# Documentation for Each File

# 1. Data ingestion.ipynb

### Definition: Data Ingestion

Data ingestion refers to the process of **collecting, loading, and preprocessing data** from various sources into a system for further processing.

### Detailed Explanation

This notebook handles different types of data sources and prepares the data for further use. It supports:

1. **Text files** (using LangChain's TextLoader).
2. **PDF files** (using PyPDFLoader).
3. **Web scraping** (using WebBaseLoader and BeautifulSoup).
4. **Research papers from ArXiv** (ArxivLoader).
5. **Wikipedia articles** (WikipediaLoader).
6. **Weather data** (WeatherDataLoader).

After data ingestion, the notebook applies **text splitting** techniques to break down long documents into smaller, meaningful chunks.

### What You Did in Your Code

- **Installed dependencies** like langchain_community, pypdf, and bs4.
- **Loaded a text file** (speech.txt) using TextLoader.
- **Read a PDF file** (Surya GPT Docs.pdf) using PyPDFLoader.
- **Fetched content from a web page** using WebBaseLoader and applied **BeautifulSoup** filtering.
- **Loaded research papers** from ArXiv with ArxivLoader.
- **Extracted Wikipedia content** about Generative AI with WikipediaLoader.
- **Applied text splitting** using RecursiveCharacterTextSplitter to break documents into manageable chunks.

### Use Case

This notebook prepares text data from multiple sources, which is essential for generating embeddings and AI-based question-answering in your interview system.

---

# 2. Embeddings.ipynb

**Definition: Embeddings**

Embeddings are **numerical vector representations of text**, used to **capture semantic meaning** and allow AI models to perform similarity comparisons.

**Detailed Explanation**

This notebook explores different **text embedding techniques**:

1. **OpenAI Embeddings** (Paid API).
2. **Ollama Embeddings** (Open Source).
3. **Hugging Face Embeddings** (Open Source).

It focuses on **Ollama-based embeddings**, which allow text-to-vector conversion for applications like **semantic search and AI-generated interviews**.

**What You Did in Your Code**

- **Installed the Ollama library** (langchain-ollama).
- **Initialized an Ollama embedding model** (llama3.2).
- **Converted text into vector embeddings** using embed_documents().
- **Tested similarity search** by embedding a query ("What is the 3rd letter of Greek Alphabet?").
- **Switched to a different model (mxbai-embed-large)** and tested another embedding conversion.

**Use Case**

This module enables **similarity-based answer evaluation** in your interview system. Instead of checking for exact matches, it compares the **meaning** of responses.

---

# 3. HTML Textsplitter.ipynb

**Definition: HTML Text Splitting**

HTML text splitting is the process of **extracting and structuring textual content from HTML documents** while removing unnecessary tags and metadata.

**Detailed Explanation**

This notebook processes **HTML-based content** and applies **structured chunking** to retain semantic meaning.

It uses:

1. **HTML Header-Based Splitting** (`HTMLHeaderTextSplitter`).
2. **Recursive JSON Splitting** (`RecursiveJsonSplitter`).

**What You Did in Your Code**

- **Defined an HTML document** with headers (`h1`, `h2`, `h3`).
- **Extracted structured text** using `HTMLHeaderTextSplitter`.
- **Applied text chunking** to preserve document hierarchy.
- **Fetched and split data from an online article** (`https://plato.stanford.edu/entries/goedel/`).
- **Loaded and split a JSON document** from an API response.

**Use Case**

This is useful when processing **web-based resumes, interview transcripts, or HTML-based documents** for AI-driven interviews.

---

# 4. Huggingface embed.ipynb

**Definition: Hugging Face Embeddings**

Hugging Face provides **pre-trained NLP models** for generating text embeddings, which capture semantic meanings for AI tasks.

**Detailed Explanation**

This notebook integrates **Hugging Face's** sentence-transformers **library** to generate embeddings. It also implements **FAISS (Facebook AI Similarity Search)** for efficient vector-based retrieval.

**What You Did in Your Code**

- **Installed Hugging Face transformers** (sentence_transformers and langchain_huggingface).
- **Initialized an embedding model** (all-MiniLM-L6-v2).
- **Converted a sample text ("This is a Test Document") into an embedding.**
- **Stored embeddings in FAISS** for fast retrieval.
- **Performed similarity search** to retrieve answers based on embeddings.
- **Saved the FAISS index** for future use.

**Use Case**

This module **improves the AI interview system** by using **semantic search** to find the most relevant questions and responses.

---

# 5. Requirements.txt

**Definition: Dependency Management**

A requirements.txt file specifies all necessary **libraries and dependencies** required to run the project.

**Detailed Explanation**

This file ensures that the AI-powered interview system runs smoothly by **listing required Python packages**.

**What You Did in Your Code**

- Listed dependencies like langchain, pypdf, bs4, faiss-cpu, sentence_transformers, and langchain_huggingface.
- Ensured the **correct environment setup** for loading documents, generating embeddings, and running AI models.

**Use Case**

This file allows easy **reproducibility**, ensuring all necessary packages are installed before running the project.

---

## 6. Speech.txt

**Definition: Sample Text for NLP Processing**

This file contains **narrative text** that may be used for **text embedding, AI response evaluation, or speech-to-text testing**.

**Detailed Explanation**

It includes a **story-like passage**, which can be analyzed using NLP techniques like:

- **Named Entity Recognition (NER)** – To extract people and places.
- **Text Embedding** – To compare and retrieve similar content.
- **Speech-to-Text Testing** – To evaluate how accurately spoken content is transcribed.

**What You Did in Your Code**

- **Loaded speech.txt in data ingestion.ipynb.**
- **Split text into smaller chunks** using RecursiveCharacterTextSplitter.
- **Converted the text into embeddings** in huggingface embed.ipynb.

**Use Case**

This file may serve as a **sample input** to test **interview answer evaluations** or **NLP-driven AI features** in your system.

---

## Final Summary

Each of these files plays a **critical role** in your AI-powered interview system:

- data ingestion.ipynb → Loads data from text, PDF, web, Wikipedia, and APIs.
- embeddings.ipynb → Generates semantic text embeddings with Ollama.

- HTML Textsplitter.ipynb → Processes HTML and JSON documents for AI analysis.
- huggingface embed.ipynb → Uses Hugging Face models for text retrieval.
- requirements.txt → Manages dependencies for AI components.
- speech.txt → Provides a test dataset for NLP tasks.