# Network Support RAG System - Cisco & Mikrotik

Emmanuel Chibua, NU ID: 002799484

July 11, 2024

## 1 Introduction

The Network Support RAG (Retrieval-Augmented Generation) System is designed to assist network engineers and technicians with queries related to Cisco and Mikrotik devices. The system leverages LangChain and OpenAI's GPT-4 to provide accurate and relevant responses based on the context provided in the input documents. This report details the approach taken in developing this system, the challenges faced, and the solutions implemented to overcome these challenges.

## 2 Approach

### 2.1 Initialization and Setup

The project began with the initialization and setup of the necessary environment. This included installing required Python packages and setting up the Streamlit interface for user interaction. The main packages used in this project were Streamlit, PyPDF2, LangChain, and OpenAI's API.

### 2.2 PDF Document Parsing

The next step involved parsing PDF documents. The PyPDF2 library was used for this purpose. The documents were split into manageable chunks using the RecursiveCharacterTextSplitter from LangChain. This was necessary to handle large documents efficiently and ensure that the text could be processed by the OpenAI model.

### 2.3 Vector Store Creation

Text chunks obtained from the PDF documents were converted into embeddings using OpenAI's Embeddings API. These embeddings were then stored in a vector store using FAISS (Facebook AI Similarity Search). This step was crucial for enabling efficient similarity search during the question-answering process.

### 2.4 Conversational Chain

A predefined prompt template was created to generate responses based on user queries. This template was used to load a question-answering chain using the GPT-4 model from OpenAI. The chain was responsible for processing user questions and generating relevant answers based on the context provided in the documents.

### 2.5 Streamlit Interface

A user-friendly web interface was developed using Streamlit. This interface allowed users to interact with the RAG system by entering their queries and viewing the generated responses. The interface also displayed the chat history to provide context for ongoing conversations.

## 3 Challenges Faced

### 3.1 Handling Large Documents

One of the main challenges was handling large PDF documents efficiently. Splitting the documents into manageable chunks was essential to avoid memory issues and ensure that the text could be processed effectively by the model.

### 3.2 Efficient Similarity Search

Performing efficient similarity search on large text embeddings was another challenge. FAISS was used to create a vector store that allowed for fast and accurate similarity searches. This step was crucial for retrieving relevant context when generating answers to user queries.

### 3.3 Streamlit Interface Design

Designing an intuitive and user-friendly Streamlit interface was important for ensuring a good user experience. This involved dynamically updating the chat history and handling user inputs efficiently. Session state management was used to store chat history and ensure that the interface responded appropriately to user actions.

## 4 Solutions Implemented

### 4.1 Recursive Text Splitting

To handle large documents, the RecursiveCharacterTextSplitter from LangChain was used to split text into smaller chunks. This ensured that the documents could be processed efficiently without running into memory issues.

### 4.2 FAISS for Vector Store

FAISS was chosen for creating the vector store due to its efficiency in performing similarity searches on large datasets. This allowed for fast retrieval of relevant context when generating answers to user queries.

### 4.3 Streamlit Session State

Session state management in Streamlit was used to store and update the chat history dynamically. This provided a seamless user experience by maintaining the context of the conversation and ensuring that the interface responded appropriately to user inputs.

## 5 Conclusion

The Network Support RAG System was successfully developed to assist network engineers and technicians with queries related to Cisco and Mikrotik devices. Despite the challenges faced, effective solutions were implemented to ensure the system's efficiency and user-friendliness. The use of LangChain and OpenAI's GPT-4, combined with a well-designed Streamlit interface, resulted in a robust and reliable support system.