High-Level Summary of rag\_qa\_system.py

This document provides a high-level summary and explanation of all functions, classes, and core components in the `rag\_qa\_system.py` file. The system implements a full Retrieval-Augmented Generation (RAG) pipeline that supports document ingestion, vector and keyword retrieval, LLM-based answering, and automatic evaluation.

## 1. Imports and Setup

The code installs and imports necessary libraries including LangChain, HuggingFace, FAISS, BM25, PyPDF, etc. It also handles authentication with HuggingFace using a login token.

## 2. Constants Definition

Constants like chunk size, embedding model, LLM model, and document directory are defined for flexible tuning.

## 3. Document Processing Functions

- `load\_documents`: Loads PDF documents using LangChain’s DirectoryLoader.  
- `chunk\_documents`: Splits large documents into overlapping chunks using RecursiveCharacterTextSplitter.

## 4. Vector Store and Indexing

- `create\_vector\_store`: Converts chunks into vector embeddings and stores them using FAISS.  
- `create\_bm25\_index`: Creates a BM25 keyword index from document chunks for keyword-based retrieval.

## 5. Retrieval Methods

- `semantic\_search`: Uses FAISS similarity search.  
- `keyword\_search`: Uses BM25 keyword-based scoring.  
- `hybrid\_search`: Combines semantic and keyword scores using weighted fusion.

## 6. Language Model Initialization

- `initialize\_llm`: Loads a HuggingFace text generation model and tokenizer, wrapped into a pipeline.

## 7. Prompt and Response Generation

- `format\_rag\_prompt`: Formats a structured prompt by combining retrieved context and question.  
- `generate\_response`: Uses the model to generate an answer based on the formatted prompt.

## 8. Evaluation and Wrappers

- `RAGEvaluator`: Runs RAGAS-based evaluations using metrics like Faithfulness and Relevance.  
- `SimpleEvaluator`: Custom evaluator that generates a reverse question and computes cosine similarity.  
- `HuggingFaceLLMWrapper` and `HuggingFaceLLM`: Used to adapt HuggingFace pipelines for use in evaluation libraries.

## 9. Core RAGPipeline Class

Main class that combines all pipeline steps:  
- `load\_and\_process\_documents`  
- `initialize\_retrieval`  
- `initialize\_llm`  
- `query`: Executes full RAG pipeline for a question.  
- `experiment` and `grid\_search`: Run various configurations for testing retrieval and generation performance.

## 10. Execution & Usage

The final part runs the pipeline step-by-step:  
- Initialize embeddings and LLM  
- Load documents and create indexes  
- Run a sample query and evaluate it  
- Conduct a grid search across different chunk sizes, search types, and k-values.