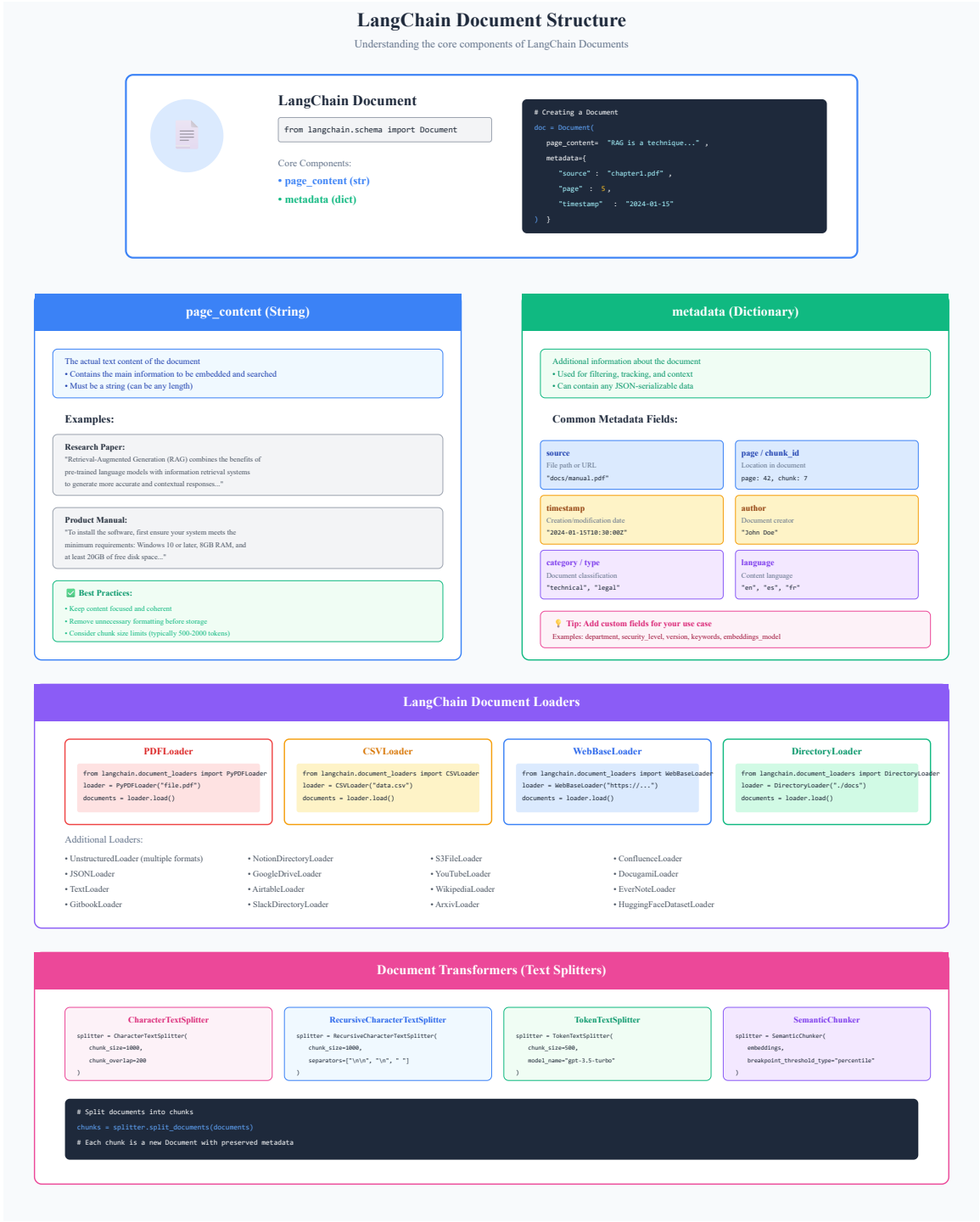
LangChain Document Structure
Understanding the core components of LangChain Documents
LangChain Document
from langchain.schema import Document
Core Components:
• page_content (str)
• metadata (dict)
# Creating a Document
doc = Document(
page_content= "RAG is a technique..." ,
metadata={
"source" : "chapter1.pdf" ,
"page" : 5 ,
"timestamp" : "2024-01-15"
) }
page_content (String)
The actual text content of the document
• Contains the main information to be embedded and searched
• Must be a string (can be any length)
Examples:
Research Paper:
"Retrieval-Augmented Generation (RAG) combines the benefits of pre-trained language models with information retrieval systems to generate more accurate and contextual responses..."
Product Manual:
"To install the software, first ensure your system meets the minimum requirements: Windows 10 or later, 8GB RAM, and at least 20GB of free disk space..."
Best Practices:
• Keep content focused and coherent
• Remove unnecessary formatting before storage
• Consider chunk size limits (typically 500-2000 tokens)
metadata (Dictionary)
Additional information about the document
• Used for filtering, tracking, and context
• Can contain any JSON-serializable data
Common Metadata Fields:
source
File path or URL
"docs/manual.pdf"
page / chunk_id
Location in document
page: 42, chunk: 7
timestamp
Creation/modification date
"2024-01-15T10:30:00Z"
author
Document creator
"John Doe"
category / type
Document classification
"technical", "legal"
language
Content language
"en", "es", "fr"
Tip: Add custom fields for your use case
Examples: department, security_level, version, keywords, embeddings_model
LangChain Document Loaders
PDFLoader
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader("file.pdf")
documents = loader.load()
CSVLoader
from langchain.document_loaders import CSVLoader
loader = CSVLoader("data.csv")
documents = loader.load()
WebBaseLoader
from langchain.document_loaders import WebBaseLoader
loader = WebBaseLoader("https://...")
documents = loader.load()
DirectoryLoader
from langchain.document_loaders import DirectoryLoader
loader = DirectoryLoader("./docs")
documents = loader.load()
Additional Loaders:
• UnstructuredLoader (multiple formats)
• JSONLoader
• TextLoader
• GitbookLoader
• NotionDirectoryLoader
• GoogleDriveLoader
• AirtableLoader
• SlackDirectoryLoader
• S3FileLoader
• YouTubeLoader
• WikipediaLoader
• ArxivLoader
• ConfluenceLoader
• DocugamiLoader
• EverNoteLoader
• HuggingFaceDatasetLoader
Document Transformers (Text Splitters)
CharacterTextSplitter
splitter = CharacterTextSplitter(
chunk_size=1000,
chunk_overlap=200
)
RecursiveCharacterTextSplitter
splitter = RecursiveCharacterTextSplitter(
chunk_size=1000,
separators=["\n\n", "\n", " "]
)
TokenTextSplitter
splitter = TokenTextSplitter(
chunk_size=500,
model_name="gpt-3.5-turbo"
)
SemanticChunker
splitter = SemanticChunker(
embeddings,
breakpoint_threshold_type="percentile"
)
# Split documents into chunks
chunks = splitter.split_documents(documents)
# Each chunk is a new Document with preserved metadata