sentence Transformers The Sentence Transformers Nikolaev and Padó (2023) library is specifically designed to generate sentence-level embeddings. This library enhances the existing transformer models by generating dense vector representations for semantic analysis.

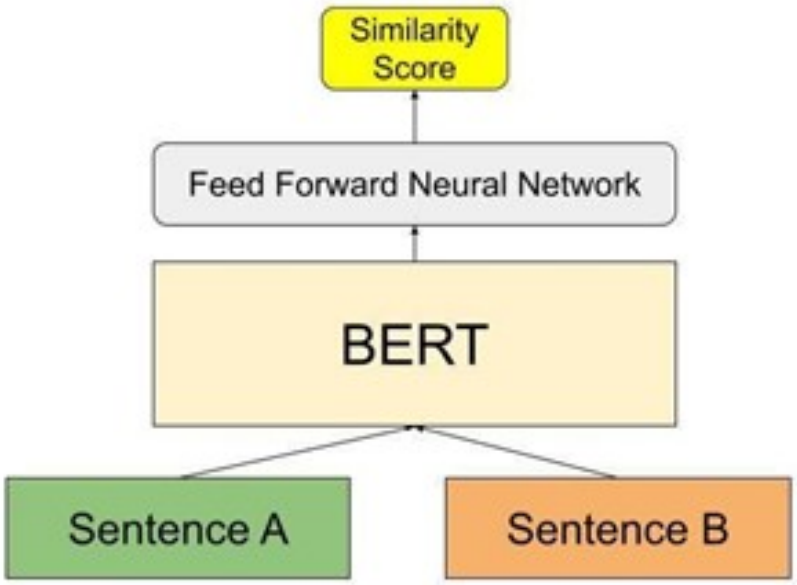

Figure 10: Basic Network architecture of generation of the similarity scores Nikolaev and Padó (2023)

The 'paraphrase-MiniLM-L6-v2' utilizes the MiniLM architecture to provide enhanced sentence embeddings. This model provides a harmonious combination of computational efficiency and semantic accuracy using a transformer that is both concise and efficient.

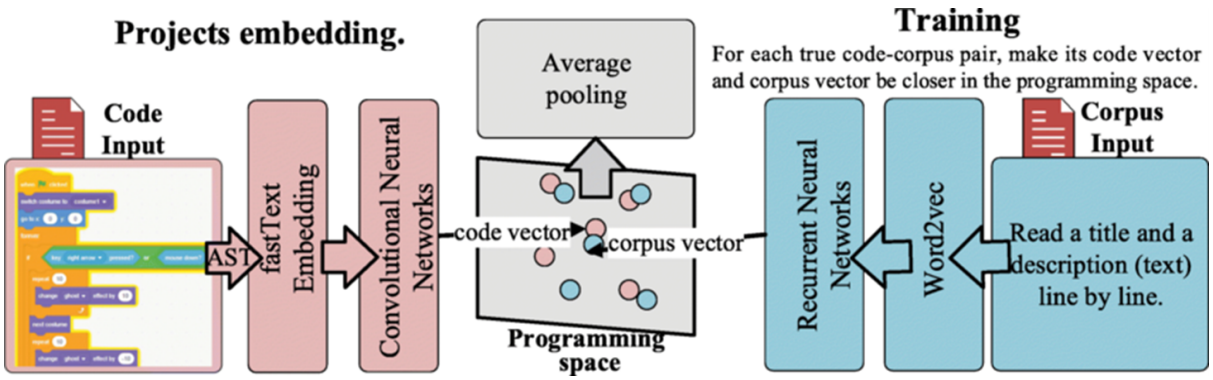

Figure 11: Creat ion of embeddings and the process for training and generation of weights Nikolaev and Padó (2023)

3.3.2 Cosine similarity In order to assess the similarity between the vector embeddings generated by the 'all-MiniLM-L6-v2' model, cosine similarity metrics were employed. The cosine similarity measure calculates the cosine of the angle between two vectors in the embedding space, resulting in a similarity score ranging from -1 to 1.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 12: Cosine Similarity

4 Implementation

4.1 Algorithm Steps

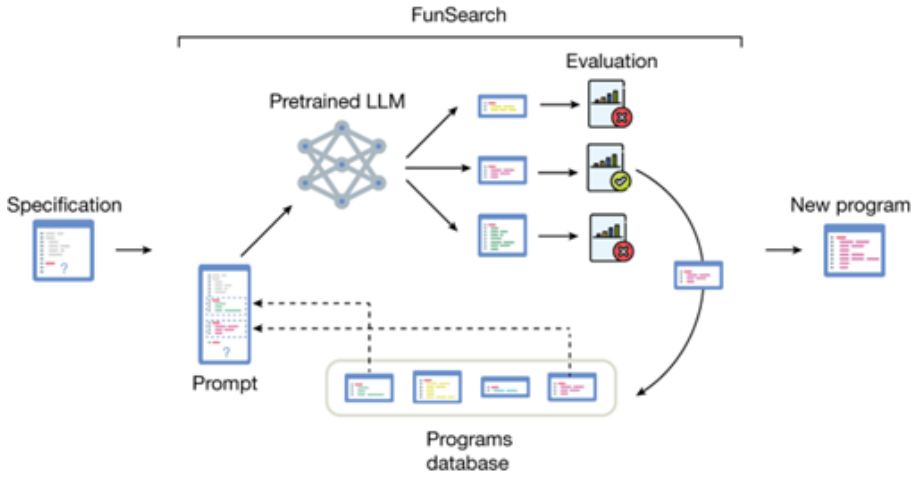

Figure 13: Back-Log analysis for the auto generated responses

Steps for Implementing LLMs in JIRA Backlog Management:

Step 1: Establishing clear objectives defining the scope of the project – The copilot will assist with essential activities including backlog refining, user story building, prioritizing, and task management. To identify the objectives of the implementation, with a specific emphasis on enhancing efficiency, precision, and user contentment.

Step 2: Choosing Suitable LLMs – To Assess several LLMs (such as Mistral, Gemini, Llama, Phi, GPT-4, Bard and T5) by considering performance indicators, cost, storage demands, response time, geographical accessibility, and integration simplicity.

Step 3: Develop the structure and organization of the system's architecture – To develop a proper strategy that can include the LLMs into JIRA, utilizing APIs or plugins to enable smooth and effortless communication, and to develop a robust backend architecture capable of efficiently managing LLM (Large Load Management) requests with a focus on scalability and dependability.

Step 4: Implement the Copilot functionalities – Backlog Refinement: For this, we will utilize NLP tools to examine and refine user stories by identifying duplicate entries and proposing improvements.

4.1.1 *Sub – step 1: User Story Generation – The best models to be employed are LLMs to produce user stories based on natural language needs supplied by users.*

4.1.2 *Sub – step 2: Prioritization – In this search, we will use a similarity matrix and ROUGH scores with the assistance of Learning-based Language Models (LLMs).*

Step 5: Develop a graphical user interface (GUI) – We will then create an intuitive interface in JIRA that allows users to easily engage with the copilot. We will make sure that the UI offers distinct choices for various copilot capabilities and presents LLM-generated ideas in a visible manner.

Step 6: Perform integration and testing – We will integrate the copilot with JIRA to establish seamless data synchronization between the platform and the LLMs.

4.2 Data

Using two comprehensive chat logs, this study reconstructs the conversations that occurred amongst project managers while they worked to establish a "Financial Analysis Tool." In the first log, we see a situation where everyone is on the same page, tasks are finished on time, and teamwork is smooth. We will use the following tags: Information Retrieval, Summary, Named Entity Recognition, and Task Assignment Tracking.

4.2.1 Synthetic Chatlog Two detailed chat logs were used in the project management simulation to show how team members interacted, what they were assigned, and how progress was tracked while building a "Financial Analysis Tool." Conversation log provides a comprehensive analysis of the complex dynamics and practical challenges that emerge in professional environments, acting as a notable case study of project management in practice.

```
chat_log = """
[Chat Log Start]

[2024-07-15 09:00 AM]

Manager: Hi team! Let's kick off our new project, the "Financial Analysis Tool".

Manager: Adding team members: Developer 1, Developer 2, Developer 3, Developer 4, Developer 5.

[2024-07-15 09:05 AM]

Developer 1: Hi everyone!

Developer 2: Hello team!

Developer 3: Hi there!

Developer 4: Hi everyone!

Developer 5: Hello team!

[2024-07-15 09:10 AM]

Manager: Our goal is to develop a powerful Financial Analysis Tool to enhance our analytical capabilities.

Manager: Here are the main tasks and their breakdowns:

[2024-07-15 09:15 AM]
```

Figure 14: Example of the head of the dataset

4.2.2 Question Answer Pairs The primary objective of the first chatlog is to analyse LLMs' response in mainly "Task Assignment Tracking", "NER", and "Information Retrieval".

```
questions = [
    # Information Retrieval
    {
        "question": "Who are the team members added by the manager?",
        "answer": "Developer 1, Developer 2, Developer 3, Developer 4, Developer 5",
        "tag": "Information Retrieval"
    },
    {
        "question": "What is the name of the new project?",
        "answer": "Financial Analysis Tool",
        "tag": "Information Retrieval"
    },
    {
        "question": "What is the goal of the project?",
        "answer": "To develop a powerful Financial Analysis Tool to enhance analytical capabilities.",
        "tag": "Information Retrieval"
```

Figure 15: Example of the head of the dataset

4.3 LLM answer generation

Selecting an appropriate model for text generation, or q&a, is the initial step in evaluating the effectiveness of various language models. This evaluation is done by measuring cosine similarity on embeddings generated by paraphrase-MiniLM-L6-v2 api modules.

4.3.1 Gemini

```
def setup_gemini():
    GOOGLE_API_KEY=userdata.get('GOOGLE_API_KEY')
    genai.configure(api_key=GOOGLE_API_KEY)
    model = genai.GenerativeModel('gemini-1.0-pro')
    return model

def get_llm_response(prompt , model):
    response = model.generate_content(prompt)
    return response.text
```

Figure 16: Code snippet for Gemini

Access to the capabilities of the generative model can be obtained by utilizing the API key, which serves as a form of authentication. Defining a prompt is the initial stage in the content generation process, which occurs after the generative model has been set up. The provided prompt serves as the input for the generative model, containing the necessary context or query for the model to effectively generate a response.

4.3.2 Mistral

Figure 17: Code snippet for Mistral

4.4 T5-Models Answer Generation

We used two different versions of the T5-Flan model, 't5_flan_large' and 't5_flan_base', for generating the query responses. Due to the models' 512-token limitation, summarizing the conversation logs was an essential first step, as they often exceeded this limit. A recursive summation strategy had to be developed for the summary process in order to successfully condense the information while preserving the vital material for providing accurate answers.

```python
from transformers import T5Tokenizer, T5ForConditionalGeneration
import torch

device = "cuda" if torch.cuda.is_available() else "cpu"

tokenizer = T5Tokenizer.from_pretrained("google/flan-t5-large")
model = T5ForConditionalGeneration.from_pretrained("google/flan-t5-large").to(device)

def summarize(text, maxSummarylength=150):
    input_text = f"summarize the chat given: {text}"
    input_ids = tokenizer(input_text, return_tensors="pt", max_length=512, truncation=True).input_ids.to(device)
    summary_ids = model.generate(input_ids, max_length=maxSummarylength, min_length=30, length_penalty=2.0, num_beams=4, early_stopping=True)
    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
    return summary

def split_text_into_pieces(text, max_tokens=900, overlapPercent=0):
    tokens = tokenizer.tokenize(text)
    overlap_tokens = int(max_tokens * overlapPercent / 100)
    pieces = [tokens[i:i + max_tokens] for i in range(0, len(tokens), max_tokens - overlap_tokens)]
    text_pieces = [tokenizer.decode(tokenizer.convert_tokens_to_ids(piece), skip_special_tokens=True) for piece in pieces]
    return text_pieces

def recursive_summarize(text, max_length=500):
    tokens = tokenizer.tokenize(text)
    pieces = split_text_into_pieces(text, max_tokens=max_length)
    summaries = [summarize(piece, maxSummarylength=max_length // 3 * 2) for piece in pieces]
    concatenated_summary = ' '.join(summaries)
    if len(tokenizer.tokenize(concatenated_summary)) > max_length:
        return recursive_summarize(concatenated_summary, max_length=max_length)
    return concatenated_summary

def summarize_chat_log(chat_log):
    return recursive_summarize(chat_log)
```

Figure 18: Code snippet for T5

An organized prompt that incorporates the collected chat log with the specific question asked is part of the systematic approach to answer generation. After that, the T5-Flan model receives the prompt and is tasked with creating a response.

```python
def get_model_response(prompt, max_length=512):
    input_ids = tokenizer(prompt, return_tensors="pt", max_length=512, truncation=True).input_ids.to(device)
    output_ids = model.generate(input_ids, max_length=max_length, num_beams=4, early_stopping=True)
    response = tokenizer.decode(output_ids[0], skip_special_tokens=True)
    return response
```

Figure 19: Response Generation

4.5 Prompting

It is crucial to create effective prompts to fully utilize the potential of LLMs and generate accurate and contextually appropriate responses. Throughout our research, we employed different prompting tactics tailored to three distinct models: T5-Flan, Gemini, and Mistral. Each model required a unique approach to prompting, tailored to its architecture and functional capabilities.

5 Results and Analysis

5.1 Gemini

5.1.1 Case 1 When it comes to Information Retrieval, the Gemini model demonstrated a lot of accuracy concerning the average of similarity, which is 0.9004. In the test where Named Entity Recognition was applied the mean similarity score was 0.7326. The scores figured were rather inconsistent and varied between 0.4251 and 1.0000. As for the tag term 'Summary', the calculated average similarity score based on the work of the algorithm was equal to 0.6451. Each of the scores, ranging between 0.4191 and 0.7893, indicates that there is potential for adding more content to the model. Comparing all the chatlogs, the average similarity score was 0.5992; the "Task Assignment Tracking" category rises to the peak of completeness levels. The scores of task assignment tracking and description varied from a low of 0.2423 to a high of 0.8300, it can be considered rather satisfactory.

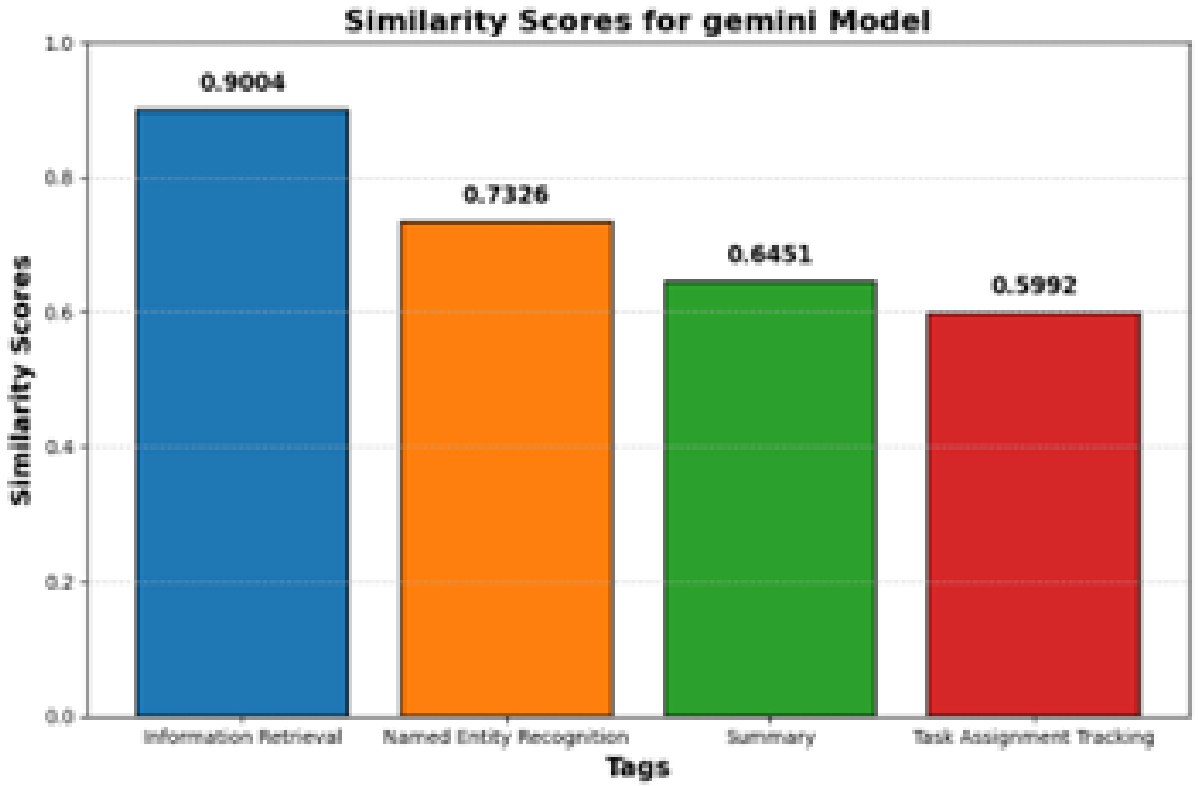

Figure 20: Gemini performance for different tasks for case 1 data

5.1.2 Case 2 Through the analysis of Case 2 for the Gemini model, we gained valuable insights into its performance across different project management and communication dimensions. The model's average processing time was found to be **5.10 seconds**, serving as a baseline for evaluating its effectiveness. The similarity score for handling communication failures was assigned a score of 0.5427. The model achieved a similarity score of 0.5449 for timeframes that were in conflict with each other. Gemini showcased impressive expertise in handling human error, evident from its outstanding similarity score of 0.6628.

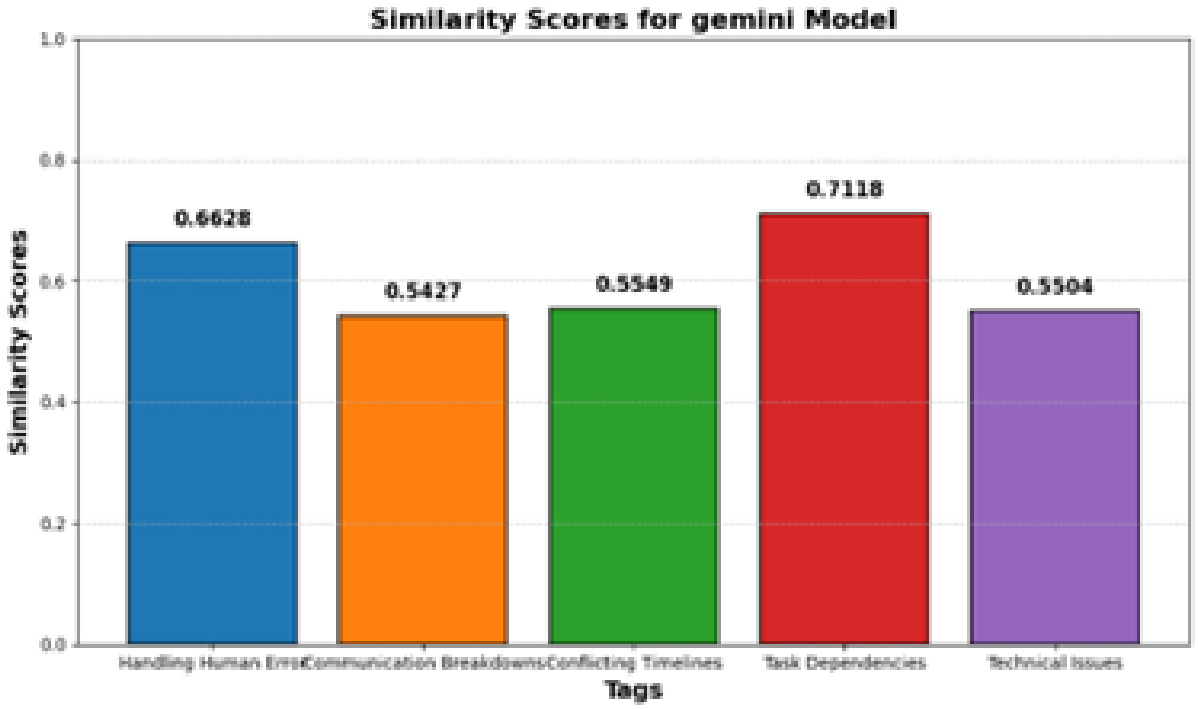

Figure 21: Gemini performance for the case 2

5.1.3 Case 3 Despite the LLM's success in maintaining privacy, there were some shortcomings in the accuracy and detail of the responses. As questions grew less specific, the LLM's responses occasionally lacked the detailed insights that could have been derived from the chat logs. Notably, there were gaps in information regarding specific actions taken to address challenges and the nature of those challenges. This issue highlights a common challenge in generalization, where responses, while privacy-compliant, sometimes lacked the depth needed to fully address the complexities of the queries. Overall, the LLM effectively balanced the need for confidentiality with the provision of relevant information. The consistent maintenance of privacy, despite leading to some generalizations, underscores the LLM's ability to handle complex and abstract questions while safeguarding sensitive information. The study concludes that the LLM demonstrated a strong commitment to privacy, successfully navigating the trade-off between confidentiality and response accuracy.

5.2 Mistral

5.2.1 *Case 1*

The similarity score for Mistral was found to be 0.7502 for the category of 'Information Retrieval' tag. The scores varied from a low of 0. 5665 to 0. 8929 which points to a relatively skewed rightwards. The model was most effective in terms of the acquisition and retrieval of specific knowledge. For the "Named Entity Recognition" category, the similarity score ranges from 0.4161 to 0.7243. Consequently, the average similarity in terms of the "Summary" tag was relatively low and attained an average value in the Mistral model of 0.7130.

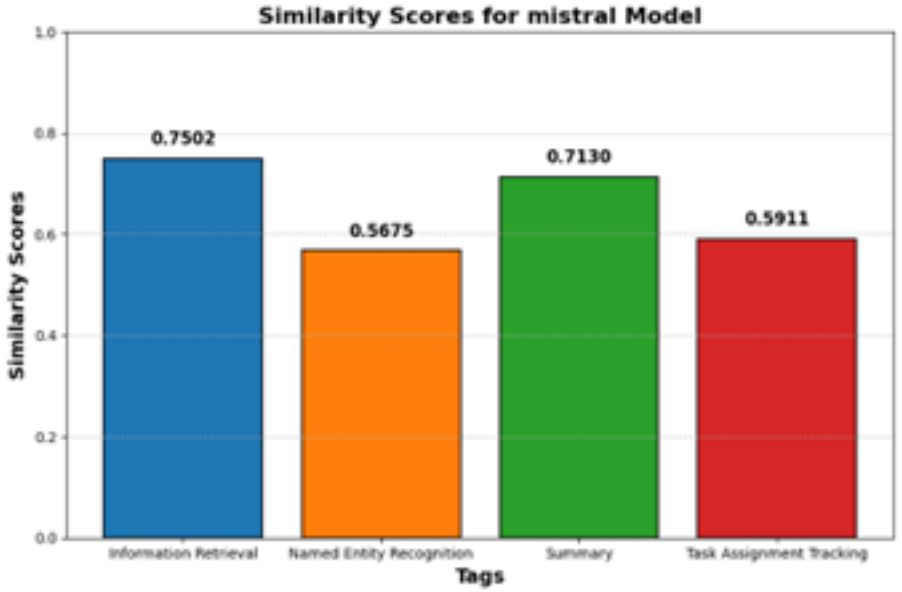

Figure 22: Mistral performance for case 1

5.2.2 *Case 2*

The average response time was found to be 5.70 seconds for the Mistral for this particular case, offering valuable insights. For the "Communication Breakdowns" tag, the Mistral achieved an average similarity score of 0.5761 also the scores indicate a range of performance levels in managing communication concerns, with values ranging from 0.5461 to 0.6062. The model achieved a similarity score of 0.6592 in the "Conflicting Timelines" category, indicating its performance was average. The scores displayed a consistently strong performance, ranging from 0.6384 to 0.6840.

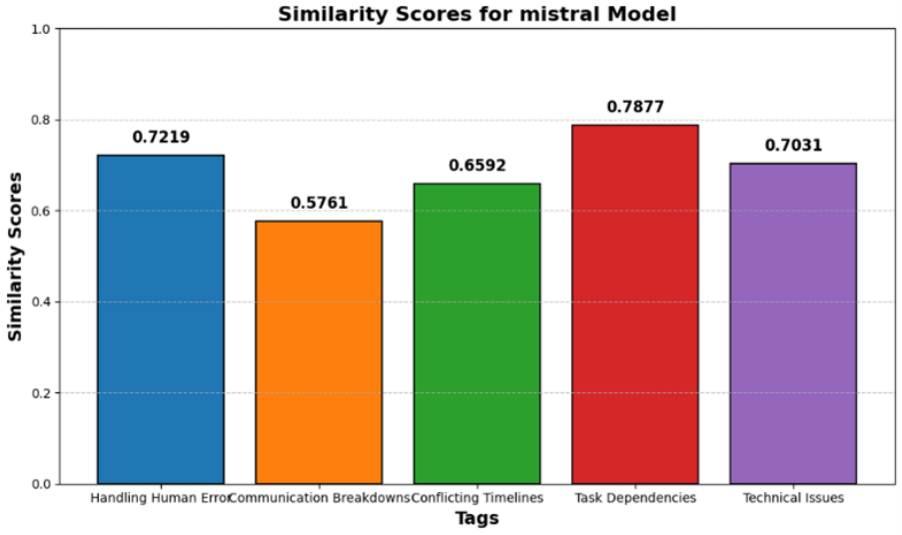

Figure 23: Mistral performance for case 2

Whereas for "Handling Human Error" category, the model achieved a similarity score of

0.7219, with values ranging from 0.6209 to 0.8411. The model effectively explained how communication breakdowns impacted task prioritization and dependency management. In the category of "Task Dependencies," the Mistral model achieved an average similarity score of 0.7877. The scores throughout the performance ranged from 0.7497 to 0.8446, indicating a strong ability to manage and communicate work dependencies.

5.2.3 *5.2.3 Case 3*

The analysis performed in case 3 involved the comparison of Mistral's answers to privacy requirements and probe points that were Accuracy, Response quality, and the trade-off between privacy and information disclosure. In Case 3, Mistral was very cautious precisely about protecting individuals' identities. This was done in a very foresighted and wise way by not disclosing internal workings of the office and individual efforts hence ensuring a strict confidentiality even in the most intricate of situations. Considering the potential complaints and their potential solutions within the conversation logs, it is objective in summing up the core problems and possible solutions. Besides, it offered specific information as to the boost of the backend's operations and solving problems at the present time.

5.3 T5 Flan Large

5.3.1 *Case 1*

The average similarity score for information retrieval was calculated to be 0.6176. Although the model achieved a perfect match in certain scenarios, such as identifying the new project name, it struggled to answer questions that need precise developer knowledge, resulting in a low similarity score of 0.1492. In the field of named entity recognition, the model achieved an average similarity score of 0.3284. While the system demonstrated proficiency in identifying specific items, such as Developer 3's assignment, it often struggled to acknowledge crucial information, as evidenced by the notably low similarity score of 0.0813 for task statuses. The tasks that were analyzed for summary had an average similarity score of 0.3535.

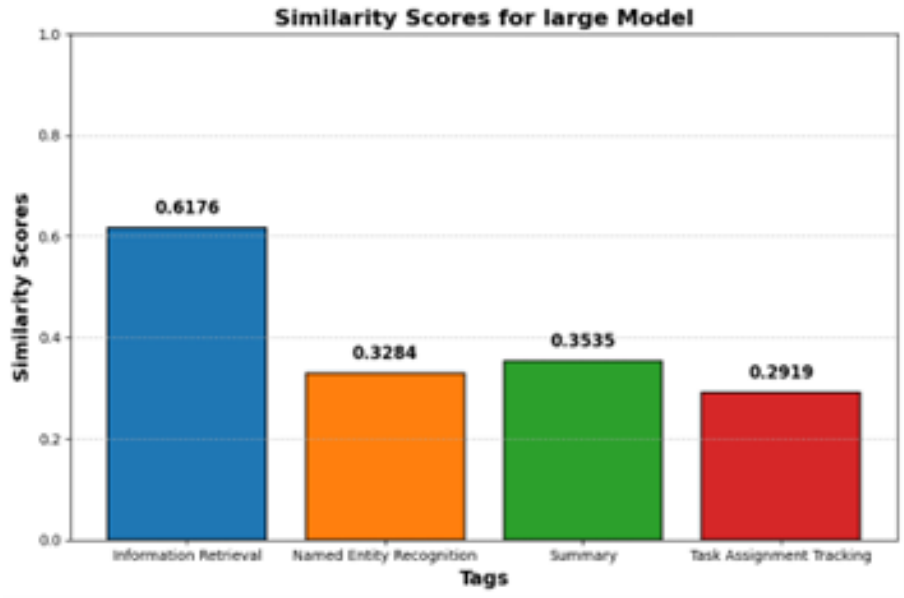

Figure 24: T5 performance in case 1

Although the model performed reasonably, its capacity to capture essential details in summaries was restricted. The task assignment tracking findings indicate that the average similarity score was 0.2919.

5.3.2 *Case 2*

The average similarity score for the category "Task Dependencies" was 0.3016. The mean similarity score for the "Technical Issues" category was 0.2994. The omission of critical details and reliance on generic information are clear signs that there is a need for improvement in managing complex project scenarios.

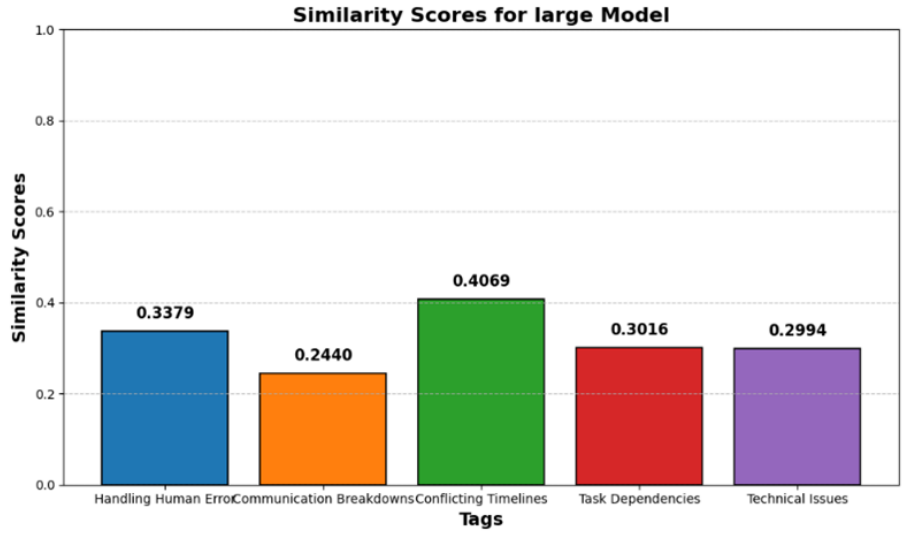

Figure 25: T5 performance for case 2

5.3.3 Case 3

Google T5 often gave answers that were missing pieces because it couldn't keep the right context across multiple questions. On other occasions, the answers were discontinuous and disjointed, which diminished the depth of the data provided. In these instances, the loss of context was obvious.

As a matter of fact, the replies were most of the time short and omitted all the other minor details asked. Essential information, as well as insufficient context, results in regression of the response quality on the whole; thus, a weak description contributes to this process as well. The given answers may be considered vague or insufficient; moreover, several of them did not answer the questions comprehensively. Google T5 managed to keep private information private, but the answers it gave were frequently less helpful because they lacked detail and context. The model's performance issues disrupted the balance between sharing useful information and protecting privacy.

5.4 T5 Flan Base

5.4.1 *Case 1*

The similarity scores with the computed average for the model were found to be 0.5576. The model's results in the context of named entity recognition were also found to be lower, with an average similarity equal to 0.4541. The model obtained an average of the two scores for similarity, which was 0.3095, indicating that the method's capacity to summarize information was comparably low. As indicated by the average similarity score of 0.392, the category of task assignment monitoring exposed the model's weaknesses.

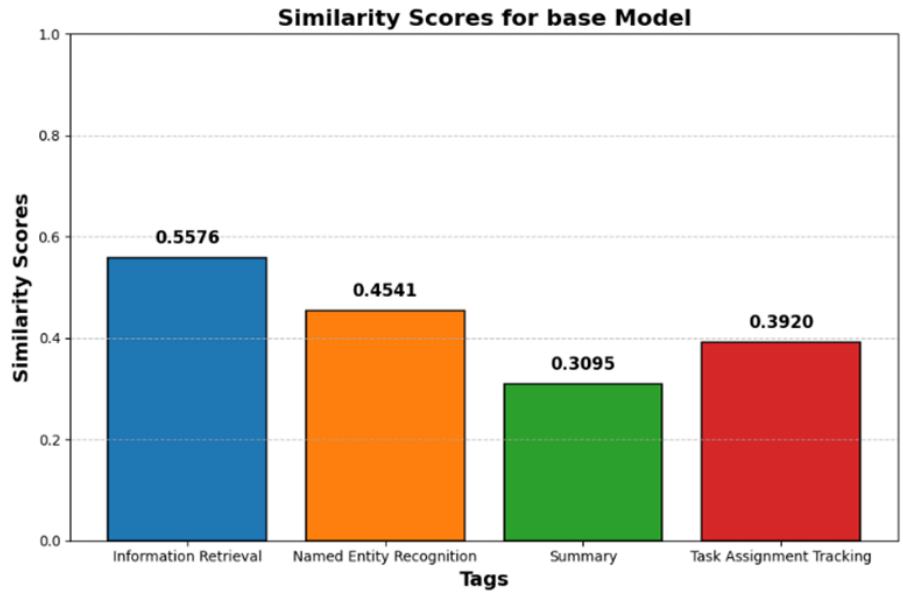

Figure 26: T5 Flan Base performance for case 1

5.4.2 *Case 2*

The similarity score of 0.3561, which was the highest score in this category, was nonetheless seen as quite low. The model's performance in the Conflicting Timelines category

was consistently low, with an average score of 0.2575 throughout the experiment. The highest similarity score for this tag was 0.4900, which suggests occasional partial accuracy. Among the several tags, the tag with the highest average score was "Handling Human Error," which scored 0.4233.

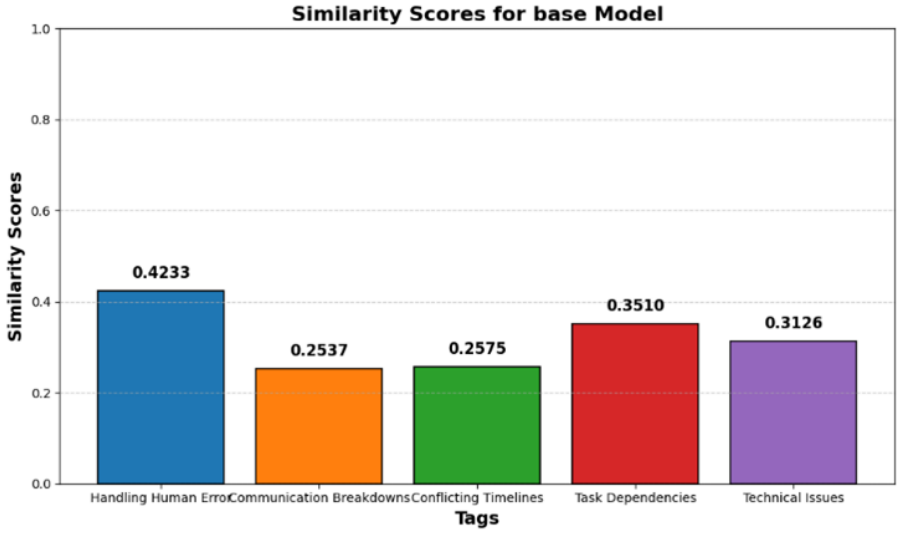

Figure 27: T5 Flan Base performance for case 2

An additional demonstration of the constraints of the model is provided by the examination of the tags for Task Dependencies and Technical Issues. Despite receiving average scores of 0.3510 and 0.3126, respectively, the responses frequently failed to appropriately identify and comment on significant dependencies and technical issues.

5.4.3 *Case 3*

Oftentimes, the model's output did not align with the expected responses, indicating issues with both relevance and level of information. For instance, the model consistently generated outcomes that were either unrelated or excessively basic, and it was unable to offer significant insights into the complex project inquiries that were posed. The tendency to provide ambiguous responses hinders the model's ability to provide actionable information in the setting of project management, where precise and detailed responses are necessary for effective decision-making.

5.5 Discussion

Gemini's score of 0.9004 in the **Information Retrieval** category is much greater than Mistral's score of 0.7502, T5-Large's score of 0.6176, and T5-Base's score of 0.5576. Gemini clearly outperforms the other models in terms of its capacity to retrieve relevant information due to a significant disparity between Gemini and the other models.

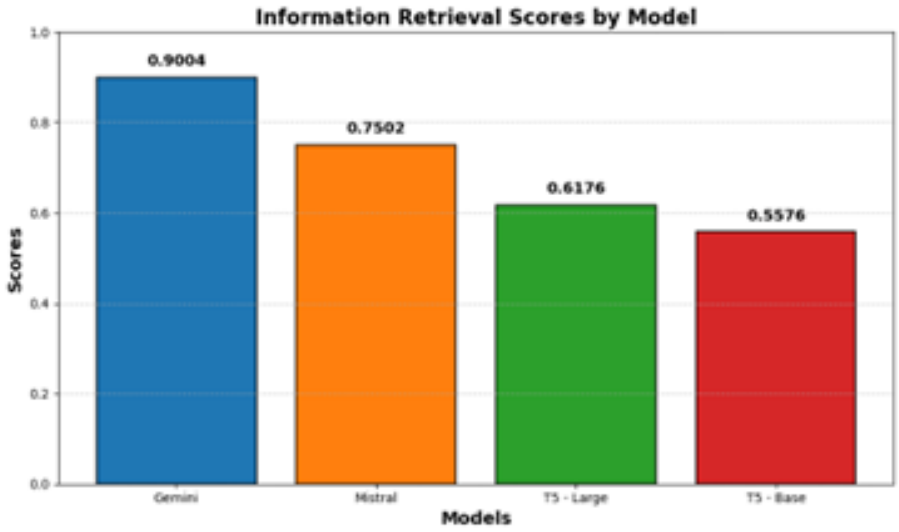

Figure 28: Performance Comparison for all the models for the Information Retrieval

Gemini outperforms all other models in **Named Entity Recognition**, achieving a score of 0.7326. Mistral follows closely in second place with a score of 0.5675. T5 - Large and T5 - Base have achieved scores of 0.3284 and 0.4541, respectively, which positions them lower in the rankings. Gemini and Mistral have a notable disparity (0.1651), indicating that Gemini possesses a more robust ability to identify named entities.

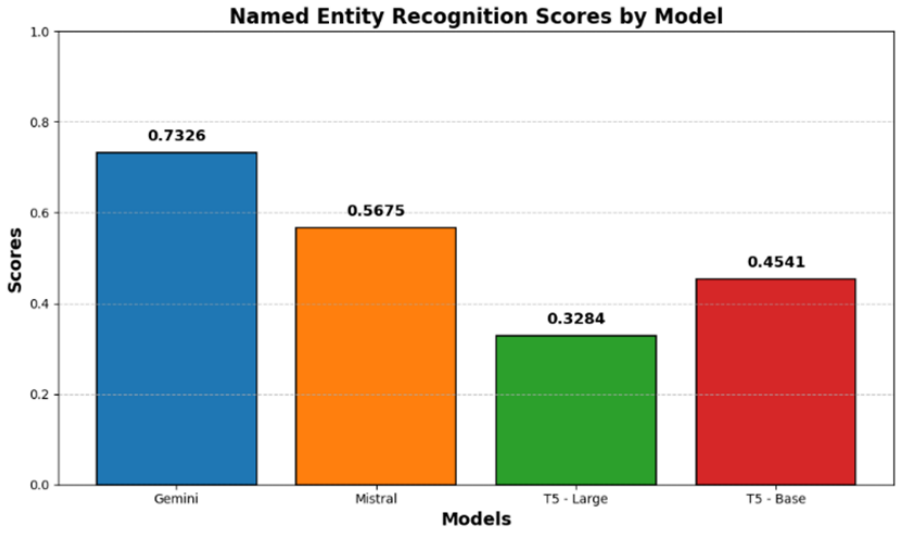

Figure 29: Performance Comparison for all the models for the NER

In the **Summary** category, Mistral achieved a score of 0.7130, surpassing Gemini's score of 0.6451. Although the disparity between Mistral and Gemini (0.0679) is small, it is nonetheless meaningful, implying that Mistral may possess a minor edge in literary summarization. The performance of T5 Large (0.3535) and T5-Base (0.3095) are significantly lower, which suggests that these models may have difficulty producing summaries that are brief. The efficiency of Mistral is shown in the disparity between the Mistral model and the T5 model while summarizing tasks.

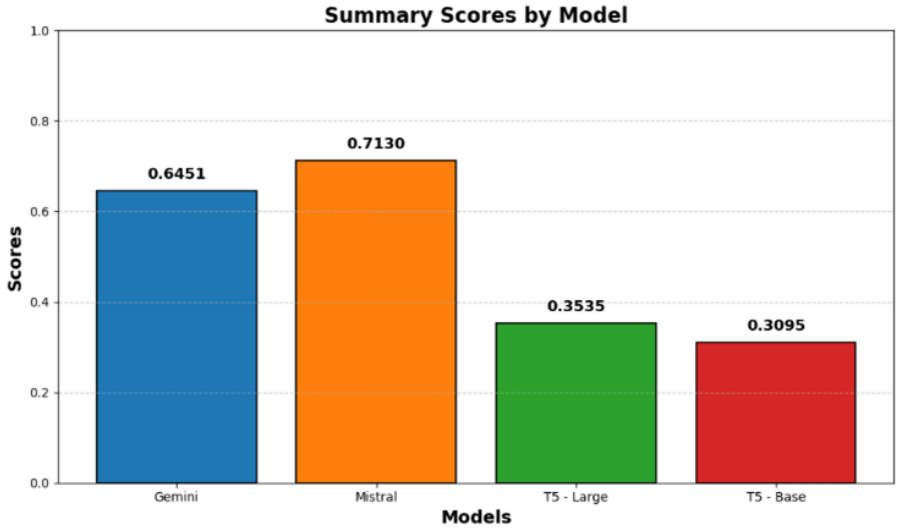

Figure 30: Performance Comparison for all the models for the Summary Scores

Gemini received a score of 0.5992 **for Task Assignment Tracking**, which is slightly higher than the score of 0.5911 received by Mistral. Given the negligible discrepancy of 0.0081 between the two models, it can be inferred that they exhibit comparable performance in this particular category.

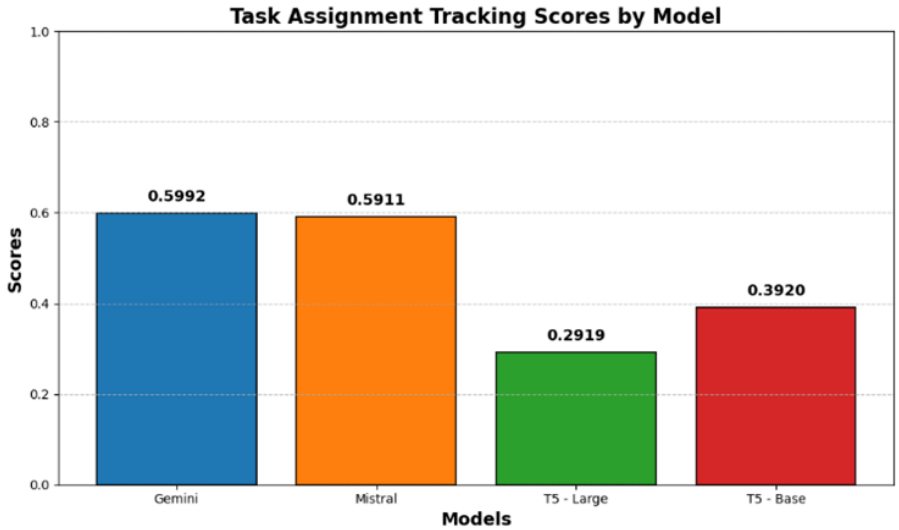

Figure 31: Performance Comparison for all the models for the Task Assignment

There is a likelihood that T5-Large (0.2919) and T5-Base (0.3920) have inferior ratings, suggesting that they may not be capable of effectively performing task assignment tracking. With a score of 0.7219, Mistral is in the lead in the category of **Handling Human Error**, surpassing Gemini's score of 0.6628. The difference, which is 0.0591, is not significant, which suggests that Mistral is slightly more adept at dealing with human errors. The lower scores for T5-Large (0.3379) and T5-Base (0.4233) indicate that these models may not be as effective at addressing human error as other models currently available.

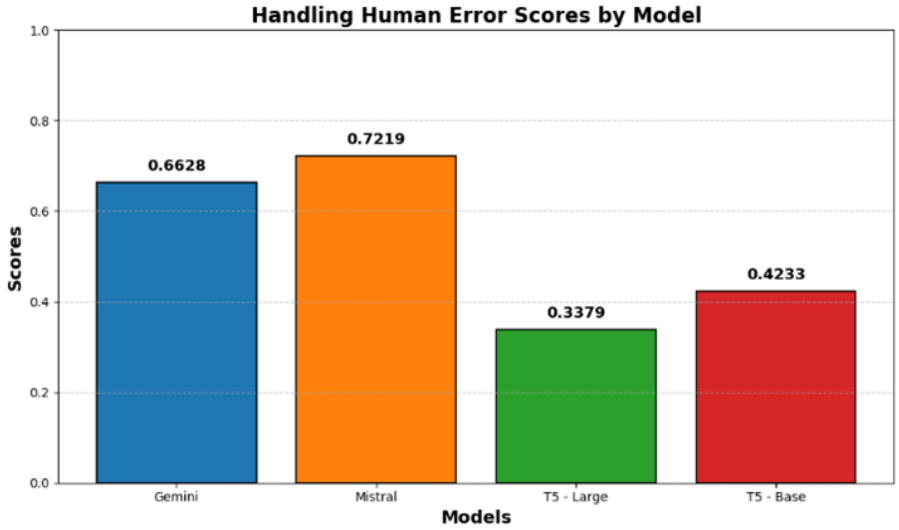

Figure 32: Performance Comparison for all the models for handling human error scores

Mistral's score of 0.5761 is significantly greater than Gemini's score of 0.5427. Mistral may be able to withstand **communication breakdowns** somewhat better than other technologies, as indicated by the difference (0.0334), which is significant but not excessively large. The performance disparity between T5-Large (0.2440) and T5-Base (0.2537) is significant, indicating potential limits in managing communication concerns for these two T5 variants.

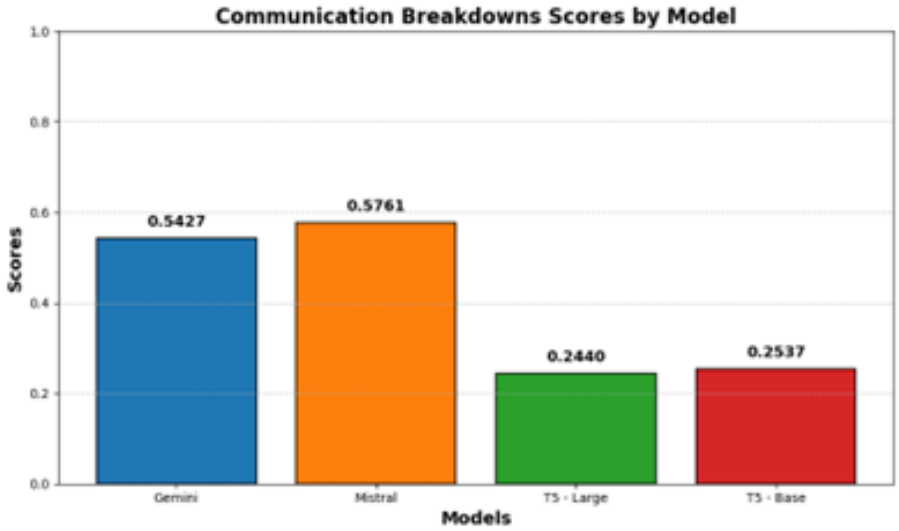

Figure 33: Performance Comparison for all the models for Communication breakdown Scores

When it comes to managing **conflicting timelines**, Mistral has the highest score of 0.6592, followed by Gemini with a score of 0.5549. Given that the difference (0.1043) is statistically significant, it may be inferred that Mistral is more effective in this case. Both T5-Large (0.4069) and T5-Base (0.2575) have lower scores, which suggests that they may have difficulty managing their timelines.

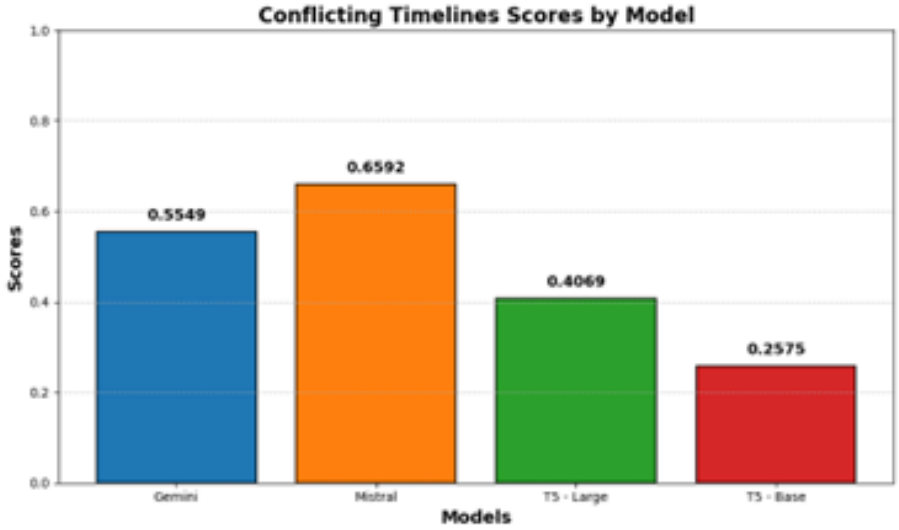

Figure 34: Performance Comparison for all the models for the Conflicting Timelines Scores

The management of **task dependencies** is an area in which Mistral excels, earning a score of 0.7877, whereas Gemini receives a score of 0.7118. In light of the fact that the difference (0.0759) is not only moderate but also significant, it is clear that Mistral is superior in terms of dependencies.

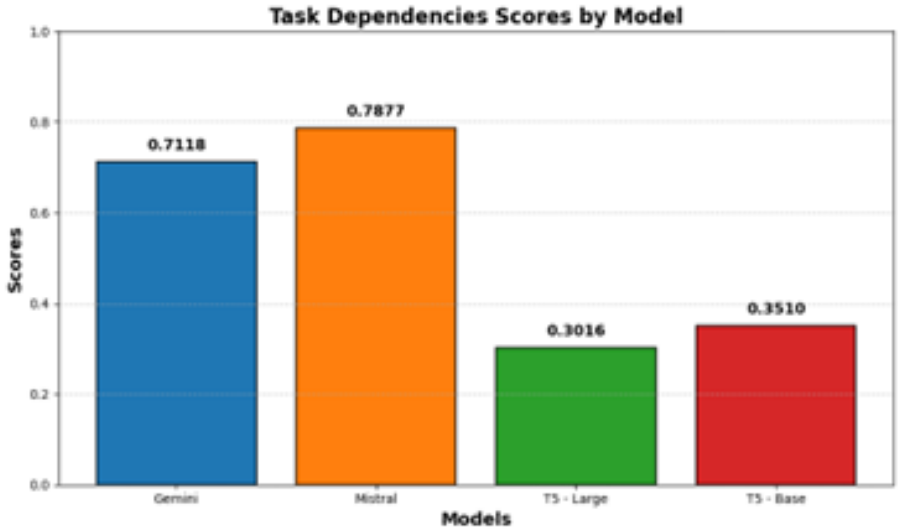

Figure 35: Performance Comparison for all the models for the Summary Scores

As far as addressing **technical issues** is concerned, Mistral is at the top with a score of 0.7031, while Gemini has a score of 0.5504. There is a significant difference (0.1527) between the two, which suggests that Mistral is preferable when it comes to addressing technological issues.

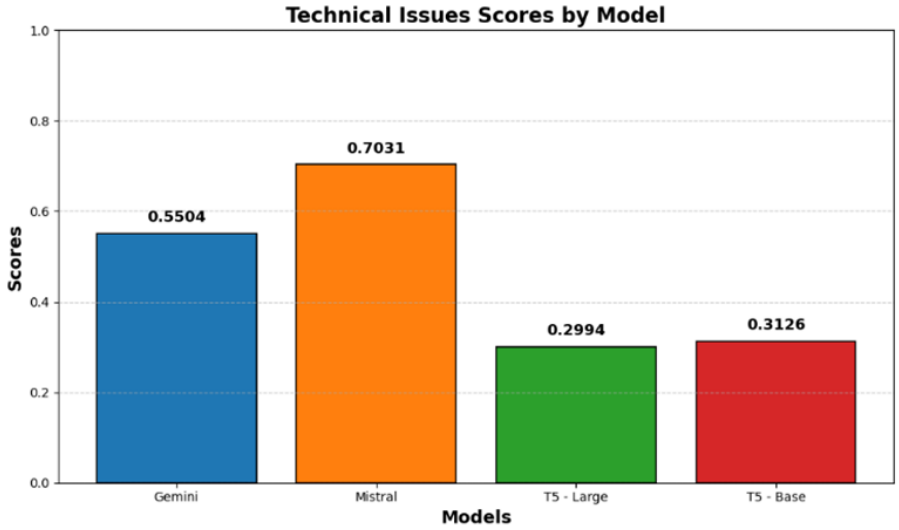

Figure 36: Performance Comparison for all the models for the Technical Issues Scores

This demonstrates that T5-Large (0.2994) and T5-Base (0.3126) have lower scores, which highlights the limits they have in terms of handling technical problems. Due to the huge difference that exists between the Mistral and T5 models, Mistral excels in this particular domain.

6 Conclusion and Future Work

A comprehensive assessment of a large number of "LMs" labeled as Google T5, Gemini, and Mistral was carried out in various tasks. The study in this paper revealed several trends in the evaluation performance patterns and gave clear indications of what could be developed. Google T5 **had significant difficulties in summarizing the dense and complex conversation** and produced either **incomplete or fragmented answers**. Consequently, **it showed that the model's inability to track context results in a massive loss of information**, especially when it involves **dialogues that require deftness in analyzing arguments**. On the other hand, the Gemini and Mistral **models performed well in tasks, surpassing T5's ability to deal with a wide range of questions and focus on a range of topics** much **more relevant to the current context**.

To give an overview, the work presented in this research is focused on the need **to enhance the Google T5 model's ability to retain contextual information** and **develop methods for generating concise summaries**. Thus, in order to manage **voluminous and hardly balanced conversations, it might be necessary to improve the structure of the model** or **add sophisticated training techniques that focus on context-saving**. To improve the performance of the network in tasks related to summarization, it is possible to apply several strategies that contain better context-aware embeddings or the application of hierarchical attention processes.

However, robustness can be improved **by implementing domain-specific fine-tuning, expanding the training sets with a greater number of cases of concern, or using such techniques as transfer learning or adversarial training**. Moreover, **compar-**

ing the effectiveness of the models in real-life scenarios and incorporating the users' feedback into the training process may yield useful observations that could be used in subsequent improvements. Instead, the key goal of all current innovative activities should be aimed at eradicating these limitations and improving the accuracy, context-sensitivity, and usefulness of the models.

References

Belzner, L., Gabor, T. and Wirsing, M. (2023). Large language model assisted software engineering: prospects, challenges, and a case study, *International Conference on Bridging the Gap between AI and Reality*, Springer, pp. 355–374.

Cabrero-Daniel, B., Herda, T., Pichler, V. and Eder, M. (2024). Exploring human-ai collaboration in agile: Customised llm meeting assistants, *International Conference on Agile Software Development*, Springer Nature Switzerland Cham, pp. 163–178.

Daniella, G., Wijaya, N., Putra, N. G. E., Efata, R. et al. (2023). Agile software development effort estimation based on product backlog items, *Procedia Computer Science* **227**: 186–193.

Dhruva, G., Shettigar, I., Parthasarthy, S. and Sapna, V. (2024). Agile project management using large language models, *2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT)*, IEEE, pp. 1–6.

Feng, P., He, Y., Huang, G., Lin, Y., Zhang, H., Zhang, Y. and Li, H. (2024). Agile: A novel framework of llm agents, *arXiv preprint arXiv:2405.14751* .

Fowler, M. and Highsmith, J. (2001). The agile manifesto, *Software Development* **9**(8): 28–35.

He, J., Treude, C. and Lo, D. (2024). Llm-based multi-agent systems for software engineering: Vision and the road ahead, *arXiv preprint arXiv:2404.04834* .

Jiang, A., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D., Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L. and Lavaud, L. (2023). Mistral 7b, *arXiv preprint arXiv:2310.06825* .

Kim, Y., Park, C., Jeong, H., Chan, Y., Xu, X., McDuff, D., Breazeal, C. and Park, H. (2024). Adaptive collaboration strategy for llms in medical decision making, *arXiv preprint arXiv:2404.15155* .

Kumar, G. and Bhatia, P. (2012). Impact of agile methodology on software development process, *International Journal of Computer Technology and Electronics Engineering (IJCTEE)* **2**(4): 46–50.

Lakkaraju, K., Jones, S., Vuruma, S., Pallagani, V., Muppasani, B. and Srivastava, B. (2023). Llms for financial advisement: A fairness and efficacy study in personal decision making, *Proceedings of the Fourth ACM International Conference on AI in Finance*, ACM, pp. 100–107.

Nasiri, S. and Lahmer, M. (2024). Ai-driven methodology for refining and clustering agile requirements, *2024 4th International Conference on Innovative Research in Applied Science, Engineering, and Technology (IRASET)*, IEEE, pp. 1–7.

Niessl, M., Gruber, C. and Eder, M. (2023). Restarting scaled agile development at austrian post. experience report, *24th International Conference on Agile Software Development*.

Nikolaev, D. and Padó, S. (2023). Representation biases in sentence transformers, *arXiv preprint arXiv:2301.13039* .

Roberts, A., Chung, H., Mishra, G., Levskaya, A., Bradbury, J., Andor, D., Narang, S., Lester, B., Gaffney, C., Mohiuddin, A. and Hawthorne, C. (2023). Scaling up models and data with t5x and seqio, *Journal of Machine Learning Research* **24**(377): 1–8.

Sami, M., Rasheed, Z., Waseem, M., Zhang, Z., Herda, T. and Abrahamsson, P. (2024). Prioritizing software requirements using large language models, *arXiv preprint arXiv:2405.01564* .

Shen, Y., Song, K., Tan, X., Li, D., Lu, W. and Zhuang, Y. (2024). Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, *Advances in Neural Information Processing Systems* **36**.

Srivastava, A., Bhardwaj, S. and Saraswat, S. (2017). Scrum model for agile methodology, *2017 International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, pp. 864–869.

Sun, P. and Shao, Q. (2023). Mapm: Multi-modal auto-agile project risk management and prediction for collaboration platform using ai/ml.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J., Yu, J., Soricut, R., Schalkwyk, J., Dai, A., Hauth, A. and Millican, K. (2023). Gemini: a family of highly capable multimodal models, *arXiv preprint arXiv:2312.11805* .

Tikayat Ray, A., Cole, B., Pinon Fischer, O., Bhat, A., White, R. and Mavris, D. (2023). Agile methodology for the standardization of engineering requirements using large language models, *Systems* **11**(7): 352.

Yang, J., Korecki, M., Dailisan, D., Hausladen, C. and Helbing, D. (2024). Llm voting: Human choices and ai collective decision making, *arXiv preprint arXiv:2402.01766* .

Zhang, Z., Rayhan, M., Herda, T., Goisauf, M. and Abrahamsson, P. (2024). Llm-based agents for automating the enhancement of user story quality: An early report, *International Conference on Agile Software Development*, Springer Nature Switzerland, Cham, pp. 117–126.