

SMARTINTERNZ

GEMINIDECODE:

**Multilanguage Document
Extraction by Gemini Pro**

**GOOGLE CLOUD GENERATIVE AI INTERNSHIP
PROJECT REPORT**

SUBMITTED BY: ASHIWIJA MAYYA

TABLE OF CONTENTS

1. ABOUT PROJECT.....	2
2. TECHNICAL ARCHITECTURE	5
3. METHODOLOGY	6
4. REQUIREMENTS: SOFTWARE TOOLS.....	7
5. PROJECT STRUCTURE	9
6. OUTPUT SNAPSHOTS.....	17
7. CONCLUSION.....	20
8. REFERENCES.....	21

1. ABOUT PROJECT

In a world where visual data is ubiquitous, the ability to process and understand images across multiple languages has become essential for businesses and individuals alike. Traditional image processing tools often fall short when it comes to extracting meaningful information or handling multilingual content effectively. GeminiDecode, powered by the Google Gemini API, addresses this challenge by offering a comprehensive solution for multilanguage document extraction from images.

With **GeminiDecode**, users can upload images and ask a wide range of questions related to the content—whether it's summarizing the information, extracting embedded text, or translating content from one language to another. The system leverages advanced AI and machine learning models to automatically detect, interpret, and translate text in various languages, making it a versatile tool for anyone working with diverse image-based data. This report delves into the architecture, capabilities, and applications of GeminiDecode, showcasing its role in streamlining multilingual document processing.

Background:

The growing need to process and understand data from a variety of sources, including images, has given rise to numerous tools for document extraction and analysis. However, most traditional solutions are often limited to handling text in specific formats, such as PDFs or word documents, and are constrained to a single language. As organizations deal with increasingly diverse data sets, including images containing multilingual content, the limitations of these tools become evident.

Image-based data presents unique challenges for extraction, as the content is often unstructured and can include a mix of text, graphics, and complex layouts. Additionally, handling multilingual text in images adds another layer of complexity, requiring accurate text recognition and translation across multiple languages. These challenges necessitate the use of advanced technologies such as optical character recognition (OCR), natural language processing (NLP), and machine learning to extract and interpret meaningful information from such content.

GeminiDecode was developed to address these challenges by providing a flexible, intelligent solution for extracting information from images. Leveraging the power of **Google Gemini API**, the system integrates state-of-the-art OCR and language translation capabilities. This combination enables users to interact with image-based data more naturally, asking questions

like “What does this document say?” or “Summarize this content,” and receiving accurate, real-time responses. Whether for businesses processing documents from international clients, or researchers analyzing multilingual archives, GeminiDecode offers a comprehensive tool for multilanguage document extraction.

Objectives:

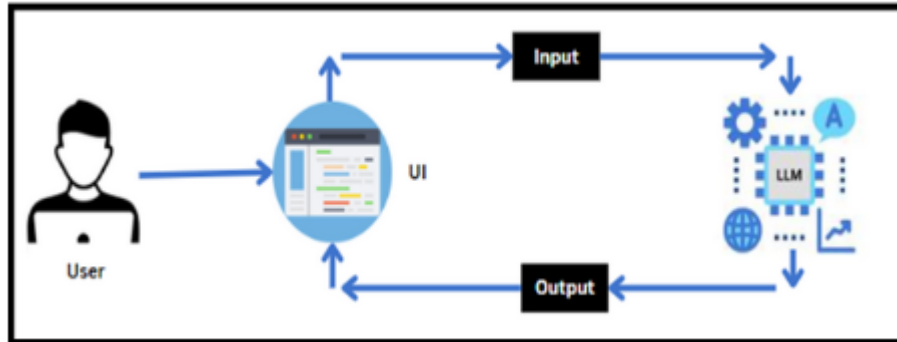
The primary goal of **GeminiDecode** is to provide a seamless, AI-driven solution for multilanguage document extraction from images. The following key objectives guide the development and functionality of the system:

- 1. Efficient Image-Based Document Processing:** To enable users to upload images and extract meaningful information, such as text and summaries, regardless of the complexity or format of the image.
- 2. Multilanguage Support:** To automatically detect and interpret text in multiple languages, ensuring that the system is capable of handling diverse language inputs with high accuracy.
- 3. Natural Language Interactivity:** To allow users to ask questions about the image content—be it summarization, specific text extraction, or language translation—and receive real-time, contextually accurate responses.
- 4. High-Quality Text Recognition and Translation:** To leverage the Google Gemini API for state-of-the-art optical character recognition (OCR) and translation, ensuring reliable extraction and conversion of text from images.
- 5. User-Friendly Interface and Workflow:** To create an intuitive user experience where users can easily interact with images and access the required information through a simple, accessible interface.
- 6. Scalability and Versatility:** To develop a solution that can scale across different industries and use cases, from business document processing to academic research and international communications.

Key Features:

- 1. Image-Based Data Extraction:** Users can upload images containing text, graphics, or complex layouts. The system processes the images to extract relevant information such as text, symbols, or other content, regardless of the document format.
- 2. Multilanguage Support:** The system automatically detects the language(s) present in the image and can handle multiple languages simultaneously. This allows users to extract and interpret content from images containing text in different languages.
- 3. Optical Character Recognition (OCR):** Powered by Google Gemini API, the built-in OCR technology accurately recognizes and extracts text from images, even when the text is embedded in challenging formats such as scanned documents or handwritten notes.
- 4. Language Translation:** Once text is extracted, the system can translate it into any target language. Users can request translations for entire documents or specific sections of the image's text.
- 5. Summarization and Text Analysis:** Users can ask the system to provide a summary of the image's content, making it easier to quickly grasp the main points of a document. Additionally, the system supports various text analysis tasks, such as keyword extraction and content categorization.
- 6. Interactive Querying:** Beyond extraction, users can ask questions directly about the content in the image. This feature allows for customized interactions, such as "What is the main topic?" or "Translate this paragraph into English."
- 7. Real-Time Processing:** The system processes images and returns results in real-time, offering immediate feedback to users. This makes GeminiDecode suitable for time-sensitive tasks like urgent document translation or analysis

2. TECHNICAL ARCHITECTURE



The image illustrates the general architecture of a large language model (LLM) application, GeminiDecode.

- **User:** The user, in this case, would be the person interacting with GeminiDecode. They would provide an image as input.
- **UI:** The user interface could be a web application, mobile app, or other platform through which the user interacts with GeminiDecode. It would receive the image from the user and send it to the LLM.
- **LLM:** The LLM, in this case, would be the Gemini model. It would process the image, analyze its content, and generate a response based on the user's query.
- **Input:** The input for GeminiDecode would be the image provided by the user.
- **Output:** The output from GeminiDecode would be the generated response, which could be a summary, text extraction, translation, or answer to a question about the image.

Specific to GeminiDecode:

- The LLM (Gemini) would use its understanding of both images and language to process the input image and generate a relevant response.
- The UI would likely provide options for different types of queries or responses, such as summarization, text extraction, or translation.

In essence, the image represents the flow of information in a typical LLM application, with GeminiDecode being a specific example of such an application focused on image processing and understanding.

3. METHODOLOGY

The methodology for building this system is fairly simple, an individual can follow all the activities listed below:

- Requirements Specification
 - Create a requirements.txt file to list the required libraries.
 - Install the required libraries
- Initialization of Google API Key
 - Generate Google API Key
 - Initialize Google API Key
- Interfacing with Pre-trained Model
 - Load the Gemini Pro pre-trained model
 - Implement a function to get gemini response
 - Implement a function to read PDF content
 - Write a prompt for gemini model
- Model Deployment
 - Integrate with Web Framework
 - Host the Application

Project Flow:

- User interacts with the UI to enter the input.
- User input is collected from the UI and transmitted to the backend using the Google API key.
- The input is then forwarded to the Gemini Pro pre-trained model via an API call.
- The Gemini Pro pre-trained model processes the input and generates the output.

4. REQUIREMENTS: SOFTWARE TOOLS

This project will be using the following technology stack:

1. Python
2. Python libraries from
 - streamlit
 - streamlit_extras
 - google-generativeai
 - python-dotenv
 - Pillow
 - langchain_google_genai
3. Streamlit
4. Google API
5. Machine Learning
6. Generative AI

4.1 Technology Stack:

The development of GeminiDecode relies on a robust technology stack, incorporating both foundational and cutting-edge tools to ensure seamless performance and advanced capabilities in image processing, text extraction, and multilanguage translation. The key components of the stack are as follows:

1. Python:

Python serves as the core programming language for the project, offering versatility and a rich ecosystem of libraries that make it ideal for AI and machine learning tasks. Python's extensive support for image processing, natural language processing (NLP), and integration with APIs makes it the optimal choice for this project.

2. Python Libraries

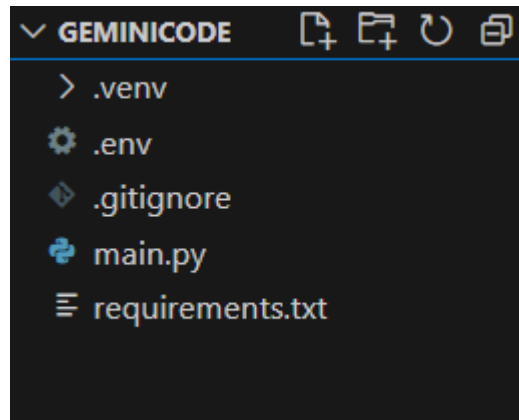
- **Streamlit:** This library provides an intuitive framework for building interactive web applications. In GeminiDecode, Streamlit is used to create the user interface, allowing users to upload images, ask questions, and view results in real time.
- **Streamlit Extras:** Enhances the functionality of Streamlit with additional components, providing custom elements and improved user interaction capabilities.
- **Google Generative AI:** This library integrates Google's cutting-edge generative

AI models, which power the natural language understanding and generation features within GeminiDecode, such as summarization and translation.

- **Python-dotenv:** Used for loading environment variables, ensuring that sensitive credentials, such as API keys for Google services, are securely managed.
 - **Pillow:** A powerful image-processing library in Python, Pillow is used for handling image uploads, preprocessing, and format conversions before the extraction and analysis tasks begin.
 - **LangChain Google GenAI:** This module integrates the specific features of Google's generative AI models into the LangChain framework, enabling GeminiDecode to harness the power of Google's state-of-the-art AI for advanced text generation and translation.
3. **Streamlit:** Streamlit is the core framework for the web-based user interface. It allows users to interact with GeminiDecode in an intuitive way, by uploading images, entering queries, and receiving responses in real-time. Streamlit's simplicity and flexibility make it a great choice for rapid development and deployment.
 4. **Google API:** The Google Generative AI and other Google Cloud APIs provide essential services to the project, including optical character recognition (OCR), text generation, and translation. These APIs enable the system to accurately process and interpret content from a wide range of languages and image formats.
 5. **Machine Learning:** Machine learning models are leveraged for tasks such as text recognition, summarization, and translation. These models are trained on large datasets and enable GeminiDecode to process complex image-based content with high accuracy and efficiency.
 6. **Generative AI:** By integrating Google's Generative AI models, GeminiDecode takes advantage of state-of-the-art language models for both text understanding and generation. This allows the system to perform advanced tasks like answering complex queries about image content, translating text in real-time, and generating summaries.

5. PROJECT STRUCTURE

Create the Project folder which contains files as shown below:

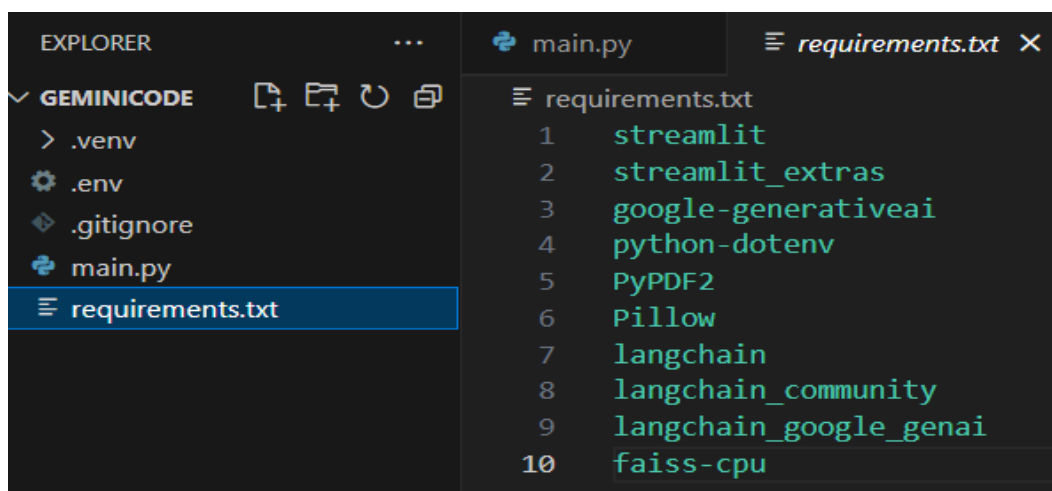


- 1) **.env file:** It securely stores the Google API key.
- 2) **main.py:** It serves as the primary application file housing both the model and Streamlit UI code.
- 3) **requirements.txt:** It enumerates the libraries necessary for installation to ensure proper functioning.

5.1 Step 1: Requirements Specification

Specifying the required libraries in the requirements.txt file ensures seamless setup and reproducibility of the project environment, making it easier for others to replicate the development environment.

5.1.1 Create a requirements.txt file to list the required libraries.



5.1.2 Install the required libraries

```
PS E:\GEMINICODE> .venv/Scripts/activate
(.venv) PS E:\GEMINICODE> pip install -r requirements.txt
```

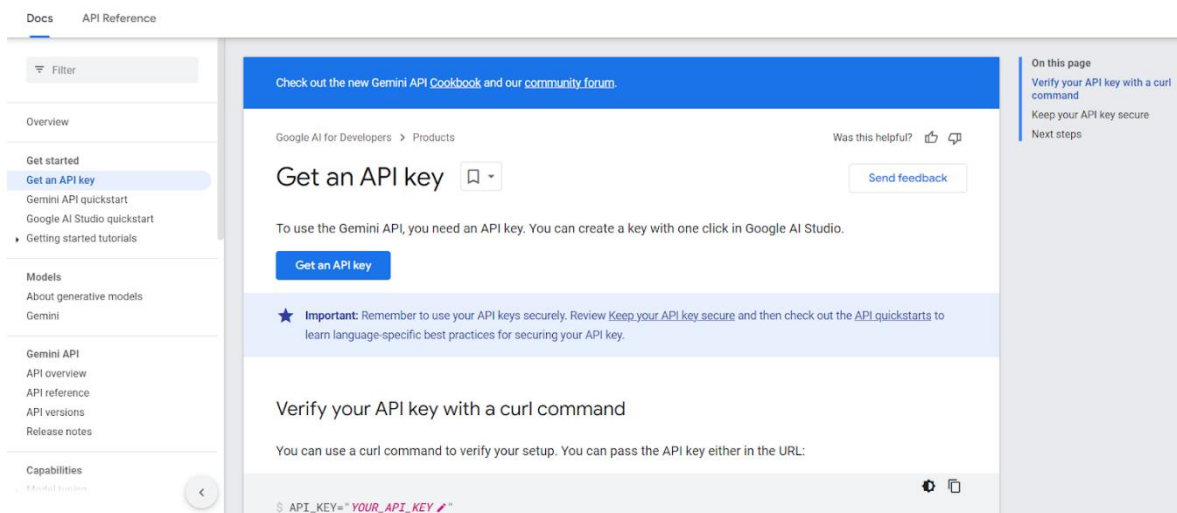
- Open the terminal.
- Run the command: `pip install -r requirements.txt`
- This command installs all the libraries listed in the `requirements.txt` file

5.2 Step 2: Initialization of Google API Key

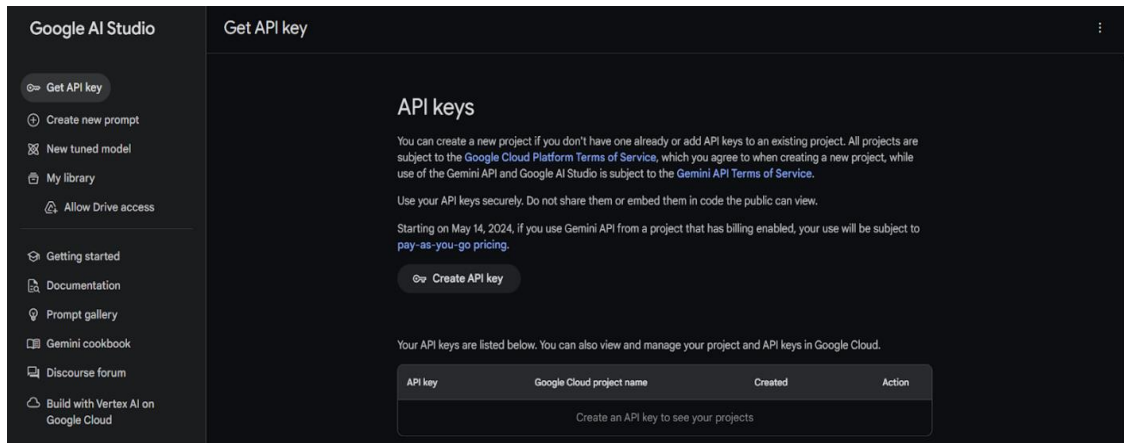
The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API requests, ensuring that only authorized users can access and utilize Google's services.

5.2.1 Generate Google API Key

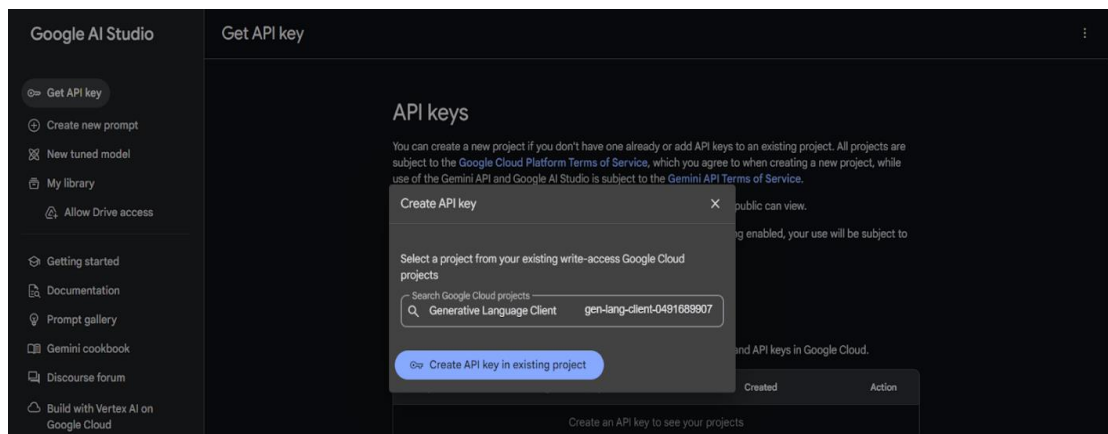
Click the link to access the following webpage: <https://ai.google.dev/gemini-api/docs/api-key>



After signing in to your account, navigate to the 'Get an API Key' option. Clicking on this option will redirect you to another webpage as shown below.

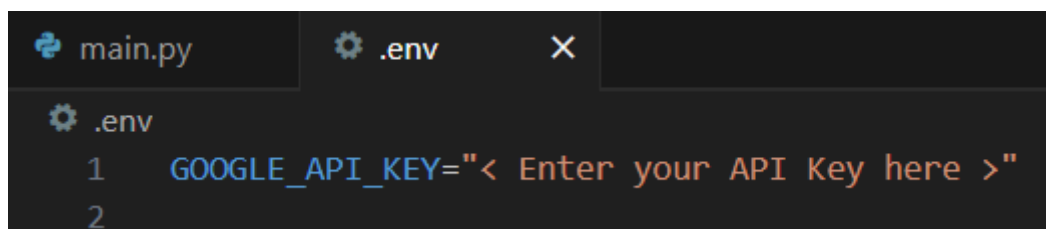


Next, click on 'Create API Key' and choose the generative language client as the project. Then, select 'Create API key in existing project'.



Copy the newly generated API key as it is required for loading the Gemini Pro pre-trained model.

5.2.2 Initialize Google API Key



- Create a .env file and define a variable named GOOGLE_API_KEY.
- Assign the copied Google API key to this variable.
- Paste the API key obtained from the previous steps here.

5.3 Interfacing with Pre-trained Model

To interface with the pre-trained model, we'll start by creating an app.py file, which will contain both the model and Streamlit UI code.

5.3.1 Load the Gemini Pro API

```
1  import streamlit as st
2  import os
3  import google.generativeai as genai
4  from PIL import Image
5  from dotenv import load_dotenv
6
7  # Load environment variables
8  load_dotenv()
9
10 # Set up the Google Generative AI API with the provided key
11 genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
12
```

The code for loading the Gemini Pro API begins by importing the necessary libraries, including `os`, `google.generativeai`, and `dotenv`. Next, the script utilizes the `load_dotenv()` function to load environment variables from a `.env` file into the runtime environment. By loading the environment variables, the script ensures that the API key is not hardcoded, which enhances security by keeping sensitive credentials confidential.

Finally, the Gemini Pro API is configured using the API key retrieved from the environment. The line `genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))` initializes the Google Generative AI client with the provided key. This configuration allows the application to authenticate requests to the Gemini API, enabling it to utilize the capabilities of the Generative AI models for tasks such as text generation, image processing, and other AI-driven functionalities.

5.3.2 Implement a function to get gemini response

```
54 # Function to interact with Gemini AI
55 def get_gemini_response(input_text, image, prompt):
56     model = genai.GenerativeModel('gemini-1.5-flash') # Ensure this is the correct usage
57     # Modify the content generation to suit your needs, based on the model's requirements
58     response = model.generate_content([input_text, image, prompt])
59     return response.text
60
```

The implementation of a function to get a response from the Gemini Pro API is a crucial part of the Streamlit application, as it facilitates interaction with the AI model. This function,

`get_gemini_response`, is designed to generate responses based on user-provided input, which includes extracted text from an image and a custom question. The function begins by defining `get_gemini_response(input_text)`, where `input_text` is a string containing the combined prompt for the Gemini model. This prompt typically consists of a predefined instruction and user-generated content, such as questions and extracted text from the uploaded document.

Within the function, the code initializes the Gemini model using the line `model = genai.Model('gemini-1.5-flash')`. This specifies the version of the Gemini model to be used for content generation. The choice of model can influence the quality and type of responses generated, so it's important to ensure that the selected model aligns with the application's goals.

Next, the function calls `response = model.generate(prompt=input_text)`, which sends the input prompt to the Gemini model for processing. The `generate` method is responsible for invoking the AI's capabilities to produce a relevant and coherent response based on the provided text.

Finally, the function extracts the generated content from the model's response.

Overall, this function serves as a bridge between user input and the Gemini Pro API, enabling the application to leverage advanced AI capabilities for document analysis and question answering.

5.3.3 Implement a function to read the Image and set the image format for Gemini Pro model Input

```
61 # Image file uploader
62 uploaded_file = st.file_uploader("Choose an image of the document:", type=["jpg", "jpeg", "png"])
63
64 # Display uploaded image if available
65 if uploaded_file is not None:
66     image = Image.open(uploaded_file) # Load the image as PIL.Image.Image
67     st.image(image, caption="Uploaded Image", use_column_width=True)
68
69     # Custom question input
70     custom_question = st.text_input("Ask a question about the document:")
71
```

The implementation of a function to read an image and prepare it for input into the Gemini Pro model is essential for the application's functionality. This function is responsible for handling the uploaded image, ensuring it is in the correct format for processing by the AI model. Typically, the function is named something like `prepare_image_for_gemini(image_file)`, where `image_file` represents the uploaded image object.

Within the function, the first step involves reading the image using a library such as Pillow, which simplifies image manipulation. The image is then loaded into memory, allowing for subsequent processing. To ensure compatibility with the Gemini Pro model, the function may convert the image to a specific format, such as RGB, which is suitable for AI processing.

Finally, the function returns the prepared image data alongside any extracted text, enabling the rest of the application to effectively utilize the image for generating queries to the Gemini Pro model. By encapsulating these tasks in a dedicated function, the code remains modular and easier to manage, ensuring smooth integration with the AI capabilities of the Gemini Pro model.

5.3.4 Write a prompt for gemini model

```
75 # Input prompt for Gemini AI
76 input_prompt = """
77 You are an expert in understanding images.
78 We will upload an image, and you will have to answer any questions based on the uploaded image.
79 """
```

Writing a prompt for the Gemini Pro model is a critical step in leveraging its capabilities for generating meaningful responses based on user input. A well-crafted prompt guides the AI in understanding the context and specifics of the request, ultimately influencing the quality of the output it generates.

To create an effective prompt, it's essential to begin with a clear and concise instruction that sets the context for the model. This could involve outlining the task at hand, such as summarizing information, answering questions, or extracting specific details from a given text. For instance, a prompt might start with a directive like, "You are an expert in understanding multilingual documents."

Next, the prompt should incorporate relevant content that the model needs to consider while generating a response. This may include extracted text from the uploaded document, details about the image, or specific user queries. For example, the prompt could include, "Based on the following text extracted from an image, please answer the question that follows."

Incorporating specific instructions within the prompt is also crucial. Clear questions or tasks help focus the model's response. For instance, one could phrase the user's question directly, such as, "What is the main idea presented in the text?" or "Can you summarize the key points from the document?".

5.4 Model Deployment

We deploy our model using the Streamlit framework, a powerful tool for building and sharing data applications quickly and easily. With Streamlit, we can create interactive web applications that allow users to interact with our models in real-time, providing an intuitive and seamless experience.

5.4.1 Integrate with Web Framework

```
13 # Streamlit app setup
14 st.set_page_config(page_title="GeminiDecode: Multilanguage Document Extraction by Gemini Pro")
15 st.header("GeminiDecode: Multilanguage Document Extraction by Gemini Pro")
16 page_bg_color = ""
17 <style>
18     [data-testid="stAppViewContainer"] {
19         background-color: #000000;
20         color: white;
21     }
22     [data-testid="stHeader"] {
23         background-color: rgba(0,0,0,0); # Make header transparent
24     }
25     [data-testid="stText"], [data-testid="stMarkdown"], h1, h2, h3, h4, h5, h6, p, label {
26         color: white;
27     }
28     .stButton>button {
29         background-color: #4CAF50; /* Optional: Style buttons if you want */
30         color: white;
31     }
32 </style>
33 ""
34 # Apply background color
35 st.markdown(page_bg_color, unsafe_allow_html=True)

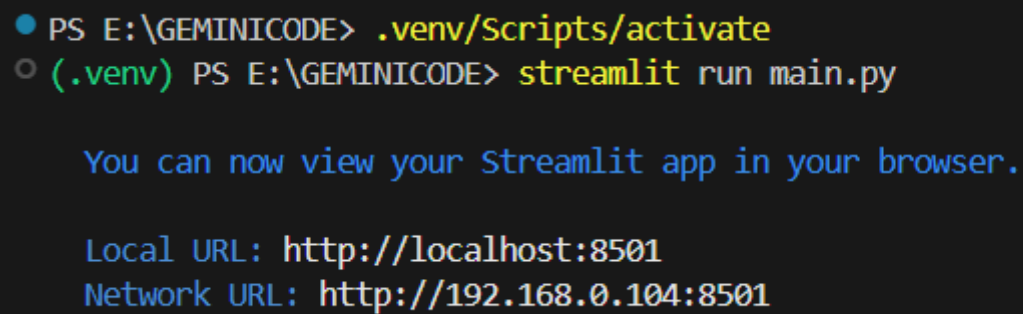
37 # Description text with custom style
38 text = ""
39 <p style='text-align: justify; font-family: serif;'>
40     Utilizing <strong>Gemini Pro AI</strong>, this project effortlessly extracts vital information from diverse multilingual
41     documents, transcending language barriers with precision and efficiency for enhanced productivity and decision-making.
42     Gemini Pro delivers unparalleled accuracy, speed, and clarity.
43 </p>
44
45 <p style='text-align: justify; font-family: serif;'>
46     With <strong>Gemini Pro's</strong> state-of-the-art AI</strong>, you can unlock valuable insights from any text, regardless of the
47     language or format, allowing you to focus on what truly matters—delivering results and achieving excellence.
48 </p>
49 ""
50
51 # Apply styled text
52 st.markdown(text, unsafe_allow_html=True)

81 # Handle the form submission
82 if submit and uploaded_file is not None and custom_question:
83     # Pass the image directly (as a PIL.Image.Image object) to the API
84     full_prompt = input_prompt + f"\nQuestion: {custom_question}"
85     response = get_gemini_response(full_prompt, image, uploaded_file.name)
86     st.subheader("The response is:")
87     st.write(response)
88 else:
89     if submit and not custom_question:
90         st.error("Please enter a question before submitting.")
91
```

In summary, integrating the Gemini Pro model with a web framework like Streamlit involves setting up the application structure, capturing user inputs, processing data through the AI model, and displaying results in a user-friendly manner. This integration not only enhances the functionality of the application but also makes advanced AI capabilities accessible to users through an intuitive web interface.

5.4.2 Host the Application

To host the application, go to the terminal, type - `streamlit run app.py`. Here `main.py` refers to a python script. Run the command to get the results.



```
● PS E:\GEMINICODE> .venv/Scripts/activate
○ (.venv) PS E:\GEMINICODE> streamlit run main.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.104:8501
```

6) OUTPUT SNAPSHOTS


6.1 UI of the Project

GeminiDecode: Multilanguage Document Extraction by Gemini Pro

Utilizing **Gemini Pro AI**, this project effortlessly extracts vital information from diverse multilingual documents, transcending language barriers with precision and efficiency for enhanced productivity and decision-making. Gemini Pro delivers unparalleled accuracy, speed, and clarity.

With **Gemini Pro's state-of-the-art AI**, you can unlock valuable insights from any text, regardless of the language or format, allowing you to focus on what truly matters—delivering results and achieving excellence.

Choose an image of the document:


 Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files


Get Document Information

6.2 Sample Image

Choose an image of the document:

 Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files


 220 Big bang.pdf-5301.jpg 359.6KB ×

Physics Factsheet

www.curriculum-press.co.ukNumber 220

The Big Bang Theory and the Evolution of the Universe

The Big Bang Theory
The hot Big Bang theory is that the whole Universe started its existence in a single point in space and has been expanding outwards. This does create the idea of the Universe exploding outwards and it is true that like an explosion, the Universe started in a dense and very hot state and has cooled as it expanded, but the more we understand about the Universe the less it looks like the dying embers of an exploded firework as it is often described.

 **The Cosmological Principle** is that the Universe looks the same from everywhere when you look at it on a very large scale. If you look at fog, it looks the same no matter where you are inside it. If you look too closely you will see the droplets, so it only remains true if you keep the scale of observation really big.

This idea allows cosmologists to simplify the Universe making it easier to describe and predict on a large scale. The scale used here is so large that if it were a fog, each droplet of water might represent a cluster of galaxies.

Olbers' Paradox – why is the night's sky dark?
The night's sky does not change from night to night, or indeed year to year. The planets move and comets come and go, but that is understood inside the confines of our Solar System. The Universe was assumed to be infinitely large, infinitely old and contain an infinite number of stars. It was only in the mid 1900s that this view started to be challenged.

6.3 Summarization task

Ask a question about the document:

Summarise the uploaded image

Get Document Information

The response is:

The image is a factsheet about the Big Bang Theory, which is the prevailing cosmological model for the universe. It explains the theory, stating that the universe started as a single point and is expanding outwards. The factsheet also discusses the "Olbers' Paradox" and the "Cosmological Principle", along with various observations that support the Big Bang Theory.

6.4 Translation task

Ask a question about the document:

Summarise the uploaded image. Make sure to translate it to german language

Get Document Information

The response is:

Dieser Text beschreibt die Urknalltheorie und erklärt die Gründe, warum diese Theorie die beste Erklärung für die Entstehung des Universums ist. Das Dokument erklärt die Urknalltheorie und ihre wichtigsten Aspekte, z.B. dass das Universum aus einem einzigen Punkt entstand und seitdem expandiert. Es geht auch auf die "Olbers Paradoxie" ein, welche besagt, dass der Nachthimmel nicht dunkel sein sollte, wenn es unendlich viele Sterne gäbe.

Die Urknalltheorie hingegen erklärt, dass die Entfernung von Sternen zu groß ist, um das menschliche Auge zu erreichen. Die Schlussfolgerung ist, dass das Universum endlich ist und einen Anfang hatte. Außerdem wird auf die Kosmische Hintergrundstrahlung (CMB) eingegangen und weitere wissenschaftliche Beweise, die die Urknalltheorie unterstützen.

6.5 Question answering task

Ask a question about the document:

What are the key ideas inferred from the image

Get Document Information

The response is:

The key idea is that the universe started as a single point in space and has been expanding outwards ever since. This idea is called the Big Bang theory, and it is supported by many observations, including the cosmic microwave background radiation, the red shift of distant objects, and the abundance of primordial helium. The Big Bang theory is a very exciting development for cosmology because it means that the universe is finite and can be measured. It also had a beginning, bringing the need for theories on how it started.

CONCLUSION

In conclusion, the development of the GeminiDecode application represents a significant advancement in the field of multilingual document extraction and analysis. By integrating the powerful capabilities of the Gemini Pro model with a user-friendly web interface built on Streamlit, this project effectively bridges the gap between complex AI functionalities and end-user accessibility.

Through the careful implementation of features such as image uploading, text extraction, and interactive querying, users can effortlessly extract valuable insights from diverse documents, transcending language barriers with remarkable precision and efficiency. The application not only enhances productivity but also empowers users to make informed decisions based on the insights generated by the AI.

Moreover, the modular approach taken in the project's architecture—covering aspects like image processing, prompt formulation, and API integration—ensures scalability and ease of maintenance. This allows for future enhancements and adaptations as user needs evolve and as advancements in AI technology continue to emerge.

Overall, the GeminiDecode project stands as a testament to the potential of generative AI in transforming how we interact with information. It opens up new possibilities for businesses, researchers, and individuals alike, enabling them to harness the power of AI-driven insights in their everyday workflows. As this technology continues to develop, the foundation laid by this project will serve as a stepping stone for further innovations in document analysis and multilingual communication, making advanced AI accessible to a broader audience.

REFERENCES

- <https://streamlit.io/>
- **NLP:** https://www.tutorialspoint.com/natural_language_processing/index.htm
- **Generative AI:** https://en.wikipedia.org/wiki/Generative_artificial_intelligence
- **About Gemini:** <https://deepmind.google/technologies/gemini/#introduction>
- **Gemini API:** <https://ai.google.dev/gemini-api/docs/get-started/python>
- **Gemini Demo:** <https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb>
- **Streamlit:** <https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/>
- Generative AI concepts