# **Assignment 2 Journal: Building Your Own AI Search Assistant using LangGraph**

## **Part 1: Web Search Implementation**

### **Design Decisions**

Explain the key design decisions you made for your web search implementation:

* What prompt engineering techniques did you use to format search results?
* If you compared generating your own answer versus using the answer returned by Tavily (**include\_answer=True**), what differences did you notice?
* How did adjusting the number of results returned by Tavily affect the quality of your answers?

### **Challenges and Solutions**

What other challenges did you face during implementation, and how did you solve them?

## **Part 2: Document RAG Implementation**

### **Design Decisions**

#### **1. Document Processing**

* Describe your approach to document loading and chunking. What chunking parameters did you use, and did you try different parameters?
* Note down how you structured your RAG prompts to effectively use the retrieved context.

#### **2. Retrieval Strategy**

* How did you determine relevance when retrieving document chunks?
* What approach did you take to handle queries that weren’t answerable from the OPM documents?
* Include an example showing how your system responds to both in-scope and out-of-scope queries.

#### **3. Source Attribution**

* How did you implement source attribution from the OPM documents?
* Share an example of how your system cites information from specific OPM reports.

### **Technical Challenges**

* Have you tried different document loaders? If so, did you notice any significant differences?
* **Imagine you need to add new documents to your knowledge store. What changes would you need to make?**

## **Part 3: Corrective RAG-lite Implementation**

### **Design Decisions**

#### **1. Relevance Assessment**

* How did you implement the ranking/classification of document chunk relevance?
* Did you use a binary approach (relevant/not relevant) or a scoring mechanism? Why?
* What threshold or criteria did you use to determine if the retrieved information was “sufficient”?

#### **2. Knowledge Source Selection**

* [Optional] Did you implement the optional feature to directly use web search for queries unrelated to OPM documents? If so, how did you determine which queries were unrelated?
* Share an example of a query where your system switched from document RAG to web search due to insufficient information.

#### **3. LangGraph Implementation**

* How did you structure your **StateGraph** to implement the Corrective RAG-lite workflow?
* How did you use conditional edges to route between document retrieval and web search?
* How did you manage state across the different nodes in your graph?

### **Technical Challenges**

* What was the most difficult aspect of implementing the relevance assessment component?
* Share a specific example of a query that was challenging for your Corrective RAG-lite system and how it was handled.

### **Key Learnings**

* What insights did you gain about the effectiveness of Corrective RAG-lite compared to using document RAG or web search alone?
* How did LangGraph’s features help or hinder your implementation of the Corrective RAG-lite workflow?
* What would you change about your approach if you were to implement a more sophisticated version of Corrective RAG?
* **Imagine you need to add new documents to your knowledge store. What changes would you need to make for the Corrective RAG-lite approach?**

## Bonus: Metrics and Instrumentation

Only answer this if you used Comet Opik to complete the bonus section of the assignment.

Similar to assignment 1, combining your learnings from the lecture videos as well as assignment experience, how did you use evaluation metrics to understand the accuracy of the system? What are your learnings from there?

## **Overall Reflection**

### **Lessons Learned**

* Describe one significant challenge you faced for each part of the assignment.
* What were the most valuable insights you gained from implementing these systems?
* [Optional] Apart from Corrective RAG-lite, have you tried implementing any other RAG-based workflows? If so, what were the results?