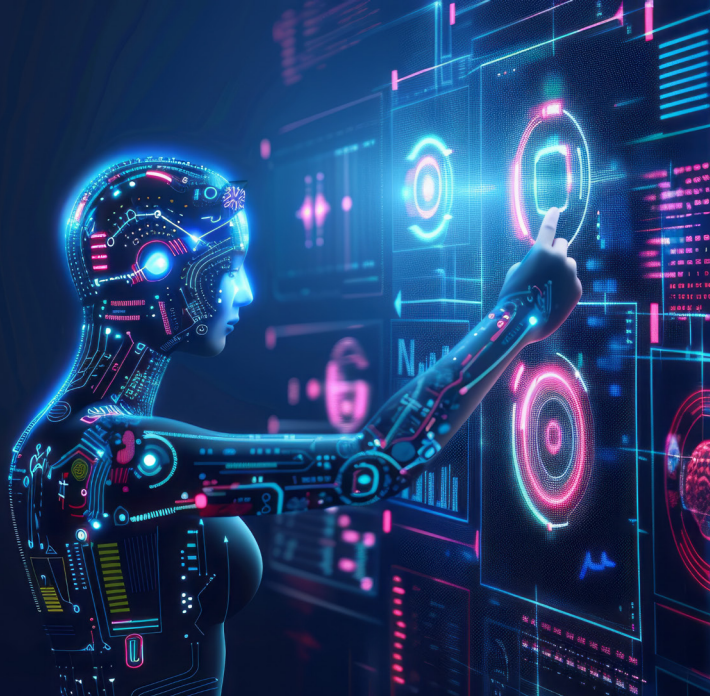
EMERGYS
An ACCSCIENT Company



Generative AI-Driven System Engineering Co-Pilot Platform for Efficient Extraction of Projects Requirements and Objectives Management

Rosemary Kioko[1], Jake Kibunja[1], Dylan Reed[1], Rohit Mittal[2], Saumitra Modak[3], Gaurav Singh[3], Hitesh Jha[3], Veeresh Dharappanavar[3], and Pranav P. Shah[3]

1. System Strategies & Analysis (SSA) Inc, MD-USA
2. Drakestone Technologies LLC, CO-USA
3. Emergys LLC company, Pune-India

KEYWORDS

Generative AI, GenAI, Large Language Models, LLM, Retrieval Augmented Generation, RAG, OpenAI, GPT-3.5, GPT-4, GPT-4 Turbo, Vectorstore, FAISS, Embeddings, OpenAI Embeddings, and Hugging Face Embeddings.

ABSTRACT

SSA Inc is a woman owned Engineering company that specializes in Space Systems Engineering, Spacecraft Mission Engineering & Operation s. Model Based System Engineering (MBSE), MBSE/MBE training, Information Technology solutions, and Program/ Project Management. SSA has developed a Generative Artificial Intelligence (Generative AI) based System Engineering Co-Pilot (SE Co-Pilot) platform to extract, organize and analyze digital engineering information from engineering documents.

The goal of developing the SE Co-Pilot platform was to address and mitigate the substantial upfront costs that are often encountered in System Engineering projects. This platform is designed to enhance the effectiveness and efficiency during the requirement development process. By streamlining this crucial phase, the SE Co-Pilot platform aims to significantly reduce the need for rework in the later project stages, thus saving time and resources while ensuring adherence to project timelines, eventually leading to more successful and cost-effective project outcomes.

The SE Co-Pilot platform utilizes Large language Models (LLM) models to streamline the extraction of complex system objectives e.g Space Missions, requirements, and other key details from systems engineering documents developed by Defense, Space, Aviation, Aerospace, Commercial, and other Federal & Defence agencies.

The SE Co-Pilot platform converts PDF documents into Markdown files to facilitate easier handling. The markdown files are processed using GPT-3.5, which divides the content into manageable chunks for initial analysis. The responses are refined through a cleaning phase to remove irrelevant information. Manual extraction ensured a high level of precision in capturing the mission's intricate objectives & requirements.

This SE Co-Pilot platform demonstrates the integration of automated Generative AI-driven techniques combined with manual oversight, achieving a high degree of accuracy and efficiency in handling large volumes of diverse documents. Developed by SSA, the platform provides a robust foundation for future applications in data-intensive environments.

Table of Contents

Keywords 1
1. INTRODUCTION 3
2. PROBLEM STATEMENT 3
3. USE CASE 1 : CATEGORIZING THE REQUIREMENTS 5
3.1.1. Objectives 5
3.1.2. Solution Methodology 5
3.1.2.1. PDF Conversion 5
3.1.2.2. Chunking Data 6
3.1.2.3. Data Cleansing 6
3.1.2.4. Manual Extraction 7
3.1.2.5. Merge Extracted Data 7
3.1.2.6. Data Preparation for Retrieval-Augmented Generation (RAG) 7
3.1.2.7. Data Embedding 8
3.1.2.8. vector store setup 8
3.1.2.9. Retrieval and Inference with GPT-4 Turbo 9
3.1.2.10. User Interface 10
3.2. USE CASE 2: GENERATING THE REQUIREMENTS 11
3.2.1. Objective 11
3.2.2. Solution Methodology 11
3.3. USE CASE 3: DETECTING similar REQUIREMENTS 11
3.3.1. Objectives 11
3.3.2. Solution Methodology 12
4. Conclusion 15
5. Future SE Co-Pilot Roadmap 15

1. INTRODUCTION

In today's data-driven world, the ability to efficiently extract and organize critical information from large volumes of documents is essential for effective decision-making and operational success. This white paper presents a Generative AI-based platform, that SSA, has designed to streamline the extraction of mission-critical information from numerous PDF documents related to Space Mission Systems Engineering requirement. By leveraging advanced Generative AI techniques, particularly OpenAI's GPT models, this SE Co-Pilot platform ensures accurate and efficient data extraction, making it invaluable for complex and data intensive Systems .

The SE-Co-Pilot platform transforms unstructured systems engineering data within PDF documents into structured, accessible information, including mission objectives, requirements, and other key details. This transformation is crucial for Defense, Space, Aviation, Aerospace, Commercial, and other Federal & Defence agencies like National Aeronautics and Space Administration (NASA), Department of Defense (DoD), where the ability to quickly retrieve and analyze mission-specific information can significantly impact the success of space missions and other critical operations.

The SE Co-Pilot platform incorporates several stages, beginning with the conversion of PDF documents into a more manageable format, followed by initial processing using GPT-3.5, and culminating the use of GPT-4 Turbo for advanced retrieval and inference. By integrating both automated Generative AI-driven processes and manual extraction efforts, this approach ensures a high degree of accuracy and comprehensiveness in the extracted data.

This document details each step of the SE Co-Pilot platform, demonstrating how AI can be harnessed to handle large datasets efficiently. The process not only simplifies data extraction but also enhances the ability to search and retrieve information, eventually improving the management and utilization of extensive document collections.

2. PROBLEM STATEMENT

Handling large volumes of unstructured data, such as PDF documents containing mission objectives and requirements for large complex systems engineering, presents significant challenges. Traditional methods of data extraction and organization are often time-consuming, labor-intensive, and inclined to errors. For organizations engaged in complex space operations, such as NASA, the ability to quickly and accurately extract relevant information from documents is critical.

The primary challenges include:

2.1. Difficulty in Converting Unstructured Data into a Structured Format:

- **Details**: PDF documents often contain a mix of text, images, and complex formatting that make it challenging to extract information in a structured manner.
- **Example**: Extracting mission requirements from a PDF document with tables, embedded images, and varied text formatting can result in loss of context and accuracy if not handled properly.

2.2. Inefficiencies in Manually Extracting and Organizing Key Details from Documents:

- **Details**: Manually processing large volumes of documents is time-consuming and prone to human error. This process often requires significant effort to identify and extract relevant information.
- **Example**: Analyzing hundreds of pages of mission objectives and requirements manually can lead to inconsistent extraction of critical data, resulting in gaps and inaccuracies.

2.3. Challenges in Maintaining Context and Continuity Across Large Documents:

- **Details**: Large documents often contain information spread across multiple sections and pages, making it difficult to maintain context and continuity when extracting data.

- **Example**: A requirement mentioned in one section might reference another section several pages away. Manually tracking these references and ensuring coherence is a significant challenge.

2.4. Limited Capabilities in Quickly Retrieving Specific Information from Vast Datasets:

- **Details**: Even after extracting and organizing data, retrieving specific information quickly from vast datasets remains a challenge without advanced search and retrieval mechanisms.
- **Example**: Quickly finding all instances of a particular mission objective or requirement in a dataset comprising thousands of pages can be inefficient and time-consuming using traditional search methods.

2.5. Inconsistency in Data Formats:

- **Details**: Different documents might follow varying formats and styles, making it difficult to apply a uniform data extraction process.
- **Example**: Documents from different projects or departments may use different terminologies, structures, and formatting conventions, complicating the extraction process.

2.6. Scalability Concerns:

- **Details**: As the volume of documents increases, the existing manual and semi-automated processes may struggle to scale effectively.
- **Example**: A growing repository of mission requirements and objectives requires more processing power and sophisticated techniques to maintain efficiency and accuracy.

2.7. Lack of Real-time Updates:

- **Details**: Manual processes cannot easily accommodate real-time updates or changes in the data.
- **Example**: If a new version of a document is released, manual updates are required to reflect changes, which can lead to delays and inconsistencies.

2.8. Difficulty in Handling Ambiguities and Variability:

- **Details**: Documents often contain ambiguous language and variability in how information is presented, which can be challenging for both manual and automated processes to interpret correctly.
- **Example**: Requirements might be stated in vague terms or use conditional language that is difficult to parse accurately without contextual understanding.

These challenges necessitate a robust solution that combines automated data extraction with manual oversight to ensure accuracy and efficiency.

3. USE CASE 1 : CATEGORIZING THE REQUIREMENTS

3.1.1. OBJECTIVES

Efficiently convert PDF documents related to Systems Engineering into structured, accessible formats e.g. Missions requirements documents.
Utilize Generative AI techniques to automate data extraction.
Integrate manual processes for nuanced information capture.
Enhance retrieval capabilities with advanced indexing and search methods.
Provide a user-friendly interface for interaction and query management.

3.1.2. SOLUTION METHODOLOGY

Fig 1 shows the Generative AI- based SE Co-Pilot platform process flow and the key components are describedbelow.

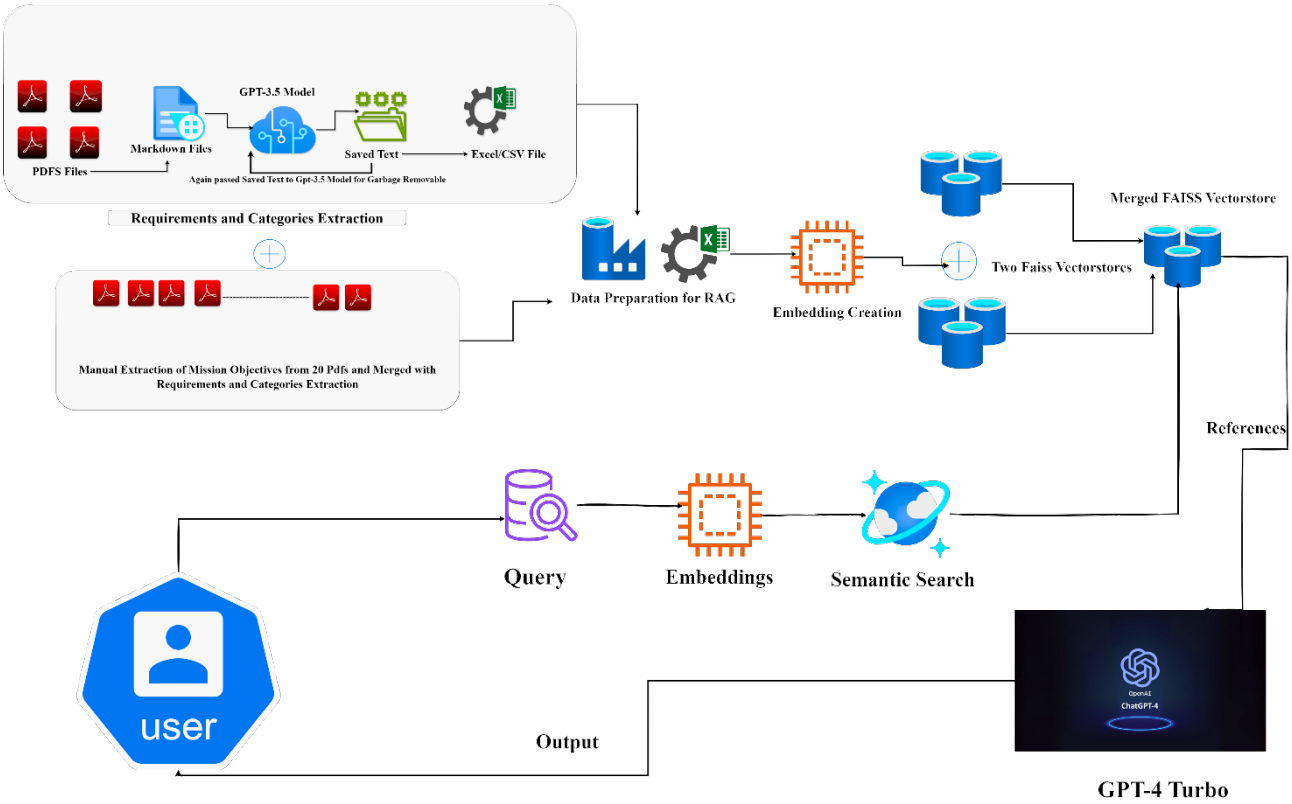


Figure 1: Process flow for Categorizing the Requirements

1. PDF Conversion: Transforming PDF documents into Markdown files for easier processing.
2. Chunking data: Dividing content into manageable chunks and obtaining preliminary insights.
3. Data Cleansing : Cleaning and refining the extracted data.
4. Manual Extraction: Ensuring accuracy by manually extracting mission-critical information.
5. Merge Extracted Data: Ensuring a comprehensive dataset containing all mission-critical information.
6. Data Preparation for Retrieval-Augmented Generation (RAG): Preparing data for advanced search and retrieval.
7. Data Embedding: Creating embeddings for advanced search capabilities.
8. Settiing up vectorstore: Building efficient search indexes.
9. Retrieval and Inference with GPT-4 Turbo: Providing sophisticated query handling and insights.
10. User Interface Implementing a simple, user-friendly UI for query interaction and history management.

3.1.2.1. PDF CONVERSION

The SE Co-Pilot platform begins with the conversion of PDF documents into Markdown files. This step is crucial as it transforms the data into a more accessible and manageable format, laying the groundwork for subsequent processing. This is particularly important for systems engineering documents, which often contain complex details and specifications.

Example: Suppose we have PDF documents from the NASA MAGIC Mission containing various mission details and requirements related to the mission payload. Each PDF is converted into Markdown format for easier processing.

```
'''in this cell we are converting our pdf content to markdown content,
and one markdown page content is treated as one chunk here'''

dict1 = {}
error_list2=[]
for bg in tqdm(range(0,len(all_pdf[0:1]))):
    chunks = []
    error_list=[]
    page_number = []
    pdf_name = []

    # filename = "pdfs/"+ str(all_pdf[bg])
    try:
        try:
            filename = all_pdf[bg]
            pdf = Pdf.open(filename)
            x = len(pdf.pages)

        # we have saved all the pages of the pdf as individual page chunk as in markdown format
        # in "original_list" list.
        original_list = [i for i in chunks]
```

Figure 2: Code Snippet for Converting PDF documents into Markdown files

3.1.2.2. CHUNKING DATA

The converted Markdown files are processed using GPT-3.5. The content is divided into chunks, each comprising three pages with an overlapping page to ensure context continuity. These chunks are sent to GPT-3.5, which processes the text and provides preliminary insights. The responses from GPT-3.5 are saved, forming the basis for the next stage. The cleaned text, which now contains extracted requirements and categories, is saved.

Example: The Markdown file from "Magic Mission Requirements.pdf" is divided into chunks of 3 pages with 1 overlap page and processed by GPT-3.5 to extract text and preliminary insights. The cleaned text file containing structured requirements and categories from "Magic Mission Requirements.pdf" is stored.

```
        #now converting 3 page as 1 chunk with one overlap page which will be passed to our llm
        #in 1st iteration
        lists_of_3 = [original_list[i:i+3] for i in range(0, len(original_list), 2)]


        #iterating through each element in "list_of_3" and calling our API fuinction above and saving
        #its responses in one single text,i.e in "category_chunks".
        category_chunks = ""
        for chunk in lists_of_3:
            category_chunks += generate_response(chunk,prompt)
```

Figure 3: Code Snippet for dividing the Markdown files into chunks of 3 pages with 1 overlap page

3.1.2.3. DATA CLEANSING

The saved responses are combined into a single text file, which is then passed back to GPT-3.5 for cleaning. This step involves removing any irrelevant or redundant information, resulting in a refined text file containing only the essential requirements and categories relevant to SE projects. This cleaned text is saved for further use.

Example: The merged text from "Magic Mission Requirements.pdf" is refined by GPT-3.5 to eliminate any unnecessary content.

```
            refined_chunks = generate_response(category_chunks, prompt_refined)
            dict1.setdefault(filename[9:-4],refined_chunks)# rename dict1
```

Figure 4: Code snippet for data cleansing

pdf_name	requirements
474-00448-01-05_JPSS-SRS-Vol-I-Part-5_0200E	category : "Preface"requirement : 'changes to this document shall be made by complete revision.
GO-RS-ESA-SY-0001	Category: "Mission Requirements"Requirement: 'The operational lifetime of the mission shall be
2020-civil-monitoring-performance-specification	category : "Verification of Signal in Space Integrity Standards"requirement : 'Civil monitoring shall
Joint Polar Satellite System-2-Req	category : "Preface"requirement : 'changes to this document shall be made by complete revision.
474-00448-01-20_JPSS-SRS-Vol-I-Part-20_0200E	category : "Preface"requirement : 'Once this document is approved, JPSS approved changes are h
474-00448-01-04_JPSS-SRS-Vol-I-Part-4_0200D	category : "Preface"requirement : 'changes to this document shall be made by complete revision.
SMOS_MRD_V5	category : "Requirements of the SMOS Mission"requirement : 'The mission shall also provide infc
474-00448-01-17_JPSS-SRS-Vol-I-Part-17_0200E	category : "Preface"requirement : 'changes to this document shall be made by complete revision.
Copy of jpssSystemreq	category : "JPSS Level 1 Requirements - J2 Follow-on JPSS-REQ-1001/470-00031, Version 2.1"requi
ICD-GPS-240D-ICD	category : "INTERFACE CONTROL DOCUMENT"requirement : 'THIS DOCUMENT SPECIFIES TECHNI(
474-00448-01-08_JPSS-SRS-Vol-I-Part-8_0200D	category : "Preface"requirement : 'Once this document is approved, JPSS approved changes are h
MRD_21Nov99	category : "Science and Mission Requirements"requirement : 'The operational lifetime of the mis:

Figure 5: CSV output after data cleansing

3.1.2.4. MANUAL EXTRACTION

While AI provides significant automation, certain aspects still require human intervention for accuracy. Mission objectives which are often nuanced and context specific in space systems, are manually extracted from the PDFs. This manual effort ensures that the critical mission objectives are accurately captured.

Example: From "Magic Mission Requirements.pdf", mission objectives like "Achieve precise orbit insertion" and "Conduct comprehensive atmospheric studies" are manually noted.

B	E	F	G	H
pdf_name	**Mission Objectives**			
01 - SD-TN-AI-1167_2_GG Mission Requirements Document	The goal of GG is to test the “Equivalence Prin			
ADM-Aeolus_MRD	MISSION OBJECTIVES The primary objective o			
CIMR-MRD-v5.0-20230211_(Issued)	CIMR Mission ObjectivesObjectives are split ir			
CO2M_MRD_v3.0_20201001_Issued	MISSION OBJECTIVESThe objective of the CO2			
Copernicus_CHIME_MRD_v3.0_Issued_21_01_2021	MISSION OBJECTIVES Scientific and technolog			
Copernicus_L-band_SAR_mission_ROSE-L_MRD_v2.0_issued				
Copernicus_LSTM_MRD_v3.0_Issued_20210514				
CRISTAL_MRD_v3_0_issued_20200529	Mission ObjectivesThe aim is reached by impl			
EarthCARE_MRD_v5	Mission Objectives The EarthCARE mission ha			

Figure 6: Snippet of CSV output for manually extracted Mission Objectives

3.1.2.5. MERGE EXTRACTED DATA

Merge Manual and Automated Data: The manually extracted mission objectives are then merged with the requirements and categories derived from the automated process. This integration ensures a comprehensive dataset that combines the precision of human judgment with the efficiency of Generative AI-driven extraction.

Example: The manually extracted objectives are merged with the cleaned requirements from "Magic Mission Requirements.pdf". Refer to image

3.1.2.6. DATA PREPARATION FOR RETRIEVAL-AUGMENTED GENERATION (RAG)

Data from multiple PDFs is prepared by creating CSV files in a specified format, listing the PDF name, mission objectives, and requirements. This ensures consistency and structure for subsequent processing.

The CSV was made with partial manual work and partially by extracting directly from the pdf. Refer below table for csv format.

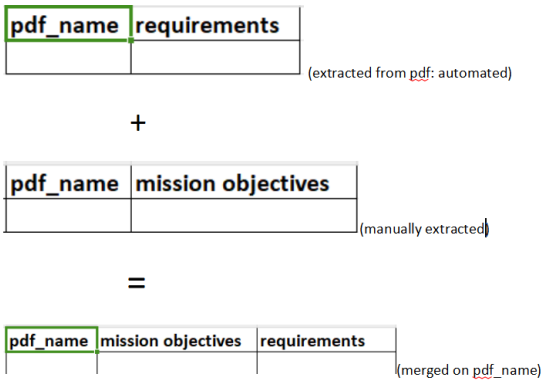


Figure 7: Process used for generating Final CSV for Requirements and Objectives

Here data under 'pdf_name' and 'requirements' columns were populated directly by extracting from the pdf and whereas data under 'misssion objectives' column was manually extracted along with the name of the pdf in a seperate csv file and later both the csv were merged on 'pdf_name' column

Example: Create a CSV file for each PDF with columns for the PDF name, mission objectives, and requirements.

Prepare Data for Additional PDFs: Similarly, prepare CSV files for another set of PDFs.

Example: Create CSV files for these additional PDFs in the same format.

	A	B	C	D	E	F	G	H
1	pdf_name	mission objectives	requirements					
2	GO-RS-ESA-	GOCE Mission Objectiv	Category: "Mission Requirements"Requirement: 'The operational lifetime of t					
3	FORUM_MI	MISSION OBJECTIVESTI	category : "Mission Requirements"requirement : 'The FORUM mission shall ha					
4	Sentinel-2_	MISSION OBJECTIVES G	category : "Definitions"requirement : 'The term "goal" denotes a non-mandat					
5	GMES_SEN	GMES Programme Obj	category : "System Requirements"requirement : 'The operational lifetime of t					
6	Copernicus	MISSION OBJECTIVES S	category : "Mission Requirements Document"requirement : 'The operational					
7	CO2M_MRI	MISSION OBJECTIVESTI	category : "MISSION REQUIREMENTS"requirement : 'The operational lifetime					
8	Manual-Lar	Mission ObjectivesThe	category : "4.1.3 Postlaunch"requirement : 'Radiometric characterization and					
9	CRISTAL_M	Mission ObjectivesThe	category : "Mission Requirements Document"requirement : 'The operational					
10	ADM-Aeolu	MISSION OBJECTIVES T	category : "OBSERVATION REQUIREMENTS"requirement : 'The operational life					
11	MAGIC_NG	Mission ObjectivesThe	category : "Mission Requirements"requirement : 'The operational lifetime of t					

Figure 8: Snippet of Final CSV with Requirements and Objectives.

3.1.2.7. DATA EMBEDDING

Embeddings are created for both datasets. These embeddings transform the textual data into a numerical format that captures the semantic meaning of the text, enabling advanced search and retrieval capabilities.

Example: Generate embeddings (OpenAI embeddings was used) for the text data in the CSV files of "Magic Mission Requirements.pdf" and other PDFs.

```
#open ai embeddings
embeddings = OpenAIEmbeddings(openai_api_key=openai_api_key)
```

3.1.2.8. VECTOR STORE SETUP

Separate FAISS Vectorstores are created for each set of embeddings. FAISS (Facebook AI Similarity Search) is a library that efficiently stores and searches through vector representations of data. These Vectorstores are then merged into a single comprehensive Vectorstore, creating a unified system for efficient information retrieval.

Example: Build separate FAISS Vectorstores for the embeddings from the two sets of PDFs and combine them into one comprehensive Vectorstore.

```
def calling_FAISS_OnePageOneChunk(docs, embeddings):
    vectorstore = FAISS.from_documents(documents=docs, embedding=embeddings)
    vectorstore.save_local('knowledge_base/'+ 'OA_2pdf_one_row_one_chunk')
```

```
'''calling above funtion'''

calling_FAISS_OnePageOneChunk(data, embeddings)
```

Figure 9: Code Snippet for Setting up Vectorstores

```
#merge two vectorstores
vectorstore_18.merge_from(vectorstore_2)
```

```
#save the merged vectorstores
vectorstore_18.save_local('knowledge_base/'+ 'OA_merge_one_row_one_chunk')
```

Figure 10: Code snippets for Merging two Vectorstores.

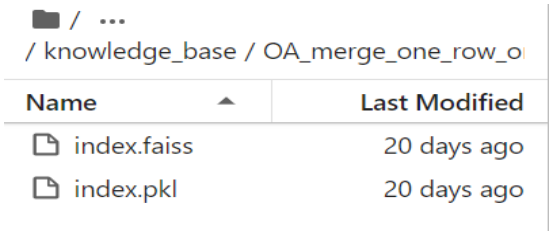


Figure 11: Snippet of saved merged Vectorstores for Usecase1.

3.1.2.9. RETRIEVAL AND INFERENCE WITH GPT-4 TURBO

The final step involves utilizing OpenAI GPT-4 Turbo for retrieval and inference using the merged FAISS Vectorstore. This setup allows for sophisticated query handling, enabling users to retrieve relevant information and gain insights from the processed documents. For instance, when queried about specific mission objectives or requirements, GPT-4 Turbo can quickly and accurately provide the necessary information.

Example: Using the combined FAISS Vectorstore, GPT-4 Turbo can retrieve relevant information and provide inferences when queried about the NASA MAGIC Mission documents.

```
'''defining our model variable'''

model = "gpt-4-turbo"
```

```
'''defining llm variable'''

llm = ChatOpenAI(temperature=0.00001, model_name=model)
```

```
result=qa_chain('list all the requirements related to ocean')
result['result']
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/langchain_core/_api/deprecation.py:139: LangCh
ainDeprecationWarning: The method `Chain.__call__` was deprecated in langchain 0.1.0 and will be removed in 0.3.
0. Use invoke instead.
  warn_deprecated(
'1. "The mission shall be designed to achieve a minimum of 95% data coverage over the global ocean every 24 hour
s."\n2. "The mission shall be designed to achieve a minimum of 95% data coverage over the global ocean, land, and
ice-covered regions every 48 hours."\n3. "The TAR product shall cover the full swath of the optical instrument wh
enever the SAR instrument is acquiring data over the ocean."'
```

Figure 12: Code snippet for Data retrieval with example

3.1.2.10. USER INTERFACE

A user-friendly interface (UI) was created. This UI allows users to submit queries and receive answers efficiently. Additionally, the interface maintains a history of questions and answers, enabling continuous and context-aware interaction even when new queries are made.

Example: A user can ask "What are the mission objectives?" and then follow up with "What are the requirements for atmospheric studies?" The system retains the context and history of interactions, providing a seamless experience.

Source Display: The source display functionality was implemented in the backend using FAISS. When the answer is displayed on the UI, it also shows the source of the answer, including the document number and page number from where the information was extracted. This feature ensures transparency and allows users to verify the origin of the data.

Example: If the user asks about specific mission objectives, the interface will display the relevant objectives along with the document number and page number where these objectives are detailed

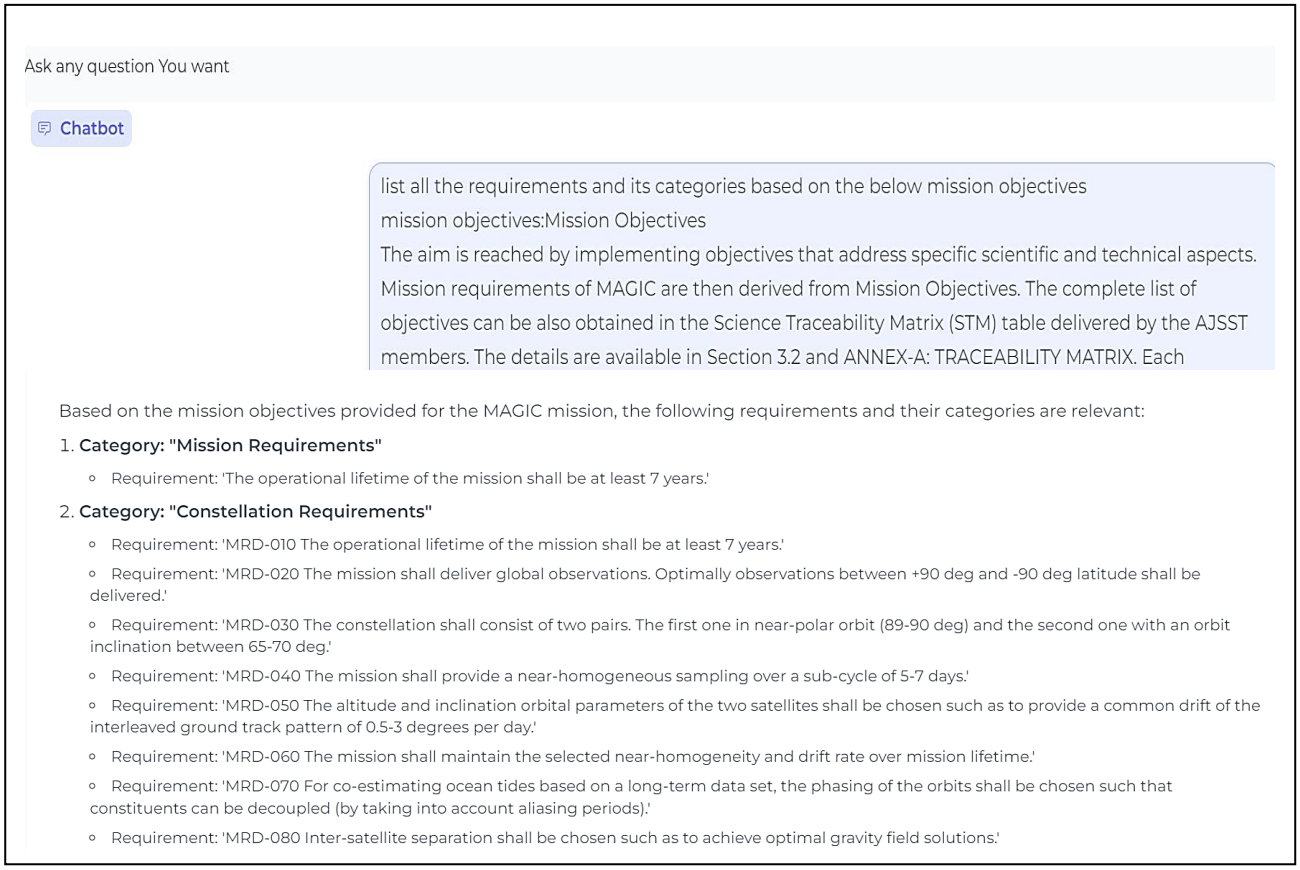


Figure 13: Snippet of the UI for Usecase 1

Example of Use Case -1 - Generating Requirements from Mission Objectives (refer to Figure 14)

Query: When a user inputs specific mission objectives, the tool generates detailed requirements that align with those objectives.

Outcome: The tool provides a structured set of requirements, categorized and relevant to the input objectives

3.2. USE CASE 2: GENERATING THE REQUIREMENTS

3.2.1. OBJECTIVE

The basic objectives for this use case are similar to those outlined in Use Case 1 (Refer to 3.1.1) , with a specific focus on utilizing the tool's capability to generate requirements from generalized prompts and categorize them accordingly. This includes both targeted mission objectives and generalized prompts, highlighting the tool's flexibility and robustness.

In addition to the basic objective, this use case emphasizes:

Utilizing the tool's capability to interpret generalized prompts.

Generating and categorizing requirements based on these generalized prompts.

3.2.2. SOLUTION METHODOLOGY

The methodology mirrors the comprehensive 10-step process outlined in Use Case 1 (refer to Section 3.1.2 for detailed steps and Figure 1 for the schematic diagram). The strategic preparation of the data ensures that the tool can not only generate requirements and categorize them based on mission objectives input into the query but also extend this capability to more generalized prompts.

For instance, the tool is designed to interpret and generate specific requirements from a straightforward prompt concerning a particular subject.

Example : The tool can handle more generalized prompts, such as "list all the requirements related to oceans."

Outcome: The tool interprets the prompt and proficiently generates a comprehensive set of requirements related to the specified subject. This functionality is illustrated in the accompanying image (Refer to Figure 15), showcasing the tool's ability to deduce and compile requirements based solely on the provided subject matter prompt.

This advanced capability significantly enhances the flexibility and utility of the tool, enabling it to address a wide range of queries with minimal input while maintaining accuracy and relevance. Such features underscore the robustness and versatility of data preparation methodology and the overall system design, making it an invaluable asset in diverse application contexts.

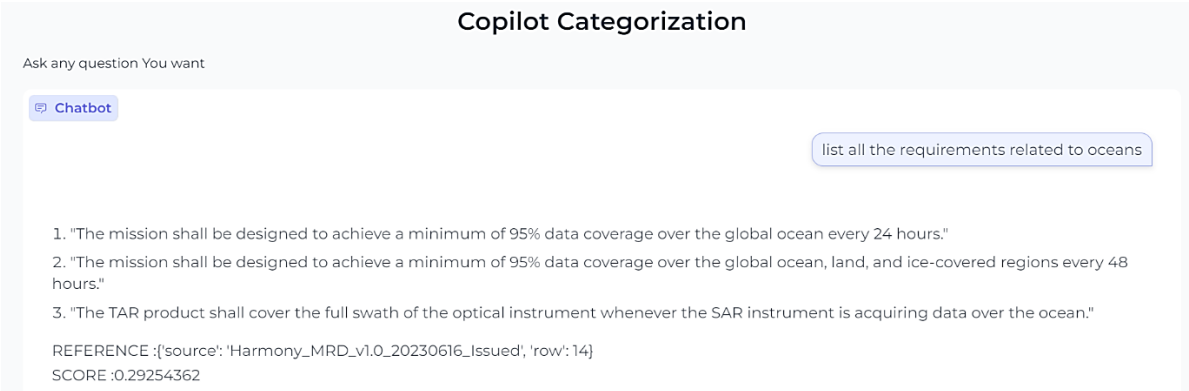


Figure 14: Snippet of the UI for Use case 2

3.3. USE CASE 3: DETECTING SIMILAR REQUIREMENTS

3.3.1. OBJECTIVES

Efficiently convert PDF documents into markdown files and split it page into single individiual page acting as a chunk.
Create embeddings of all the pages.
Store it in FAISS vectorstore.
Utilise the Similarity Search Function of FAISS to effectively identify similar documents/requirements.

3.3.2. SOLUTION METHODOLOGY

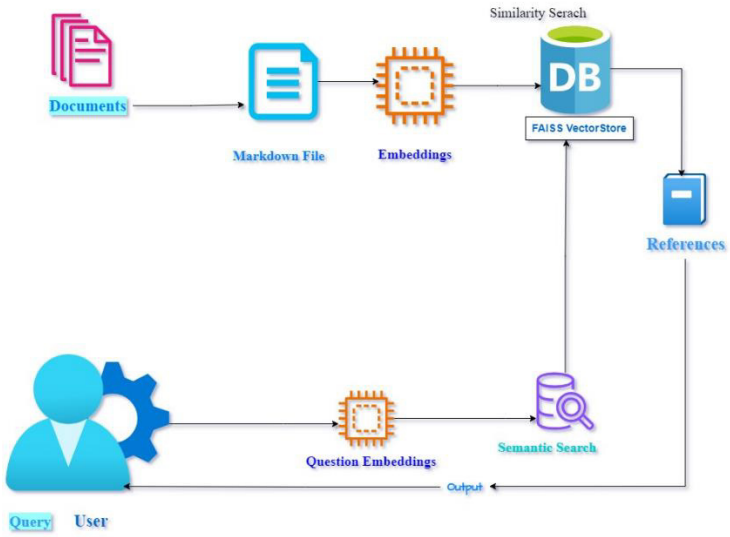


Figure 15: Process Flow for Detecting duplicate requirements

1. PDF Conversion: Transforming PDF documents into Markdown files for easier processing.
2. Chunking data: Dividing content into manageable chunks and obtaining preliminary insights.
3. Data Embedding: Creating embeddings for advanced search capabilities
4. Settiing up vectorstore: Building efficient search indexes.
5. Retreival: Retreiving the most similar content.
6. User Interface :Implementing a simple, user-friendly UI for query interaction and history management.

3.3.2.1. PDF CONVERSION

The SE Co-Pilot begins with the conversion of PDF documents into Markdown files. This step is crucial as it transforms the data into a more accessible and manageable format, laying the groundwork for subsequent processing. This is particularly important for systems engineering documents, which often contain complex details and specifications. Example: Suppose we have PDF documents from the NASA MAGIC Mission containing various mission details and requirements related to the mission payload. Each PDF is converted into Markdown format for easier processing.

```
'''in this block, we are converting the pdf into markdown contents and converting one page of the
markdown content as one chunk and saving it in "docs" named list. if any pdf has 100 pages than 100
chunks will be saved in "docs" list. Along with the page content/chunk its metadata i.e the source
of the content will also be saved in the same chunk.'''

chunks = []
page_number = []
pdf_name = []
error_list=[]

for bg in tqdm(range(0, len(all_pdf))):
    # Attempt to open the PDF file
    try:
        filename = all_pdf[bg]
        pdf = Pdf.open("pdfs"+"//"+filename)
    except Pdf.PdfError as e:
        # Log the error and continue to the next file
```

Figure 16: Code snippet for converting PDF into Markdown files.

3.3.2.2. CHUNKING DATA

The converted markdown files are splitted into each individual pages and these pages act as an single chunk.We had taken one page as one chunk for further processing.

```
metadata = [{"page_number": f"{i}", "pdf_name": f"{j.split('/')[-1]}"} for i, j in zip(page_number, pdf_na
docs = []

for i in range(len(chunks)):
    docs.append(Document(page_content=chunks[i], metadata=metadata[i]))
```

Figure 17: Code snippet for chunking data

3.3.2.3. DATA EMBEDDING

Embeddings are created for all the chunks.These embeddings transform the textual data into a numerical format that captures the semantic meaning of the text, enabling advanced search and retrieval capabilities.

Example: Generate embeddings(Hugging Face BAAI embeddings was used) for the text data in the chunks of "Magic Mission Requirements.pdf" and other PDFs.

```
'''initiating the hugging face embedding model.if want to use the open AI embeddings,
than just uncomment openAI embedding and comment hugging face embedding model'''


bi_enc_dict = {'mpnet-base-v2': "all-mpnet-base-v2",
               'instructor-large': 'hkunlp/instructor-large',
               'FlagEmbedding': 'BAAI/bge-large-en-v1.5'}
embeddings = HuggingFaceInstructEmbeddings(model_name=bi_enc_dict['FlagEmbedding'],
                                           query_instruction='Represent the question for retrieving supporting pa
                                           embed_instruction='Represent the paragraph for retrieval: ', model_kwa
```

Figure 18: Code snippet for OpenAI embeddings used for Usecase-2

3.3.2.4. SETTING UP VECTORSTORE

FAISS Vectorstore are used for storing all the embeddings.FAISS (Facebook AI Similarity Search) is a library that efficiently stores and searches through vector representations of data.

```
'''function to make the embedding,vectorstore and saving it on local.
it takes chunk size:int,overlap:int ,docs: preprocessed content ,embeddings: embedding model and
knowledge base : folder where vectorstore to be saved'''

def calling_FAISS_OnePageOneChunk(chunk_size, overlap, docs, embeddings, knowledge_base):
    text_splitter = CharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=overlap)
    documents = text_splitter.split_documents(docs)
    vectorstore = FAISS.from_documents(documents=documents, embedding=embeddings)
    vectorstore.save_local('knowledge_base/'+ knowledge_base)
```

Figure 19: Code snippet for Setting vectorstore for Usecase-2

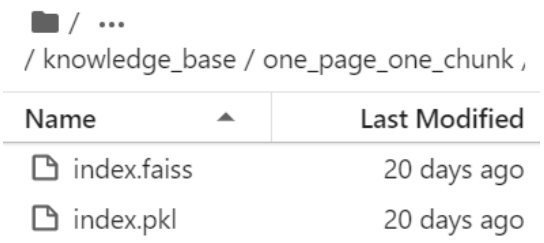


Figure 20: Snippet of saved Vectorstores for Usecase-2

3.3.2.5. RETRIEVAL

Utilised 'similarity_seacrh_with_score' functionality of FAISS vectorstore where we retreive the most similar documents from the vectorstore based on our query.The functionality returns us with a score as well which is based on the distance between two vectors.These scores ranges between 0 to 1 and 0 being exactly same and 1 being not at all similar, hence lower the score,better the results.

```
'''check the redundancy and similarity score'''

docs_and_scores = vectorstore.similarity_search_with_score(query)
dict1 = docs_and_scores[0][0].metadata
page_num = dict1['page_number ']
pdf_name = dict1['pdf_name ']
score = docs_and_scores[0][-1]
result = f"page_number : {page_num} , pdf_name {pdf_name} , score :{score}"
print(result)
```

```
page_number : 24 , pdf_name GMES_Sentinel3_MRD_V2.0_update.pdf , score :0.4585839807987213
```

Figure 21: Code snippet for retrieving similar documents with score functionality.

3.3.2.6. USER INTERFACE

A simple, user-friendly interface was created.UI allows users to submit queries and receive answers efficiently. Additionally, the interface maintains a history of questions and answers, enabling continuous and context-aware interaction even when new queries are made.

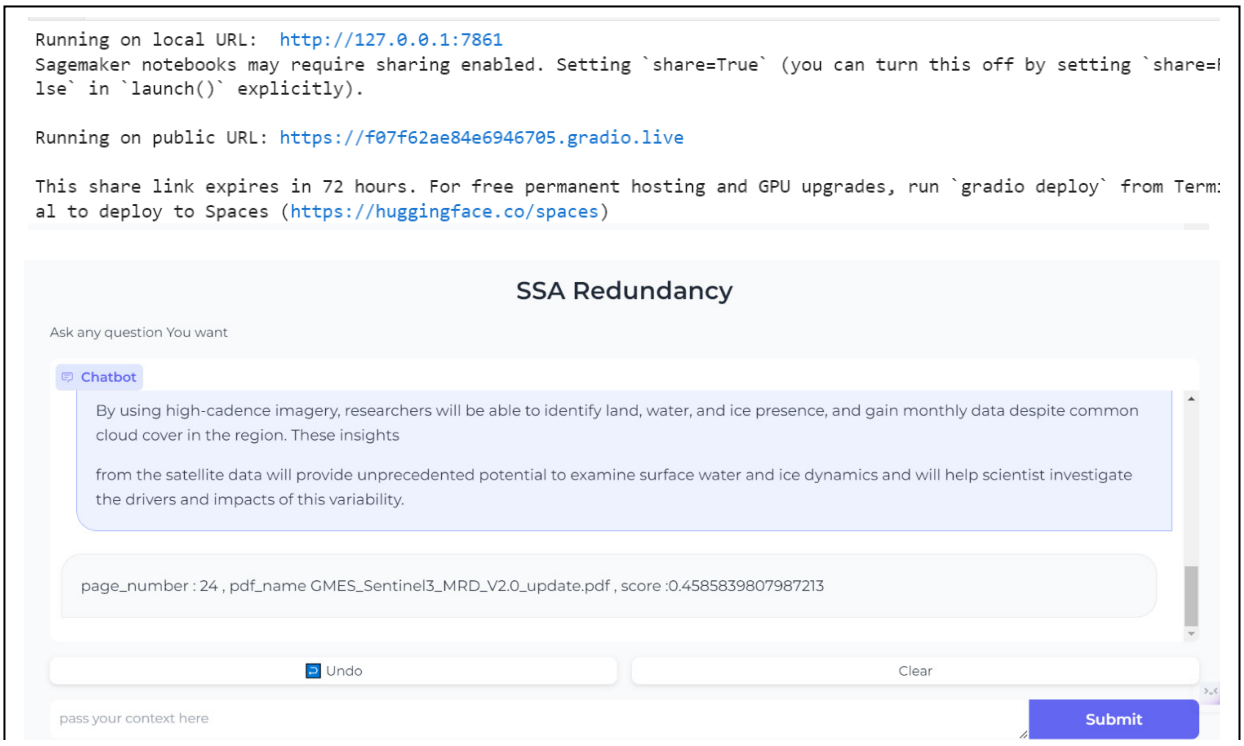


Figure 22: Snippet for code and UI along with example for Usecase2

4. CONCLUSION

The outlined process exemplifies the integration of manual extraction and advanced Generative AI-driven techniques to handle large volumes of complex documents in complex systems. By converting PDFs to Markdown, leveraging GPT-3.5 for initial processing and cleaning, manually extracting mission-critical information, and creating structured datasets, the SE Co-Pilot ensures both accuracy and efficiency. The creation of embeddings and FAISS Vectorstores further enhances the retrieval capabilities, culminating in the use of GPT-4 Turbo for sophisticated inference and query resolution. This systematic approach ensures that essential data is meticulously extracted, cleaned, organized, and made readily accessible, significantly enhancing the capability to manage and utilize large datasets effectively in systems engineering

5. FUTURE SE CO-PILOT ROADMAP

Building upon the success of the current implementation, the SE Co-Pilot platform enhacements are listed below.

Improved and advanced user interface.
Seamless uploading of complex systems engineering requiremeents documents.
Further fine-tuning for handling larger datasets.
Establishing quality checks for categorized data.
Measuring requirement quality.
Suggesting requirement improvements.
Classifying requirement prioritization.
Providing requirement auto-complete suggestions.
Offering insights based on selected requirements.
Generating requirement architecture.
Modeling requirement relationships.
Offering insights on similar objectives based on simple queries about new objectives.