**Step#2 : Add Few SHot examples for better Response Customization**

Adding few shot examples to instruction set of the assistant api will tailor the original response into short and sweet response.

“But P**roblem is the OpenAI assistant continues to respond in its default style(reads more like something from ChatGPT ) rather than referring to the provided document and tailoring its responses accordingly.”**

**STEP#3 : Adding Knowledge(RAG)**

Setting up RAG is complex, it consists of handful of non-trivial steps:

1. Chunking Documents
2. Setting up Vector Database
3. Building semantic search function
4. Fusing search results into Context Window

However this entire step is not required when building with OpenAI api, we just need to

1. Upload our documents for retrieval
2. Add RETRIEVAL CAPABILITY to AI assistant

OpenAI automatically parses and chunks your documents, creates and stores the embeddings, and use both vector and keyword search to retrieve relevant content to answer user queries.

How RAG(retrieval augmented generation) is different from OpenAI internet browsing tools?

-→ In internet browsing, we have no control on which data we are about to access(so which data we are about to augment to the model’s response), we have no control on searching the document. All the process i.e. searching, ranking or Search Engine Operation is designed and controlled by google , no customization.

RAG gives us ability to create our own search engine. Here is how it works

1. It takes the user input, Simplifies it(

* Rewrites user queries to optimize them for search, Better queries —> Better Search—> Better Document finding—> Better Result/Response.
* Breaks down complex user queries into multiple searches it can run in parallel (Fast Searching through the content).

1. Search Operation is done in our custom store —> Vector\_stores ( • Runs both keyword and semantic searches across both assistant and thread vector stores.)
2. Ranking or Re-Ranking is also handled.( • Reranks search results to pick the most relevant ones before generating the final response.)