Text Analytics & Business Application

Chatbots

Qinglai He
Department of Operations and Information Management
Wisconsin School of Business

WISCONSIN SCHOOL OF BUSINESS

Outline of Today's Class

- History & Applications
- A Taxonomy of Chatbots
- Prompt Engineering
- Retrieval Augmented Generation (RAG) for LLMs

History & Applications

Chatbots

- Chatbots are interactive systems that allow users to interact in natural language.
- They generally interact via **text** but can also use speech interfaces.
- Examples:
  - Apple Siri
  - Facebook Messenger
  - Google Assistant
  - Amazon Alexa

Some Important Milestones of Chatbots

- **ELIZA** was the very first chatbot.
  - It was created by Joseph Weizenbaum in 1966 and it uses pattern matching and substitution methodology to simulate conversation
- **Dr. Sbaitso** is a chatbot created by Creative Labs for MS-Dos in 1992.
  - It is one of the earliest efforts of incorporating A.I. into a chatbot and is recognized for its full voice operated chat program.
- **SmarterChild** was in many ways a precursor of Siri and was developed in 2001
- **Siri** was formed by Apple for iOS in 2010; it is an intelligent personal assistant and learning navigator that uses a natural language UI.
- **ChatGPT**…

Why Chatbots?

Studies show:

- 71% of people use chatbots to solve their problem fast.
  - 56% would rather message than call customer service.
  - 53% are more likely to shop with businesses they can message.
- 1/5 consumers would consider purchasing goods and services from a chatbot.

Applications of Chatbots

- Shopping and E-commerce
- News and Content Discovery
- Customer Service
- Medical
- Legal

A Taxonomy of Chatbots

A Taxonomy of Chatbots

- **Exact answer or FAQ bot**
  - These chatbots are linked to a fixed set of responses and retrieve a correct response based on understanding the user's query
  - Generally, one response from the user does not depend on the previous responses

FAQ Bot

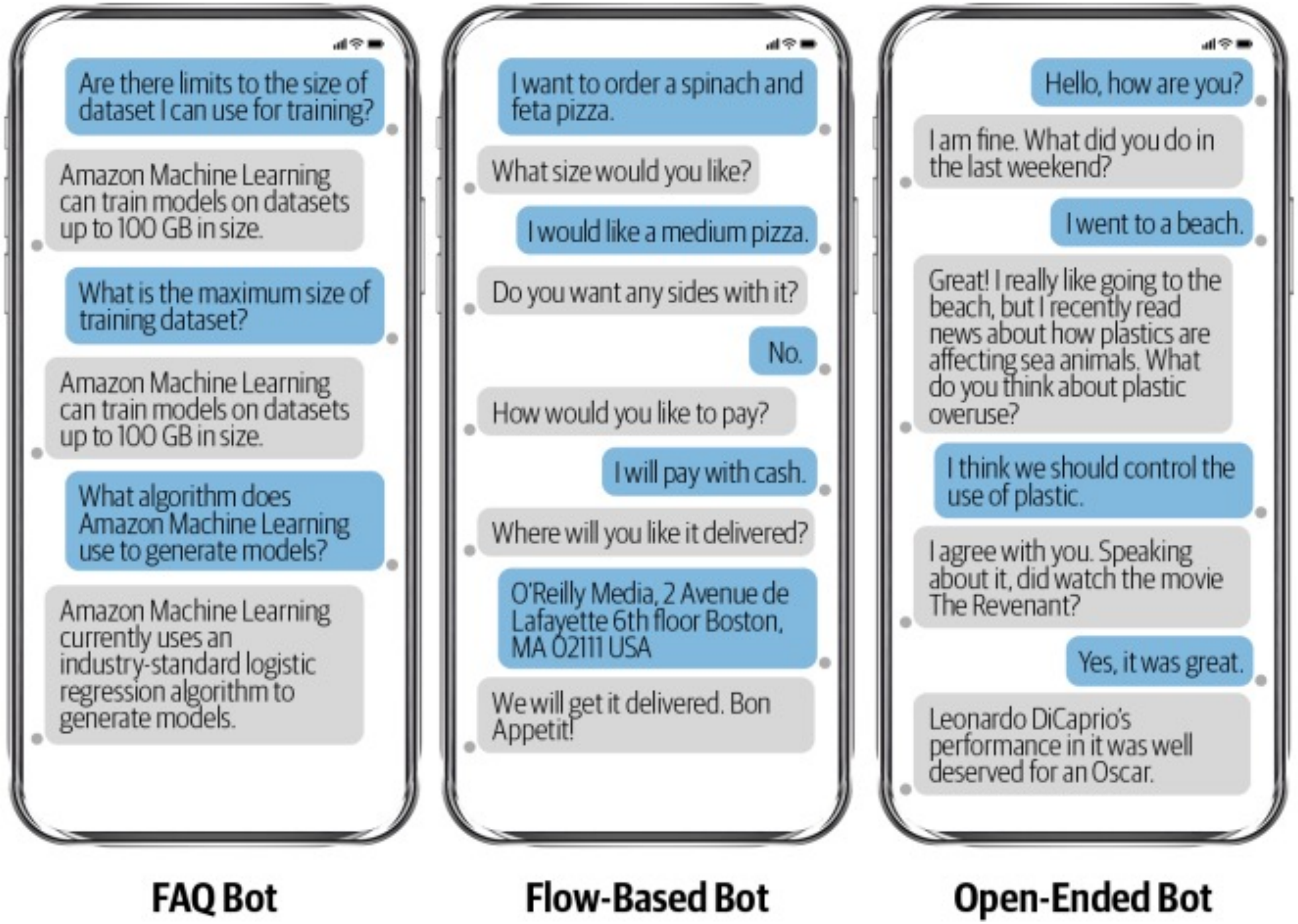
- **Exact answer or FAQ bot** is generally a **search-based system** where, given a question, it looks for correct answers and provides them to the user.
- A subset of Amazon Machine Learning Frequently Asked Questions:

Questions	Answer
What can I do with Amazon Machine Learning? How can I use Amazon Machine Learning? What can Amazon Machine Learning do?	You can use Amazon Machine Learning to create a wide variety of predictive applications. For example, you can use Amazon Machine Learning to help you build applications that flag suspicious transactions, detect fraudulent orders, forecast demand, etc.
What algorithm does Amazon Machine Learning use to generate models? How does Amazon Machine Learning build models?	Amazon Machine Learning currently uses an industry-standard logistic regression algorithm to generate models.
Are there limits to the size of the dataset I can use for training? What is the maximum size of training dataset?	Amazon Machine Learning can train models on datasets up to 100 GB in size.

A Taxonomy of Chatbots

- **Exact answer or FAQ bot** with limited conversations
- **Flow-based bot**
  - More complex than FAQ bots
  - The bot can understand and track this information throughout the conversation to successfully generate a response every time
- **Open-ended bot**
  - Mainly for entertainment, where the bot is supposed to converse with the user about various topics
  - The open-ended bot carries out a conversation without any pre-existing template or fixed question-answer pairs

Different Types of Chatbots



How to Build a Chatbot

- Dialog systems
  - The pipeline for building dialog systems Includes components such as speech recognition, natural language understanding, dialog manager, task manager, natural language generation, text-to-speech synthesis.
- Language models
  - Some techniques involved in the process are sequence-to-sequence model, deep reinforcement learning, human-in-the-loop.

Prompt Engineering

References & More Resources: An Online Guide

What Are Prompts?

- Prompts involve instructions and context passed to a language model to achieve a desired task
- Prompt engineering is the practice of developing and optimizing prompts to efficiently use language models (LMs) for a variety of applications

```
What is prompt engineering?

Prompt engineering is a process of creating a set of prompts,
or questions, that are used to guide the user toward a desired
outcome. It is an effective tool for designers to create user
experiences that are easy to use and intuitive. This method
is often used in interactive design and software development,
as it allows users to easily understand how to interact with a
system or product..
```

Why Prompt Engineering?

- Why learn prompt engineering?
  - Important for research, discoveries, and advancement
  - Helps to test and evaluate the limitations of LLMs
  - Enables all kinds of innovative applications on top of LLMs

ANTHROP\C

Prompt Engineer and Librarian

APPLY FOR THIS JOB

SAN FRANCISCO, CA / PRODUCT / FULL-TIME / HYBRID

Anthropic's mission is to create reliable, interpretable, and steerable AI systems. We want AI to b safe for our customers and for society as a whole.

Anthropic's AI technology is amongst the most capable and safe in the world. However, large language models are a new type of intelligence, and the art of instructing them in a way that delivers the best results is still in its infancy — it's a hybrid between programming, instructing, ar teaching. You will figure out the best methods of prompting our AI to accomplish a wide range o tasks, then document these methods to build up a library of tools and a set of tutorials that allov others to learn prompt engineering or simply find prompts that would be ideal for them.

Compensation and Benefits*

Anthropic's compensation package consists of three elements: salary, equity, and benefits. We are committed to pay fairness and aim for these three elements collectively to be highly competitive with market rates.

Salary - The expected salary range for this position is $250k - $335k.

Equity - Equity will be a major component of the total compensation for this position. We aim to offer higher-than-average equity compensation for a company of our size, and communicate equity amounts at the time of offer issuance.

Source: https://jobs.lever.co/Anthropic/e3cde481-d446-460f-b576-93cab67bd1ed

First Basic Prompt (Zero-shot Prompt)

Instruction

```
The sky is

blue

The sky is a beautiful blue color during the day. The
blue of the sky is created by the Earth's atmosphere
scattering the sun's light. The blue is a result of the
air molecules in the atmosphere reflecting the shorter
wavelength of blue light back to our eyes.
```

Response

Model: “text-davinci-003”
temperature: 0.7
top-p: 1

Elements of a Prompt

- A prompt is composed with the following components:
  - Instructions
  - Context
  - Input data
  - Output indicator

```
Classify the text into neutral, negative or positive
Text: I think the food was okay.
Sentiment:
```

Settings to Keep in Mind

- You can get very different results with prompts when using different settings
- One important setting is controlling how deterministic the model is when generating completion for prompts
  - Temperature and top_p are two important parameters
  - Generally, keep these low if you are looking for exact answers
  - Keep them high if you are looking for more diverse responses

Use Prompts for Various Tasks

- Traditional NLP Tasks
  - Text summarization
  - Text classification
  - Question answering, etc.
- Other Tasks:
  - Role playing
  - Code generation
  - Reasoning
  - Expanding
  - Open-ended chat, etc.

Text Summarization

Context

Instruction

```
Antibiotics are a type of medication used to treat
bacterial infections. They work by either killing the
bacteria or preventing them from reproducing, allowing
the body's immune system to fight off the infection.
Antibiotics are usually taken orally in the form of
pills, capsules, or liquid solutions, or sometimes
administered intravenously. They are not effective
against viral infections, and using them inappropriately
can lead to antibiotic resistance.

Explain the above in one sentence:


Antibiotics are medications used to treat bacterial
infections by either killing the bacteria or stopping
them from reproducing, but they are not effective against
viruses and overuse can lead to antibiotic resistance.
```

Text Classification

```
Classify the text into neutral, negative or positive.

Text: I think the food was okay.

Sentiment: Neutral
```

Question Answering

```
Answer the question based on the context below. Keep the
answer short and concise. Respond "Unsure about answer"
if not sure about the answer.

Context: Teplizumab traces its roots to a New Jersey drug
company called Ortho Pharmaceutical. There, scientists
generated an early version of the antibody, dubbed OKT3.
Originally sourced from mice, the molecule was able to
bind to the surface of T cells and limit their cell-
killing potential. In 1986, it was approved to help
prevent organ rejection after kidney transplants, making
it the first therapeutic antibody allowed for human use.

Question: What was OKT3 originally sourced from?

Answer: Mice.
```

Context obtained from: https://www.nature.com/articles/d41586-023-00400-x

Role Playing

```
The following is a conversation with an AI research
assistant. The assistant tone is technical and
scientific.

Human: Hello, who are you?
AI: Greeting! I am an AI research assistant. How can I
help you today?
Human: Can you tell me about the creation of blackholes?
AI: Sure! Black holes are regions of spacetime where the
gravitational force is so strong that nothing, not even
light, can escape from it. They are created when a very
massive star dies and its core collapses in on itself,
forming a singularity of infinite density. The intense
gravity of the singularity pulls in all the matter and
radiation around it, creating the black hole.
```

Code Generation

```
"""
Table departments, columns = [DepartmentId,
DepartmentName]
Table students, columns = [DepartmentId, StudentId,
StudentName]
Create a MySQL query for all students in the Computer
Science Department
"""

SELECT StudentId, StudentName
FROM students
WHERE DepartmentId IN (SELECT DepartmentId FROM
departments WHERE DepartmentName = 'Computer Science');
```

Reasoning

```
The odd numbers in this group add up to an even number: 15,
32, 5, 13, 82, 7, 1.

Solve by breaking the problem into steps. First, identify
the odd numbers, add them, and indicate whether the result
is odd or even.

Odd numbers: 15, 5, 13, 7, 1
Sum: 41
41 is an odd number.
```

Advanced Prompting Techniques

Prompt Engineering Techniques

- Many advanced prompting techniques have been designed to improve performance on complex tasks
  - Few-shot prompts
  - Chain-of-thought (CoT) prompting
  - Self-Consistency

Few-shot Prompts

- Few-shot prompting allows us to provide exemplars in prompts to steer the model towards better performance

```
The odd numbers in this group add up to an even number: 4,
8, 9, 15, 12, 2, 1
A: The answer is False.

The odd numbers in this group add up to an even number: 17,
10, 19, 4, 8, 12, 24.
A: The answer is True.

The odd numbers in this group add up to an even number: 16,
11, 14, 4, 8, 13, 24.
A: The answer is True.

The odd numbers in this group add up to an even number: 17,
9, 10, 12, 13, 4, 2.
A: The answer is False.

The odd numbers in this group add up to an even number: 15,
32, 5, 13, 82, 7, 1.
A: The answer is True.
```

Chain-of-Thought (CoT) Prompting

- Prompting can be further improved by instructing the model to reason about intermediate steps of the task when responding
  - This is very useful for tasks that requiring reasoning
  - You can combine it with few-shot prompting to get better results
  - You can also do zero-shot CoT where exemplars are not available

```
The odd numbers in this group add up to an even number: 4,
8, 9, 15, 12, 2, 1
A: Adding all the odd numbers (9, 15, 1) gives 25. The
answer is False.
The odd numbers in this group add up to an even number: 15,
32, 5, 13, 82, 7, 1.
A: Adding all the odd numbers (15, 5, 13, 7, 1) gives 41.
The answer is False.
```

Source: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Zero-Shot CoT

- Involves adding "Let's think step by step" to the original prompt

```
I went to the market and bought 10 apples. I gave 2 apples
to the neighbor and 2 to the repairman. I then went and
bought 5 more apples and ate 1. How many apples did I
remain with?

11 apples
```

```
I went to the market and bought 10 apples. I gave 2 apples
to the neighbor and 2 to the repairman. I then went and
bought 5 more apples and ate 1. How many apples did I
remain with?
Let's think step by step.

First, you started with 10 apples.
You gave away 2 apples to the neighbor and 2 to the
repairman, so you had 6 apples left.
Then you bought 5 more apples, so now you had 11 apples.
Finally, you ate 1 apple, so you would remain with 10
apples.
```

Source: [Large Language Models are Zero-Shot Reasoners]

Self-Consistency

- Self-Consistency aims to improve on the naive greedy decoding used in chain-of-thought prompting
- The idea is to sample multiple, diverse reasoning paths through few-shot CoT, and use the generations to select the most consistent answer.
- This helps to boost the performance of CoT prompting on tasks involving arithmetic and commonsense reasoning

```
When I was 6 my sister was half my age. Now
I'm 70 how old is my sister?

35
```

Many examples were generated but the model kept responding 35 as the answer

Source: Self-Consistency Improves Chain of Thought Reasoning in Language Models

Self-Consistency Example

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: We start with 15 trees. Later we have 21 trees. The difference must be the number of trees they planted. So, they must have planted 21 - 15 = 6 trees. The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.

...

Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: She bought 5 bagels for $3 each. This means she spent 5

Q: When I was 6 my sister was half my age. Now I'm 70 how old is my sister?
A:

Output 1

When I was 6 my sister was half my age, so she was 3. Now I am 70, so she is 70 - 3 = 67. The answer is 67.

Output 2

When the narrator was 6, his sister was half his age, which is 3. Now that the narrator is 70, his sister would be 70 - 3 = 67 years old. The answer is 67.

Output 3

When the narrator was 6, his sister was half his age, which is 3. Now that the narrator is 70, his sister would be 70 - 3 = 67 years old. The answer is 67.

Summary

- LLMs have a lot of limitations. One prominent limitation is hallucination (i.e., making statements that sound plausible but are not true).
- Prompting is an iterative process
  - Try the initial prompt
  - Analyze where the results do not give what you want
  - Clarify instructions, give the model more time to think, or try alternative prompting methods
- More prompt engineering techniques keep coming out …

Tools & Applications

Tools & IDEs

- There are many tools, libraries, and platforms with different capabilities and functionalities
- Capabilities include:
  - Developing and experimenting with prompts
  - Evaluating prompts
  - Versioning and deploying prompts

Dyno
Prompt Engineering IDE

LangChain

DUST

PROMPTABLE

More tools here: https://github.com/dair-ai/Prompt-Engineering-Guide#tools--libraries

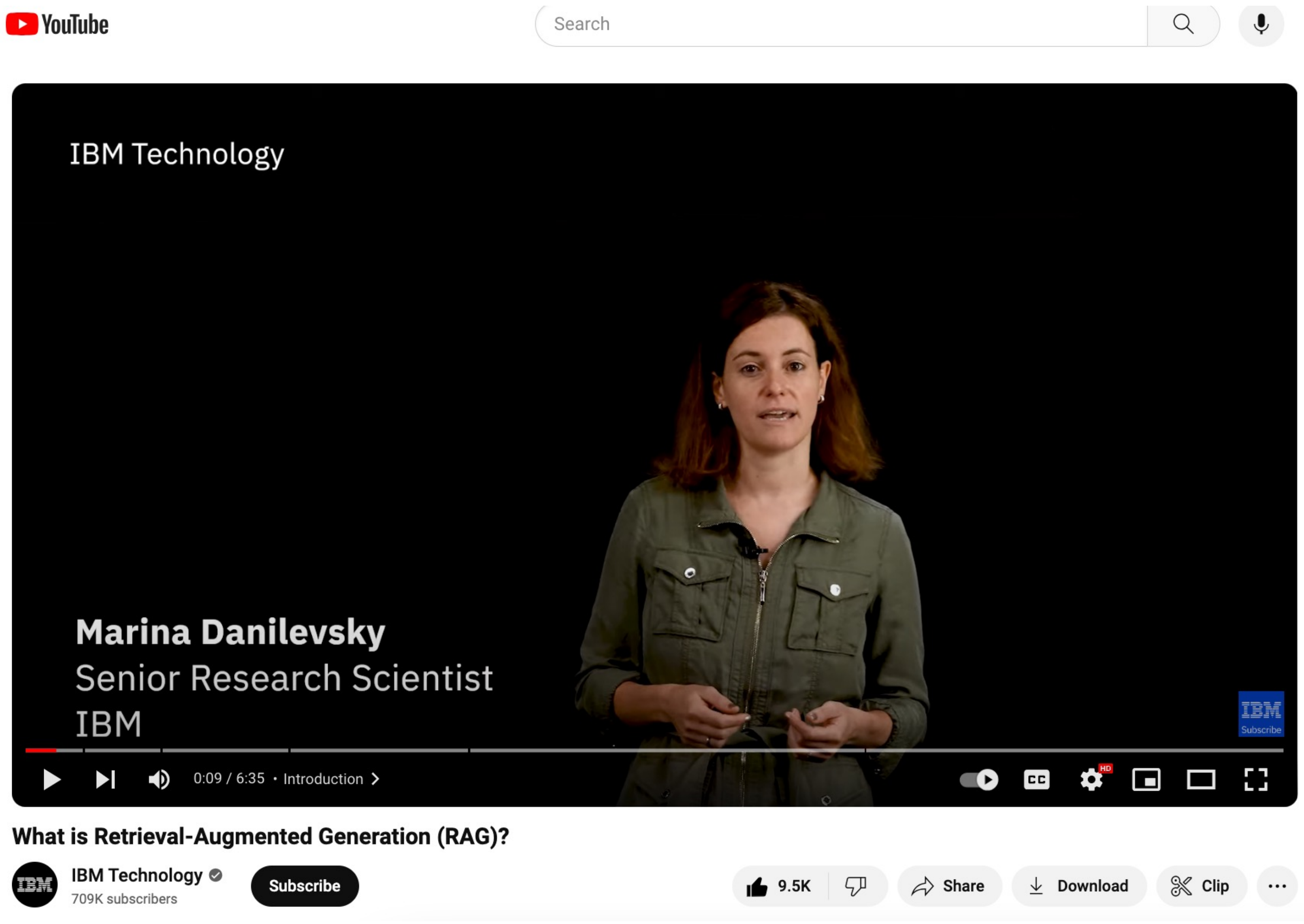
Applications - LLMs with External Tools

- The generative capabilities of LLMs can be combined with an external tool to solve complex problems.
- The components you need:
  - An agent powered by LLM to determine which actions to take
  - A tool used by the agent to interact with the world (e.g., search API, Wolfram, Python REPL, database lookup)

Applications - Data-Augmented Generation

- For many real-world applications there is a need to augment the generation of a model by incorporating external data (e.g., RAG)
- Steps involved:
  - Fetching relevant data
  - Augmenting prompts with the retrieved data as context
- External data can include:
  - APIs
  - Databases
  - User provided data

Retrieval Augmented Generation



Links

Take 10 minutes break…

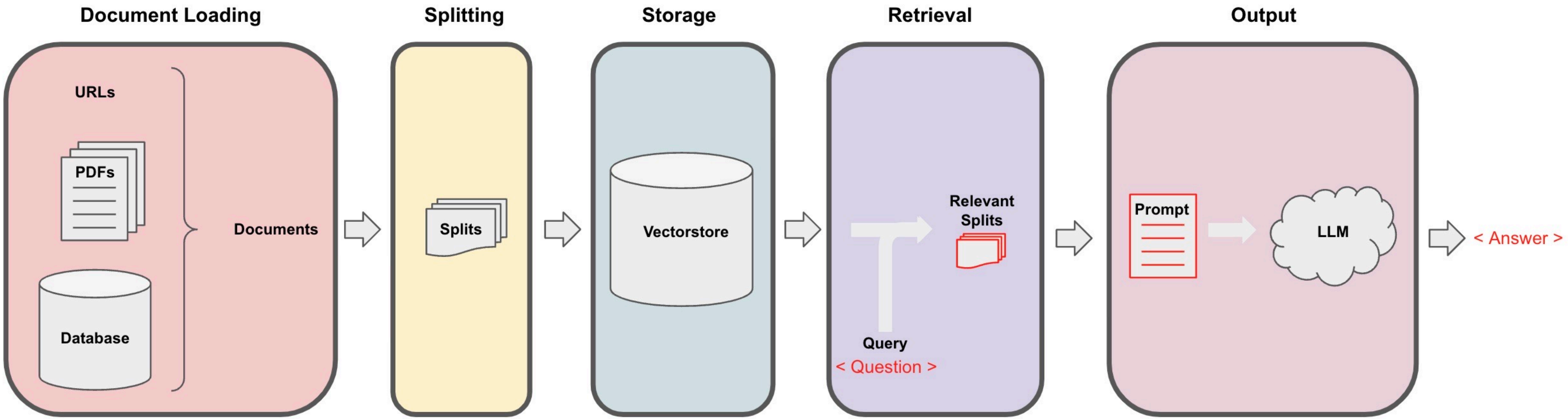
Retrieval Augmented Generation (RAG)

Why RAG?

- RAG is based on research produced by the Meta team in 2020 to advance the natural language processing capabilities of LLMs.
- Why RAG?
  - **LLMs are static**. They lack up-to-date information. It is challenging and costly to update the gigantic training dataset.
  - **LLMs lack domain-specific knowledge**. They are trained for generalized tasks, meaning the training data does not include internal and proprietary data.
  - **LLMs function as "black box"**. It is not easy to understand which sources an LLM was considered when they arrived at their conclusions.
  - **LLMs are inefficient and costly to produce**. Few organizations have the financial and human resources to produce and deploy foundation models.

References

RAG Framework



(1) Document Loading

- Loaders deal with the specifics of accessing and converting data. It returns a list of ‘Document’ objects.
- Code example

```
from langchain.document_loaders import PyPDFLoader

# Load PDF
loaders = [
    # Duplicate documents on purpose - messy data
    PyPDFLoader("/content/LLM check.pdf"),
    PyPDFLoader("/content/Large Language Models.pdf"),
]
docs = []
for loader in loaders:
    docs.extend(loader.load())
```

(2) Document Splitting

- Langchain.text_splitter
  - CharacterTextSplitter() – Implementation of splitting text that looks as characters
  - MarkdownHeaderTextSplitter() – Implementation of splitting markdown files based on specified headers.
  - TokenTextSplitter() – Implementation of splitting text that looks at tokens.
  - RecursiveCharacterTextSplitter() – Implementation of splitting text that looks at characters. Recursively tries to split by different characters to find one that works.
  - NLTKTextSplitter() – Implementation of splitting text that looks at sentence using NLTK
  - SpacyTextSplitter() – Implementation of splitting text that looks at sentences using Spacy

(2) Document Splitting

- Code example

```
# Split
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size = 1500,
    chunk_overlap = 150
)
```

```
splits = text_splitter.split_documents(docs)
```

(3) Vectorstores and Embedding

- Embedding vector captures content/meaning
- Text with similar content will have similar vectors

```
from langchain.vectorstores import Chroma
from langchain.embeddings.openai import OpenAIEmbeddings
persist_directory = 'docs/chroma/'
embedding = OpenAIEmbeddings()
vectordb = Chroma(persist_directory=persist_directory, embedding_function=embedding)
```

(4) Retrieval

- Accessing/indexing the data in the vector store
  - Basic semantic similarity
  - Maximum marginal relevance
    - You want to retrieve a diverse set of information
  - Including metadata
- LLM aided retrieval
  - One is SelfQuery, where we use an LLM to convert the user question into a query
- Compression, etc.

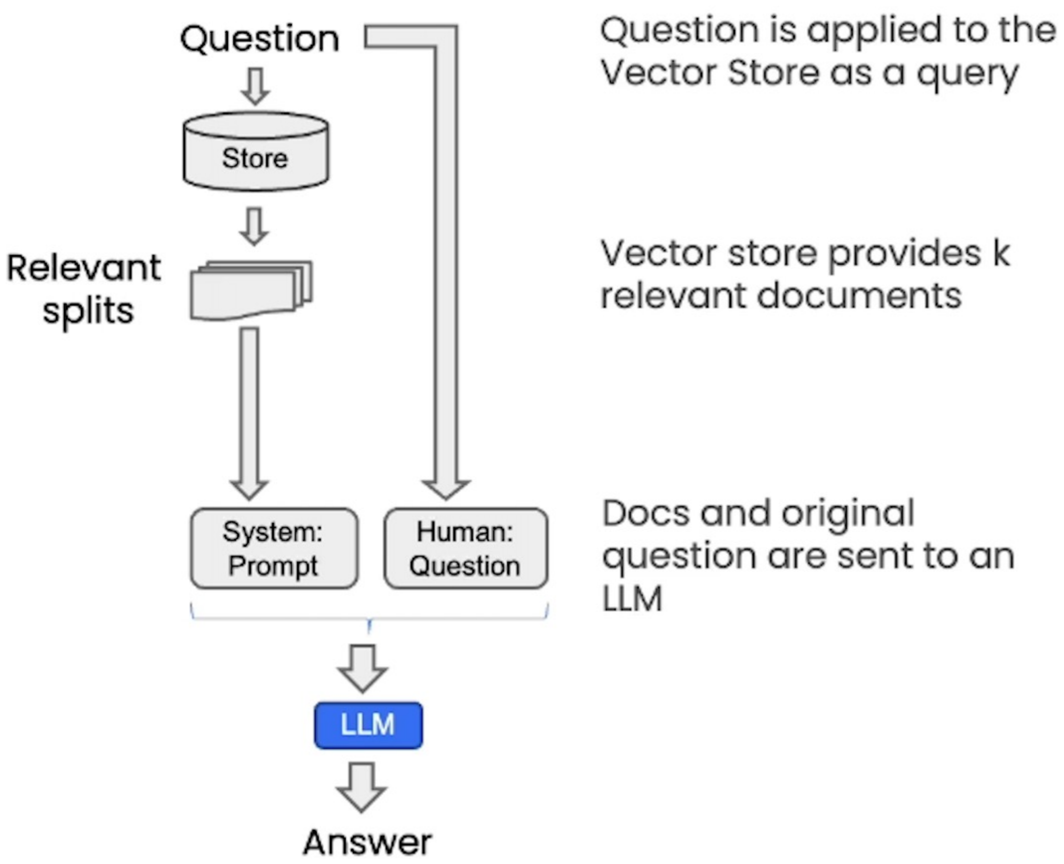
(4) Retrieval

- Code example

```
question = "What is LLM?"
docs = vectordb.similarity_search(question,k=3)
len(docs)
```

(5) Output – Question Answering

- Multiple relevant documents have been retrieved from the vector store
- Potentially compress the relevant splits to fit into the LLM context
- Send the information along with our questions to an LLM to select and form an answer
- When more documents retrieved and they couldn't fit the context
  - Three additional methods: Map_reduced, refine and map_rerank.



(5) Output – Question Answering

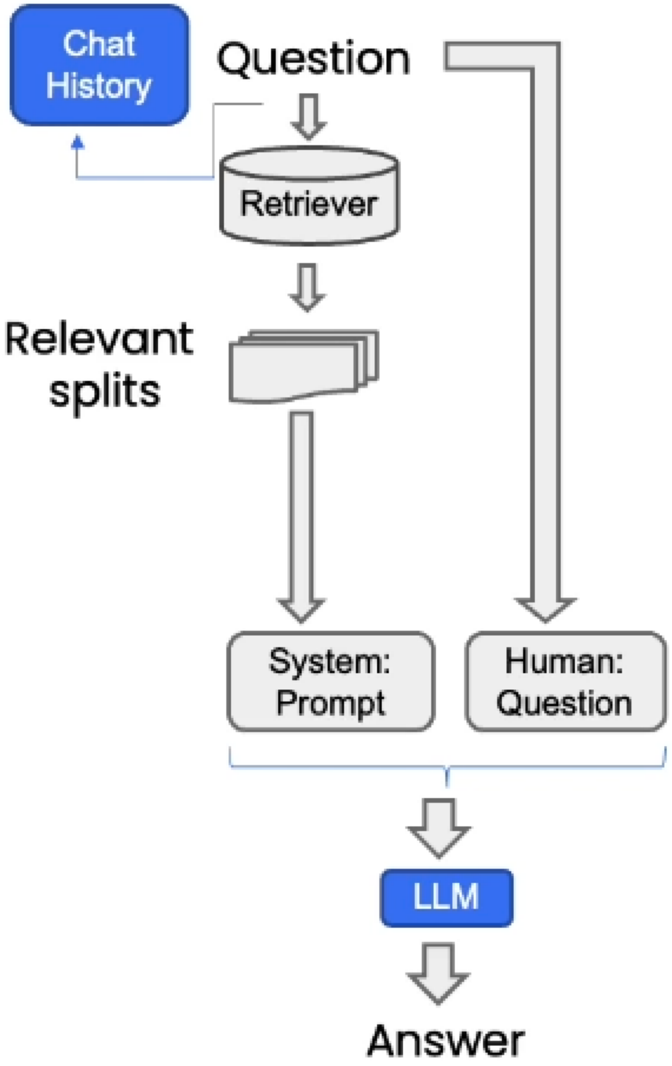
- Code example

```
# Build prompt
from langchain.prompts import PromptTemplate
template = """Use the following pieces of context to answer the question at the end. If you do
{context}
Question: {question}
Helpful Answer:"""
QA_CHAIN_PROMPT = PromptTemplate(input_variables=["context", "question"],template=template,)

# Run chain
from langchain.chains import RetrievalQA
question = "Is probability a class topic?"
qa_chain = RetrievalQA.from_chain_type(llm,
                                       retriever=vectordb.as_retriever(),
                                       return_source_documents=True,
                                       chain_type_kwargs={"prompt": QA_CHAIN_PROMPT})


result = qa_chain({"query": question})
result["result"]
```

(6) Output – Chatbot

```
qa = ConversationalRetrievalChain.from_llm(ChatOpenAI(temperature=0),
vectorstore.as_retriever(), memory=memory)
```

- Compared to retrieval QA chain, the ConversationalRetrievalChain adds a step that adds the history and question and then condenses it into a stand-alone question to pass to the vector store to look up relevant documents.



(6) Output – Chatbot

- Code example

```
from langchain.chains import ConversationalRetrievalChain
retriever=vectordb.as_retriever()
qa = ConversationalRetrievalChain.from_llm(
    llm,
    retriever=retriever,
    memory=memory
)
```

```
question = "Is probability a class topic?"
result = qa({"question": question})
```

```
result['answer']
```

References & More Resources

DeepLearning.AI

LangChain Chat with Your Data

Taught in English

Instructor: Harrison Chase

Links

LangChain Chat with Your Data

- Introduction
- Document Loading
- Document Splitting
- Vectorstores and Embedding
- Retrieval
- Question Answering
- Chat
- Conclusion

Practice using LLMs