**Here are several techniques commonly used in Generative AI to automatically create predefined prompts for an uploaded document:**

### **1. Template-based Prompt Generation**

* **Description:  
   Predefined placeholders and static templates.**
* **Example:  
   "Summarize the key points from {section\_name}."**

### **2. Metadata-based Prompt Generation**

* **Description:  
   Generate prompts dynamically using document metadata like title, author, and date.**
* **Example:  
   "Explain the implications of the report titled '{document\_title}' published on {publish\_date}."**

### **3. Extractive Prompt Generation (Keyword Extraction)**

* **Description:  
   Extract important keywords or terms using NLP (e.g., TF-IDF, RAKE).**
* **Example:  
   "Define and discuss the following terms extracted from the document: {extracted\_terms}."**

### **4. Semantic Prompting (Embedding-based)**

* **Description:  
   Use document embeddings to generate semantically relevant questions or prompts.**
* **Example:  
   "Discuss similarities between {section\_A} and {section\_B}, based on semantic analysis."**

### **5. Summarization-based Prompting**

* **Description:  
   Generate summary-based prompts using abstractive or extractive summarization models.**
* **Example:  
   "Provide a detailed explanation for: {extracted\_summary\_point}."**

### **6. Question Generation (QG)**

* **Description:  
   Automatically generate questions from the document content using Question Generation models.**
* **Example:  
   "What are the main arguments presented in section {section\_number}?"**

### **7. Named Entity-based Prompting**

* **Description:  
   Use named entity recognition (NER) to create entity-focused prompts.**
* **Example:  
   "Explain the significance of {named\_entity} as discussed in the document."**

### **8. Structured Document Prompting**

* **Description:  
   Leverage structural elements (headers, sections, tables) for prompts.**
* **Example:  
   "Describe the main points outlined under the heading '{header\_name}'."**

### **9. Prompt Refinement (LLM-based)**

* **Description:  
   Use large language models iteratively to refine initial prompts.**
* **Example:  
   Initial: "Give details about finance." → Refined: "Summarize the financial projections for fiscal year {year} in section {section}."**

### **10. Agentic Prompt Generation**

* **Description:  
   Agent-based systems dynamically generating prompts based on interactions and document context.**
* **Example:  
   "After reviewing section {section\_name}, discuss potential areas requiring further clarification."**

### **Commonly Used Tools & Libraries:**

* **OpenAI GPT Models (GPT-3.5, GPT-4, GPT-4 Turbo)**
* **Hugging Face Transformers (for summarization, question generation, and embeddings)**
* **SpaCy/NLTK (for NLP tasks like NER, Keyword extraction)**
* **LangChain/LlamaIndex (for structured prompting and embedding-based semantic retrieval)**
* **Azure Document Intelligence/Azure Cognitive Services (metadata extraction, layout analysis)**

## **Evaluation of GenAI Approaches for Generating Predefined Prompts**

### **1. OpenAI GPT-4 / GPT-3.5-turbo**

* **Description: Language models capable of generating human-like text.**
* **Use Cases: Generating example prompts, summaries, section-specific questions.**
* **Resources Needed:**
  + **OpenAI API key**
  + **Token management**
  + **Prompt engineering expertise**
* **Benefits:**
  + **High language fluency and contextual accuracy**
  + **Easy integration via API**
* **Limitations:**
  + **API cost and token limit**
  + **May generate irrelevant prompts if context is poor**
  + **Not deterministic; outputs may vary**

### **2. LangChain**

* **Description: Python framework for developing LLM-powered applications with modular components.**
* **Use Cases: Chaining document loaders, retrievers, and LLMs to generate prompts.**
* **Resources Needed:**
  + **LangChain library**
  + **Python knowledge**
  + **Chain configuration skills**
* **Benefits:**
  + **Modular and composable**
  + **Supports memory and prompt templates**
* **Limitations:**
  + **Steep learning curve**
  + **Slower execution with large chains**
  + **Debugging complexity**

### **3. LlamaIndex**

* **Description: Framework for indexing and querying document content with LLMs.**
* **Use Cases: Automatically extract relevant sections to base prompt generation.**
* **Resources Needed:**
  + **Document loaders**
  + **Index storage (in-memory/vector DB)**
  + **Python-based setup**
* **Benefits:**
  + **Efficient document structuring**
  + **Query-aware context fetching**
* **Limitations:**
  + **Additional pre-processing step**
  + **Costly if using embeddings frequently**
  + **Dependent on document cleanliness**

### **Combined Workflow Architecture**

1. **Document Upload**
2. **Text Extraction/OCR (Azure, Tesseract, etc.)**
3. **LlamaIndex (Indexing Document)**
4. **LangChain (Chaining Query + Prompt Template)**
5. **OpenAI GPT-4 (Generate Predefined Prompts)**
6. **Present Prompts in UI or Chatbot**

## **Summary Table**

| **Approach** | **Resources Needed** | **Benefits** | **Limitations** | **Implementation Effort** |
| --- | --- | --- | --- | --- |
| **GPT-4 / GPT-3.5** | **API Key, Prompt Engineering** | **Fluent output, flexible** | **Token limits, API costs, hallucinations** | **Medium** |
| **LangChain** | **Python, LangChain, chaining knowledge** | **Modular, memory support** | **Debug complexity, learning curve** | **High** |
| **LlamaIndex** | **Index setup, document loaders** | **Structured content retrieval** | **Preprocessing needed, sensitive to document quality** | **Medium** |