LangChain 3-Hour Exercise Guide: News API + RAG System

OVERVIEW

Duration: 3 Hours Total (Exercise 1: 1.5 hours | Exercise 2: 1.5 hours)
Goal: Master LangChain loaders and complete RAG pipeline using free APIs

---

EXERCISE 1: News API Loader with AI Enhancement (90 minutes)

FREE NEWS API SETUP

Using NewsData.io API (Recommended - Like OpenWeather)

- **Sign up**: Go to https://newsdata.io/register (Free account)
- **Free Tier**: 200 requests/day (perfect for learning)
- **API Pattern**: Exactly like your weather API example

```
Base URL: https://newsdata.io/api/1/news
Required Parameter: apikey=YOUR_API_KEY
Optional Parameters: country=us, language=en, category=general
```

Example API Call Structure (Same as Weather API):

```
https://newsdata.io/api/1/news?
apikey=YOUR_API_KEY&country=us&language=en
```

STEP-BY-STEP IMPLEMENTATION

Step 1: Environment Setup (10 minutes)

1. Create `.env` file and add your NewsData API key
2. Create `config.py` with API key loader function (copy from your weather example)
3. Test basic API connection using `requests.get()`

Step 2: Create Custom NewsLoader Class (20 minutes)

Follow this exact pattern from your weather example: - Create class similar to `WeatherAPILoader` - Add constructor with `api_key` parameter - Add `load()` method that makes API request - Return the JSON response (list of news articles)

Step 3: AI Enhancement Pipeline (30 minutes)

Create AI enhancement for each news article: - Use your existing `PromptTemplate` pattern - Create prompts that take news article and generate: - AI summary (2 sentences max) - Sentiment analysis (positive/negative/neutral) - Key topics extraction (3-5 topics) - Credibility assessment (brief explanation)

Step 4: JSON Output Generator (20 minutes)

Combine original news + AI enhancements: - Loop through news articles - For each article, get AI enhancements - Structure final JSON with original data + AI additions - Save to `enhanced_news.json` file

Step 5: Testing & Validation (10 minutes)

- Test with 5-10 articles
- Verify JSON structure
- Check AI enhancement quality

Expected Output: `enhanced_news.json` file with original news data + AI enhancements

---

EXERCISE 2: Organization RAG System (90 minutes)

** FOCUS: Complete RAG Pipeline**

Core Learning: Data → Chunks → Vector Store → Retrieval → LLM Response

** STEP-BY-STEP RAG IMPLEMENTATION**

Step 1: Choose Organization & Data Sources (15 minutes)

Pick ONE organization with rich free data: - Tesla (recommended - lots of public data) - NASA - Your university - Local government - Microsoft

Data Sources to Use: - Official website pages (3-5 key pages) - Wikipedia page about the organization - 1-2 PDF documents (annual report, company overview)

Step 2: Data Loading Phase (20 minutes)

Use LangChain loaders to collect ALL data:

Web Content: - Use `WebBaseLoader` for official website pages - Load 3-5 important pages (About, Products, History, etc.)

Wikipedia: - Use `WikipediaLoader` to get comprehensive organization info - Load main article + related pages

PDF Documents: - Use `PyPDFLoader` for annual reports or public documents - Download PDFs first, then load with PyPDFLoader

Combine all documents into one list

Step 3: Chunking Strategy (15 minutes)

Split all documents for better retrieval: - Use `RecursiveCharacterTextSplitter` - Set chunk_size=1000, chunk_overlap=200 - Apply to ALL loaded documents - Count total chunks created

Step 4: Vector Store Creation (15 minutes)

Store chunks in Chroma database: - Use your existing embeddings setup - Create Chroma database with `persist_directory` - Store ALL document chunks - Name database after your organization (e.g., `./chroma_tesla`)

Step 5: Boundary Enforcement System (10 minutes)

Create organization-only question filter: - Create simple function to check if question mentions your organization - Use basic keyword matching or simple LLM prompt - Return "I only answer questions about [ORGANIZATION]" for off-topic queries

Step 6: Retrieval System (10 minutes)

Set up document retrieval: - Create retriever from your vector store - Set k=3 (retrieve top 3 relevant chunks) - Test retrieval with sample questions about your organization

Step 7: Complete RAG Pipeline (5 minutes)

Chain everything together: 1. Check if question is about organization 2. If yes: retrieve relevant documents 3. Format documents as context 4. Create prompt with context + question 5. Get LLM response 6. Return answer with source indication

COMMAND LINE INTERFACE REQUIREMENTS

Your CLI must handle:

```
Welcome to [ORGANIZATION] Assistant
Ask questions about [ORGANIZATION] only!

User: "What does Tesla do?"
```

```
  Assistant: [Answer based on retrieved documents]

User: "What is Apple's stock price?"
  Assistant: "I only answer questions about Tesla. Please ask
about Tesla."

User: "quit"
  System: Exit gracefully
```

TESTING YOUR RAG SYSTEM

Test Cases to Verify:

1. **Boundary Enforcement**: Ask about different organizations
2. **Document Retrieval**: Ask specific questions that should find relevant chunks
3. **Source Context**: Verify answers use retrieved information
4. **Edge Cases**: Empty questions, very long questions

Quality Checks:

- Can retrieve information from web pages? ✓
- Can retrieve information from Wikipedia? ✓
- Can retrieve information from PDFs? ✓
- Rejects off-topic questions? ✓
- Provides relevant answers with context? ✓

DELIVERABLES CHECKLIST

Exercise 1 Outputs:

- [ ] Working NewsLoader class
- [ ] AI enhancement pipeline
- [ ] `enhanced_news.json` file with 10+ articles
- [ ] Clean, documented Python code

Exercise 2 Outputs:

- [ ] Multi-source data loading (web + wiki + PDF)
- [ ] Chunked and stored documents in Chroma
- [ ] Working CLI with organization boundary enforcement
- [ ] Successful retrieval and response generation

TIME MANAGEMENT GUIDE

Hour 1: News API Exercise

- 0-10 min: API setup and testing
- 10-30 min: NewsLoader class creation
- 30-60 min: AI enhancement pipeline

Hour 2: Complete News Exercise + Start RAG

- 0-30 min: JSON output and testing (complete Exercise 1)
- 30-45 min: Choose organization and plan data sources
- 45-60 min: Load all data sources

Hour 3 for DEEPSEED's WAR: Complete RAG System

- 0-15 min: Chunk and store in vector database
- 15-30 min: Build retrieval system
- 30-45 min: Create boundary enforcement and CLI
- 45-60 min: Testing and debugging

COMMON PITFALLS TO AVOID

1. **API Limits**: Don't make too many API calls during testing
2. **Large Files**: Check PDF size before loading (keep under 10MB)
3. **Chunk Size**: If retrieval is poor, adjust chunk size
4. **Organization Scope**: Keep boundary checking simple but effective
5. **Error Handling**: Add basic try-catch for API calls

SUCCESS CRITERIA

Exercise 1 Complete When: - News API returns data successfully - AI enhancements work for each article - JSON file contains structured, enhanced news data

Exercise 2 Complete When: - Can load data from multiple sources - Documents are chunked and stored in Chroma - CLI answers organization questions correctly - Rejects off-topic questions appropriately - Retrieval system finds relevant information

Both exercises should run without errors and produce expected outputs.