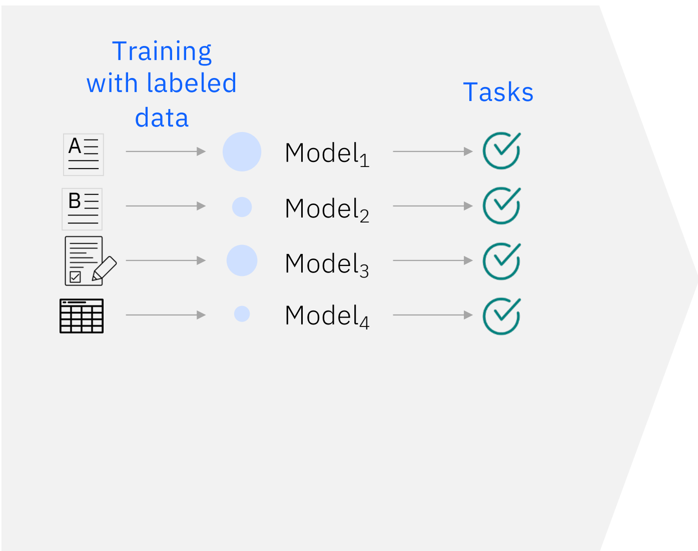
Introduction to key concepts

Agenda

- FM, Prompt Engineering, Prompt Tuning and Fine Tuning
- Use Gen-AI API from python code in Notebooks or VS Code
- Leverage LangChain
- Retrieval Augmented Generation
- Introduction to Embeddings and VectorDB
- Position IBM Watson & other products as relevant

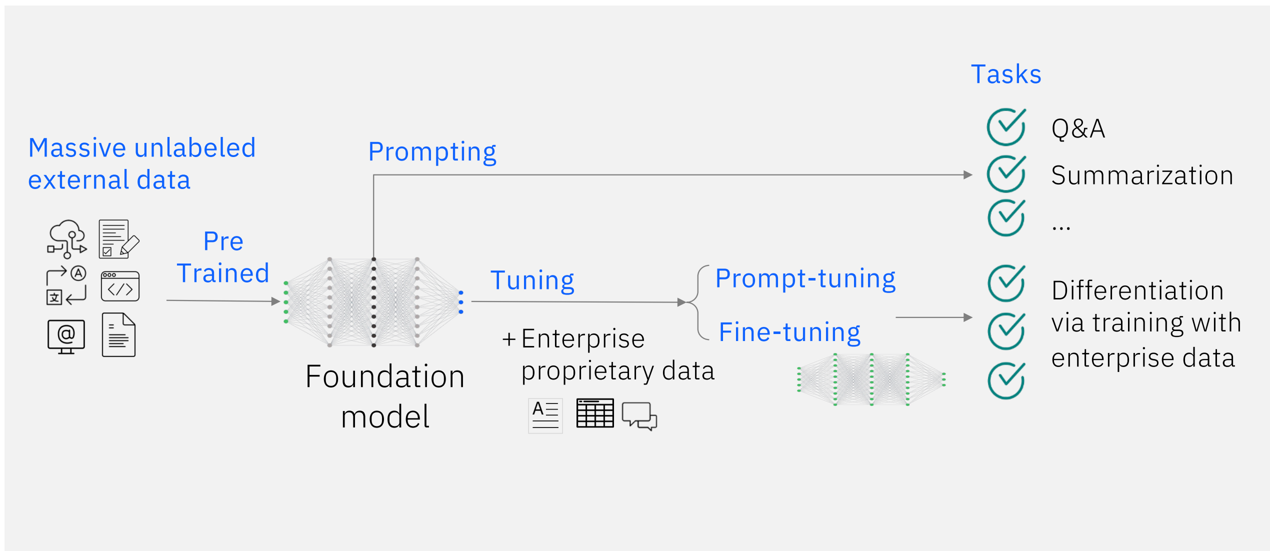
Foundational models enable a new paradigm of data-efficient AI development – generative AI

Traditional AI models



- Individual siloed models
- Require task specific training
- Lots of human supervised training

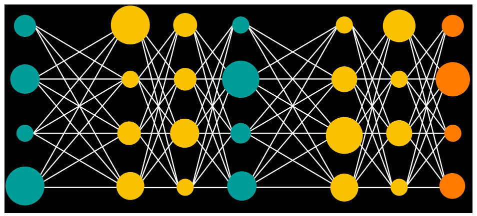
Foundation Models



- Rapid adaptation to multiple tasks with small amounts of task-specific data
- Pre-trained unsupervised learning

Foundation models are ...

Self-supervised training



Foundation model

Pre-trained

On unlabeled datasets of different modalities (e.g., language, time-series, tabular)

Self-learning

Systems that leverage self-supervised learning

Multiple applications

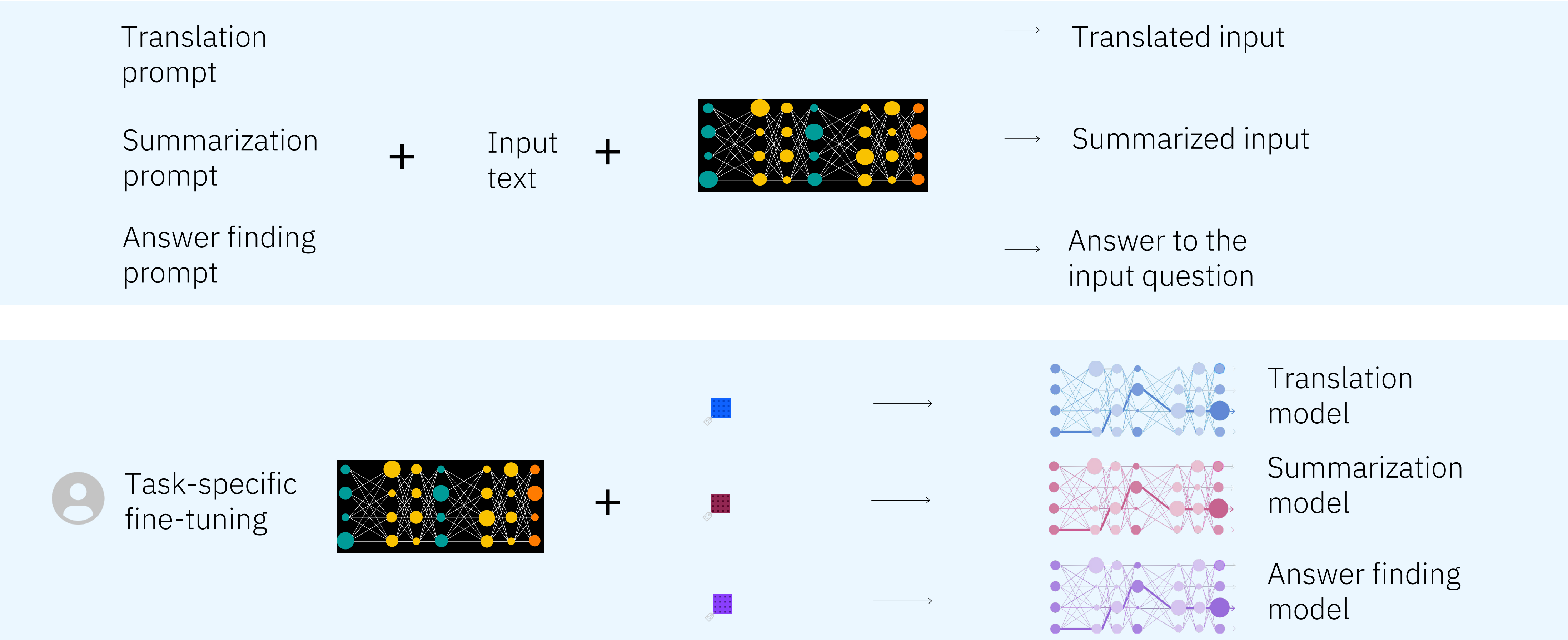
Able to learn generalizable and adaptable data representations that can be effectively used in a variety of domains and tasks (code generation, question answering, sentiment analysis)

Large language models

A type of foundation model trained withy language-related data

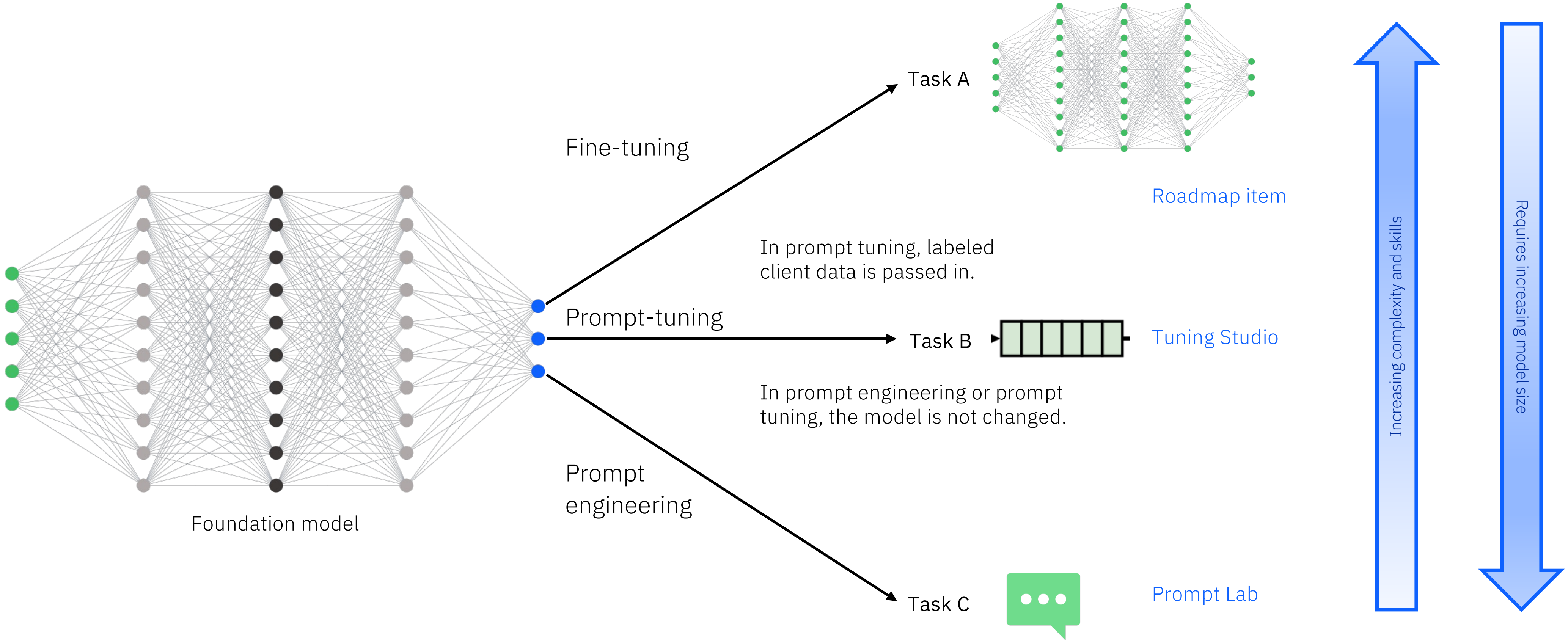
ChatGPT is based on a large language model

Foundation models: generalizable and adaptable



Rapid adaptation to multiple tasks with small amounts of task-specific data

Fine-tuning requires labeled data and more resources to tune the model. When a model is fine-tuned, some of the weights are modified and clients get a private instance of the model.



When to tune a model?

Always start with prompt engineering the largest LLM suited for your task.

This should provide some indication that the task is suited to be addressed by LLMs. It is also helpful to experiment with different labelled examples and understand which prompt formats work best on the target task.

Decision to tune can be motivated by:

(1) Achieving better performance with base model

By tuning the model on a large number of labelled examples, we can enhance the model performance compared to prompt engineering alone.

(2) Reducing costs at scale by deploying smaller model

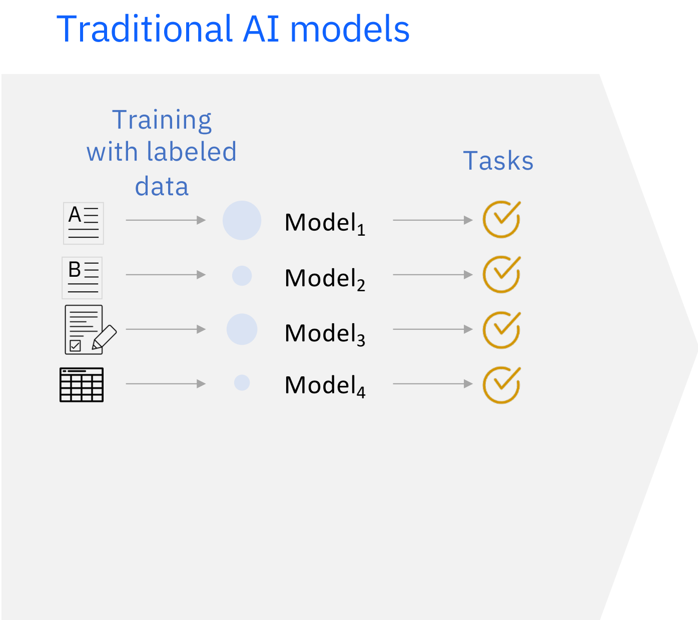
By tuning a smaller base model to perform similarly to a significantly bigger model, we can reduce costs when model deployed at scale.

Cost of labelled data acquisition is an important consideration in the decision process.

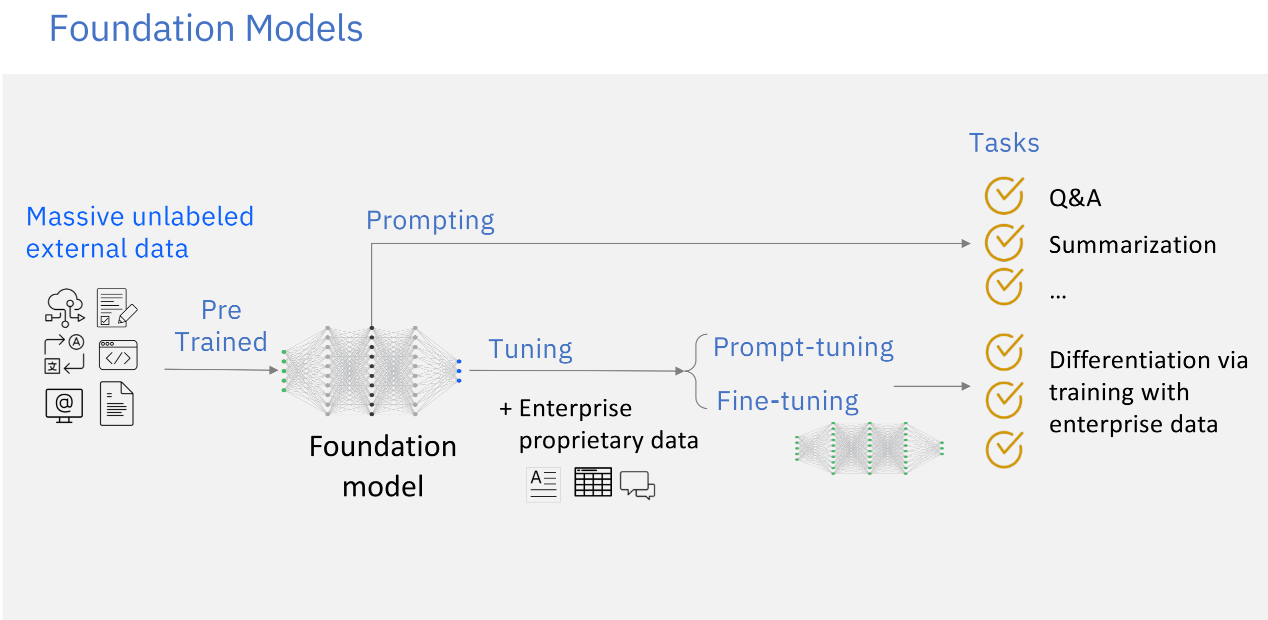
Suggested workflow	Step 1	Step 2	Step 3
Stage	**Initial PoC**	**Pilot deployment**	**Deployment at scale**
Goal	*Prove the use case with minimal effort*	*Reduce costs as permitted within PoC duration*	*Maximize ROI*
Recommendation	Use the largest model and minimal labeled data	Prompt engineer or prompt-tune a medium-size model.	Consider using additional data gathered to fine-tune / prompt-tune a small model.
	Create labeled test dataset to measure model accuracy	You may need to gather additional labeled data	Deploy the tuned model
Inference costs	$$$	$$	$

Note: *fine-tuning a model requires creating a copy of the model specific to the user.*
The cost of hosting this model may impact the ROI analysis compared to prompt tuning.

Foundational models enable a new paradigm of data-efficient AI development – generative AI



- Individual siloed models
- Require task specific training
- Lots of human supervised training



- Rapid adaptation to multiple tasks with small amounts of task-specific data
- Pre-trained unsupervised learning

IBM partnership with open-source models provider

HUGGING FACE

- IBM **watsonx.ai** clients have access to the latest and greatest open-source foundation models from Hugging Face.
- The IBM and Hugging Face partnership demonstrates a joint commitment to deliver an open ecosystem to clients, allowing them to find the best foundation models for their business needs.

Most common generative AI tasks implemented today

Summarization

Transform text with domain-specific content into personalized overviews that capture key points.

Conversation summaries, insurance coverage, meeting transcripts, contract information

Classification

Read and classify written input with as few as zero examples.

Sorting of customer complaints, threat and vulnerability classification, sentiment analysis, customer segmentation

Generation

Generate text content for a specific purpose.

Marketing campaigns, job descriptions, blog posts and articles, email drafting support

Extraction

Analyze and extract essential information from unstructured text.

Medical diagnosis support, user research findings

Question-answering

Create a question-answering feature grounded on specific content.

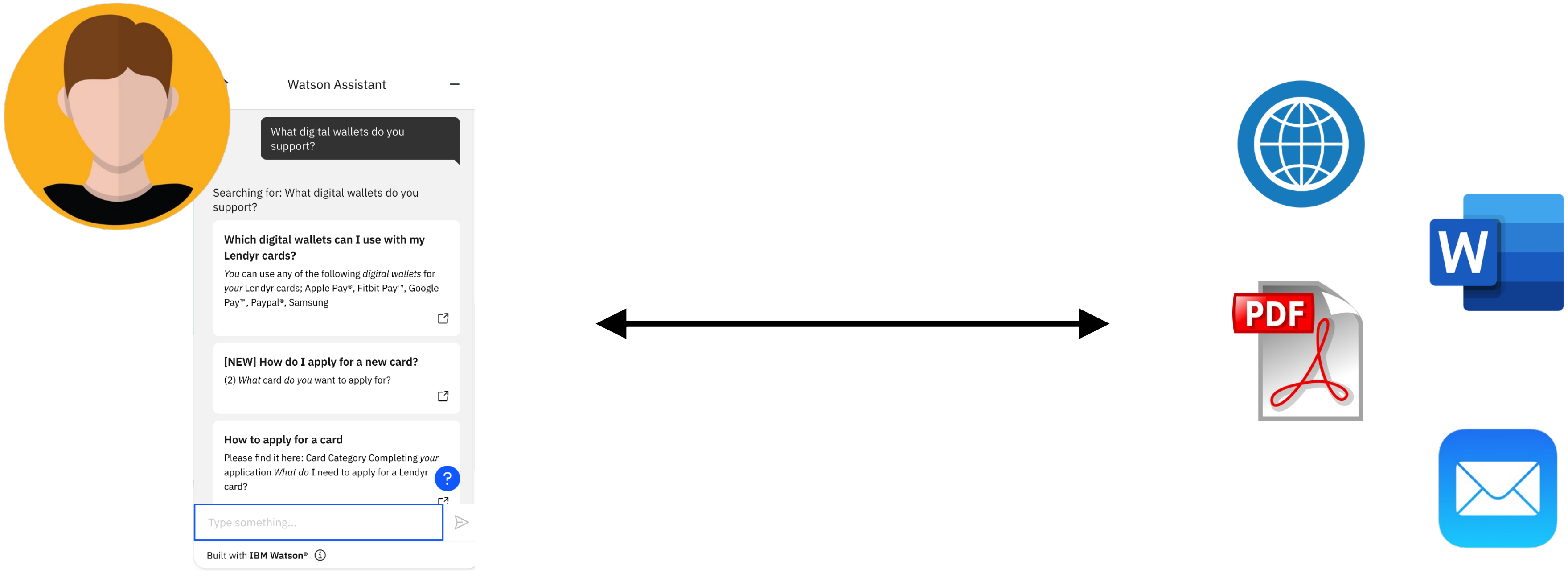
Build a product specific Q&A resource for customer service agents.

LangChain for LLMs

- LangChain is an open-source framework designed to simplify creating applications using LLMs.
- Models
- Prompt Templates
- Parsers
- Chains
- Question Answer

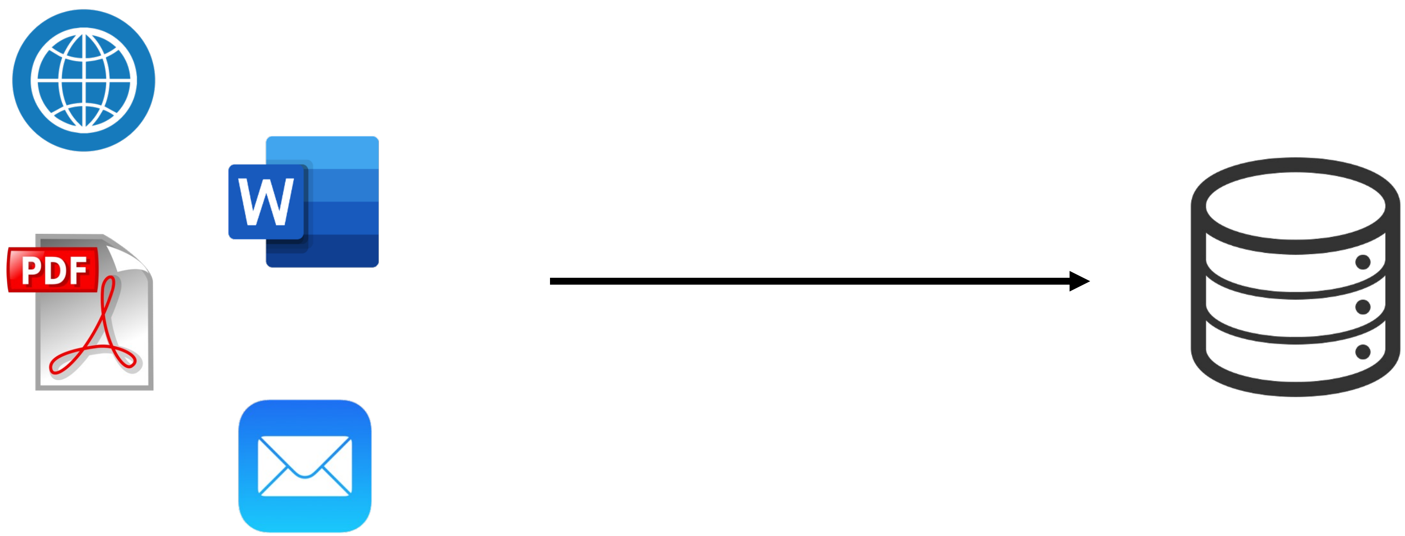
Retrieval Augmented Search - Overview

Conversational search – Q&A for documents



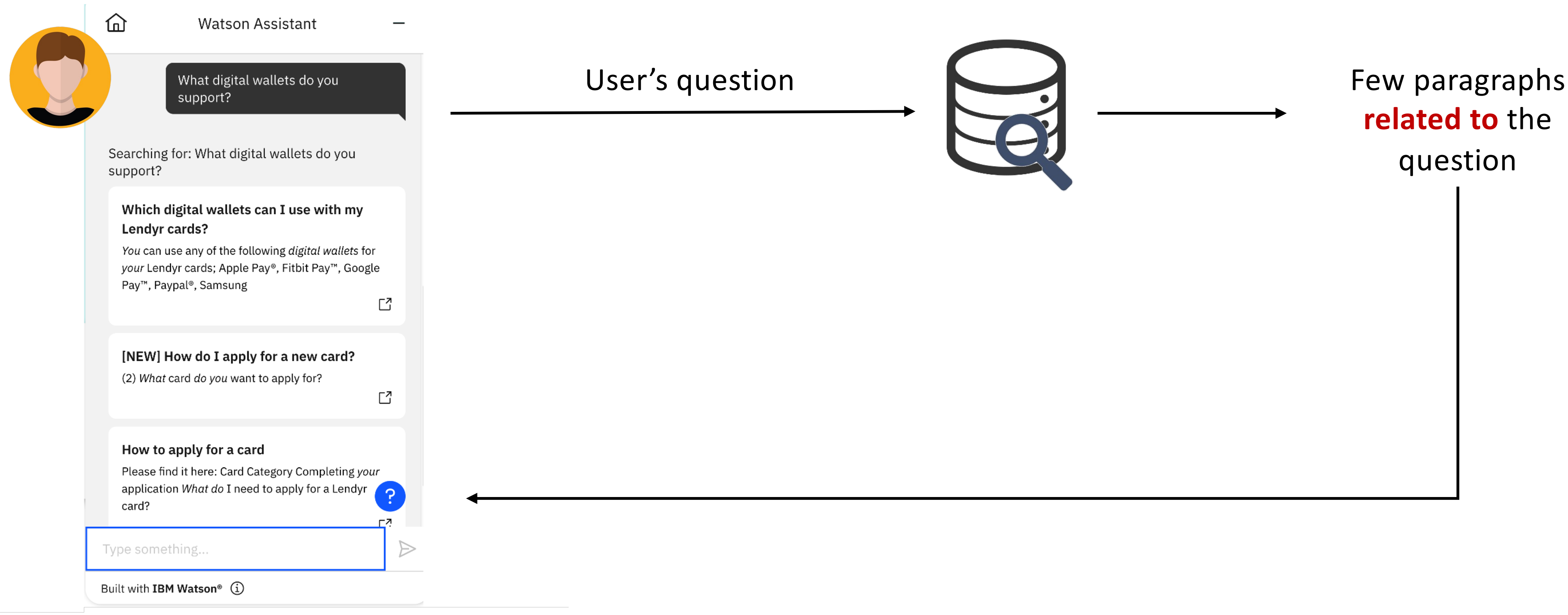
Phase 1

Prepare the data



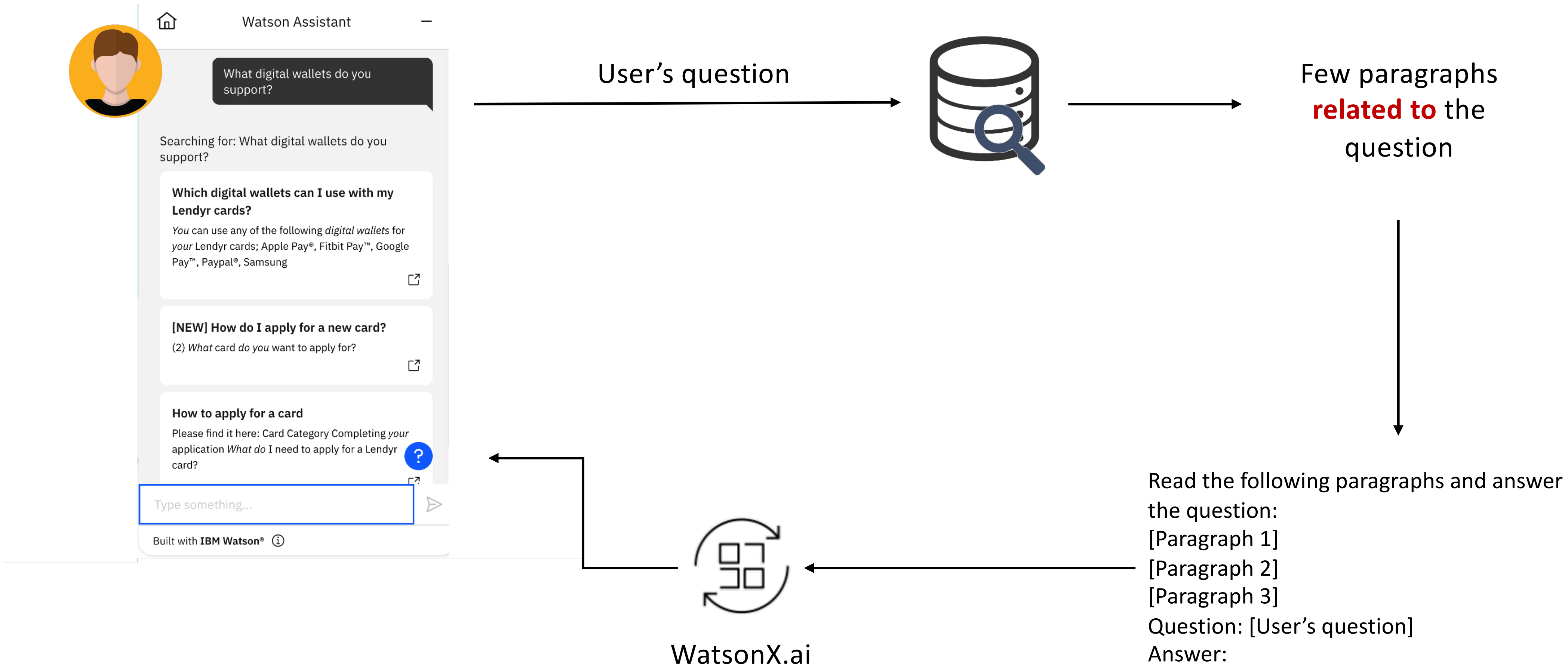
Phase 2

Query the data



Phase 2 (new steps based on LLMs)

Query the data

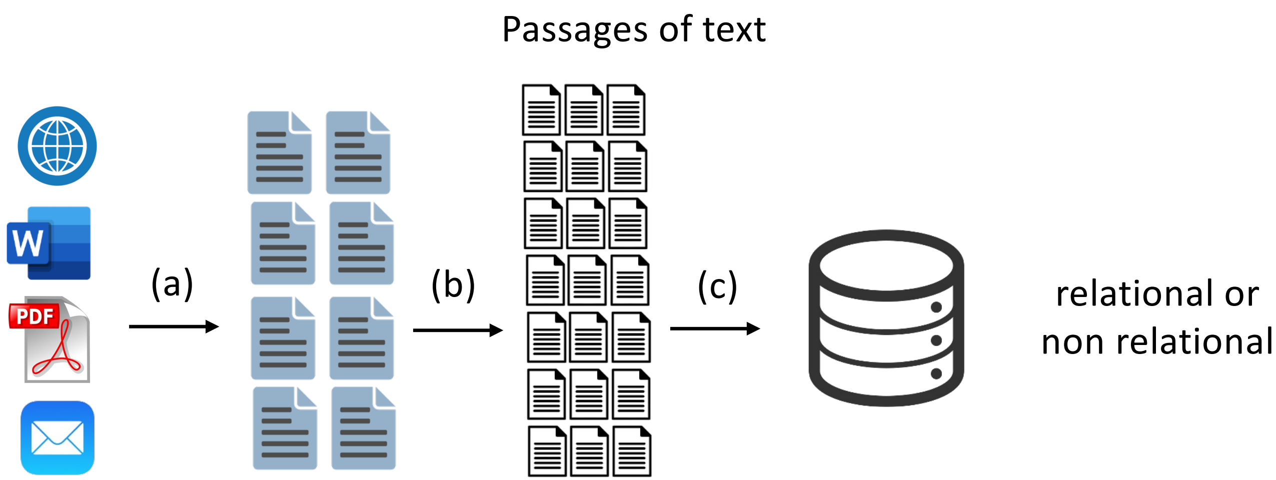


Phase 1: The “traditional” way

Phase 1

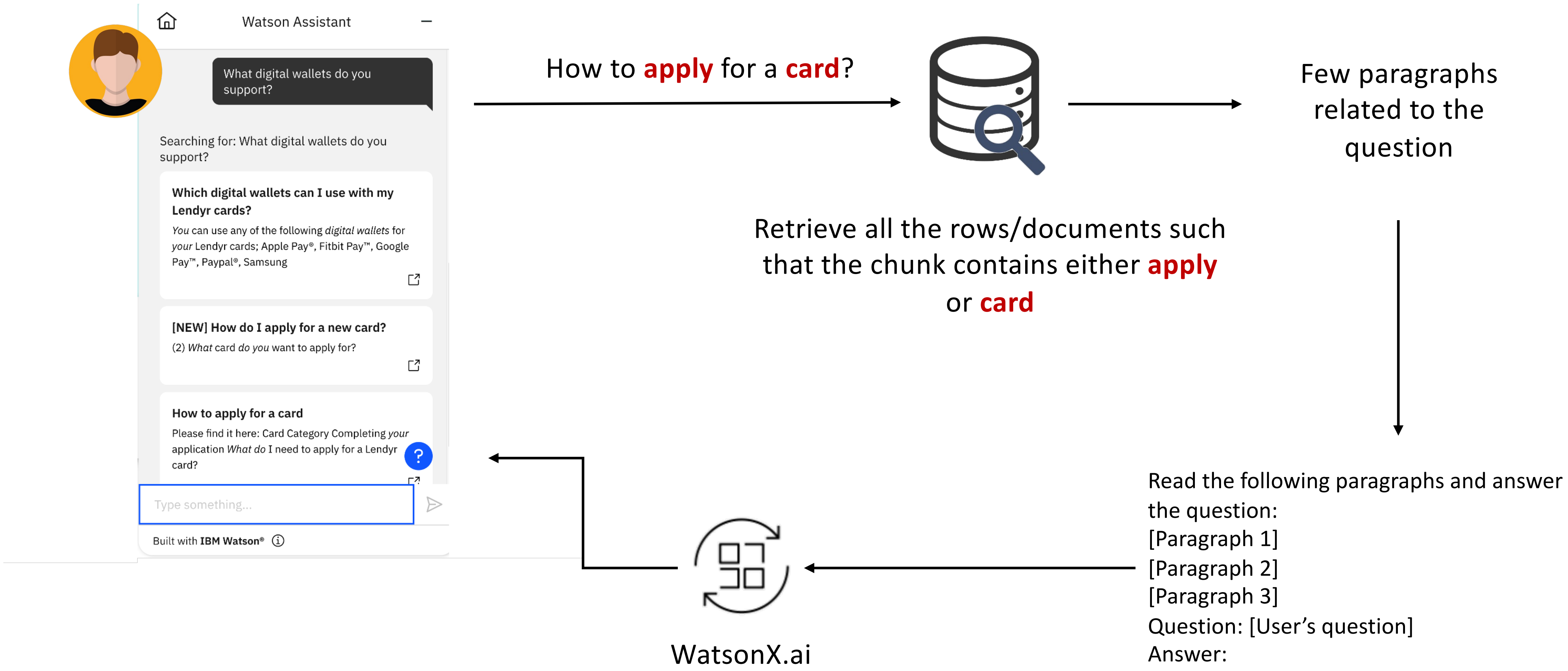
Ingest your data

(a) Original files to documents
(b) Documents to chunks
(c) Chunks to database



Phase 2: Syntactic search

Query the data

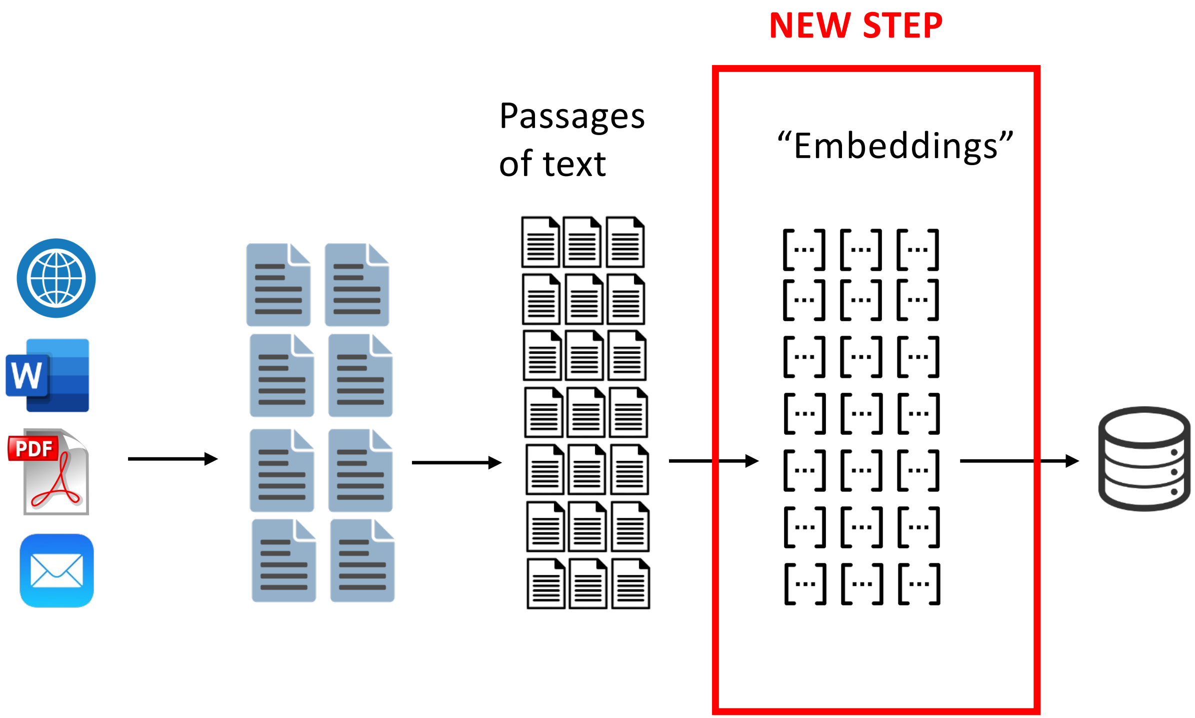


Phase 1: The “embeddings” way

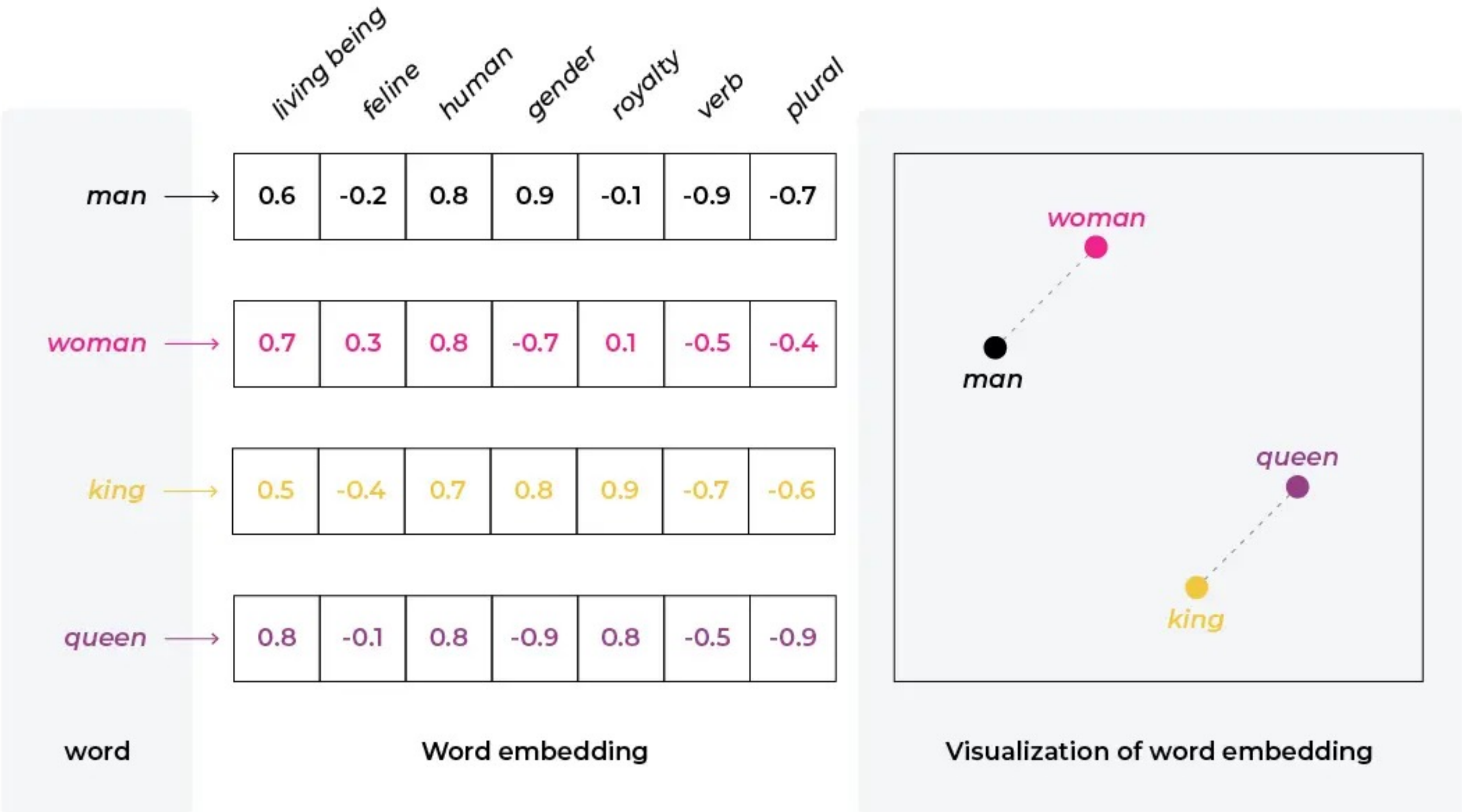
Phase 1

Ingest your data

(a) Original files to documents
(b) Documents to chunks
(c) Chunks to embeddings
(d) Embeddings to vector store

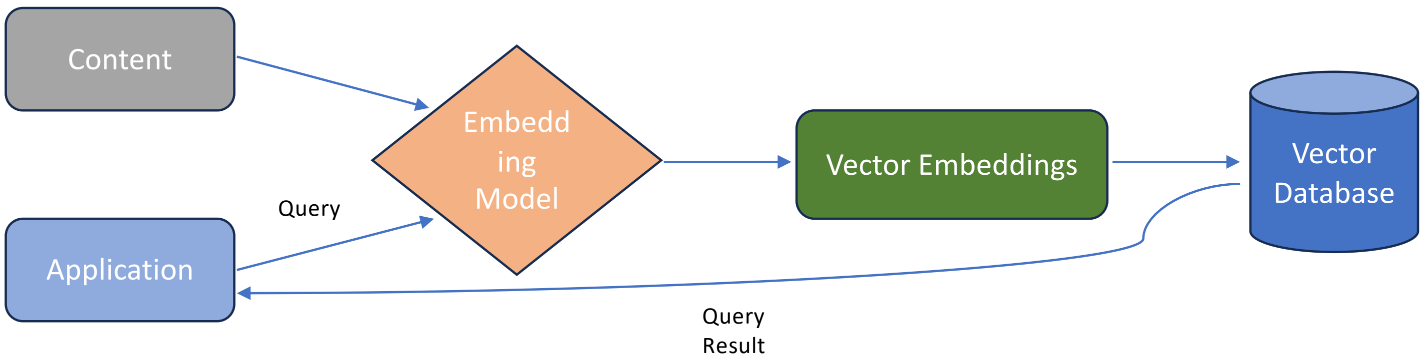


Phase 1: The "embeddings" way



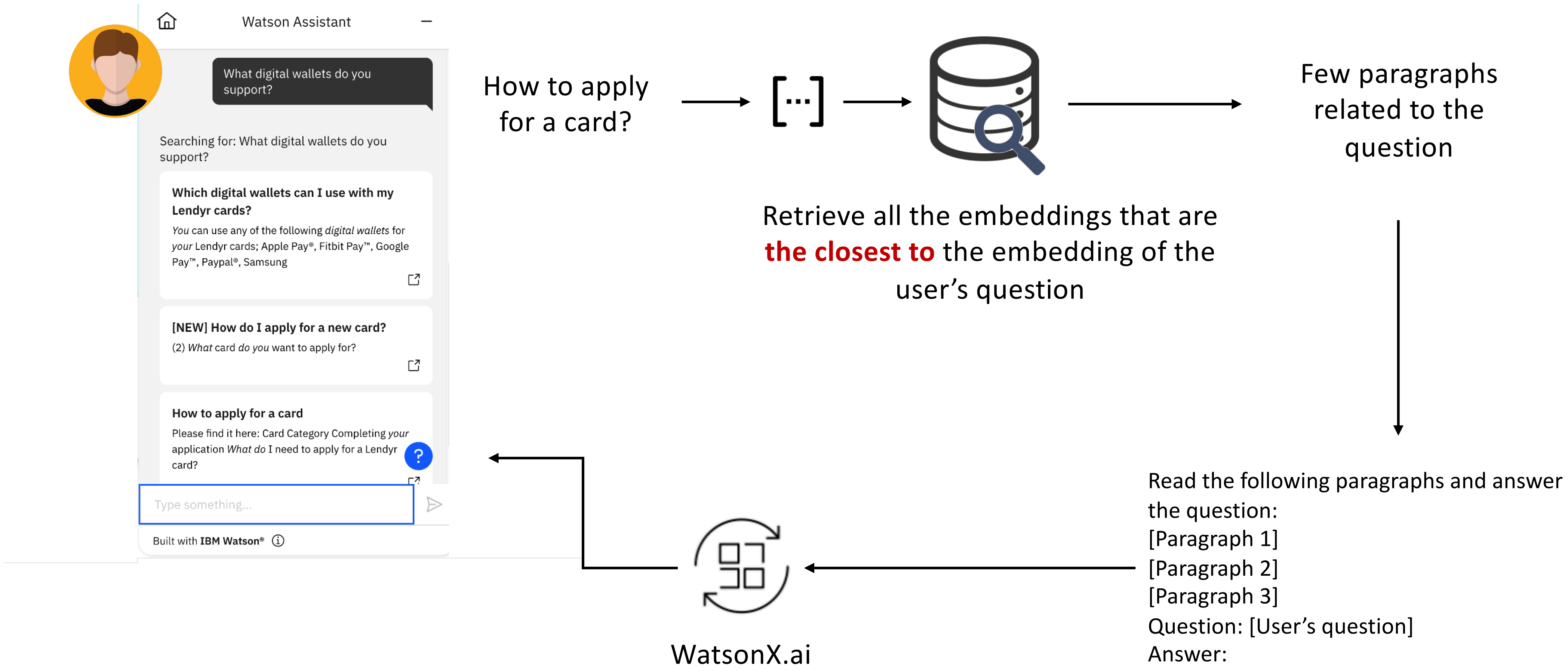
Phase 1: The "embeddings" way – VectorDB

- A vector database is a special type of database that can store high-dimensional vectors which are mathematical representations of the features.
- This data is nothing, but a vector created through embeddings
- Use cases
  - Recommendation systems
  - Anomaly detections
  - NLP



Phase 2: Semantic search

Query the data



Syntactic vs. Semantic search

Why semantic is a “better” way of searching for information

The user expresses himself in his/her own way, whereas the documents usually use “specialized” terms.

Examples:

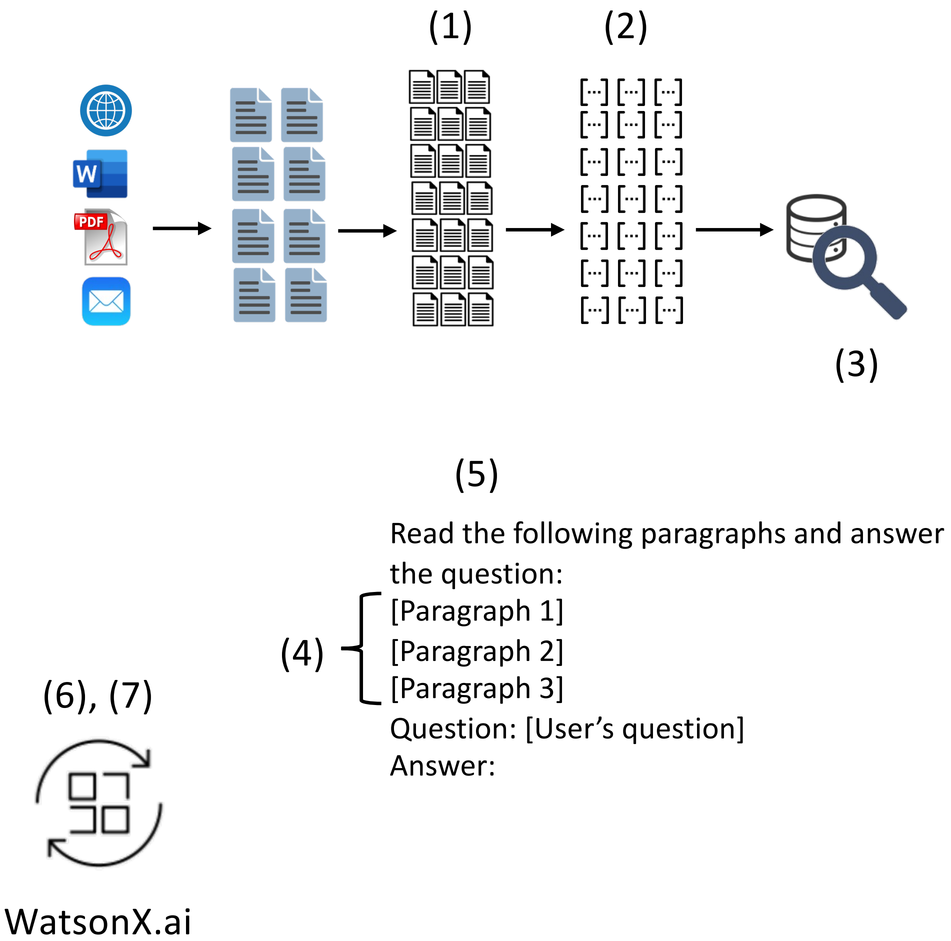
	
Paid leave of absence (IBM HR documents)	Day off
Corporate assets (Bank’s code of ethics)	Company’s laptop
Revenue, profits, benefits (10K form)	Money

How to improve the accuracy?

Optimize the config at each and every step:

1. Length of the chunks of texts
2. Choice of embeddings library
3. Distance function between embeddings
4. Number of chunks retrieved from the database
5. Prompt
6. LLM parameters (temperature, topK, top, etc)
7. Choice of LLM
8. Etc.

BUT there are more efficient ways

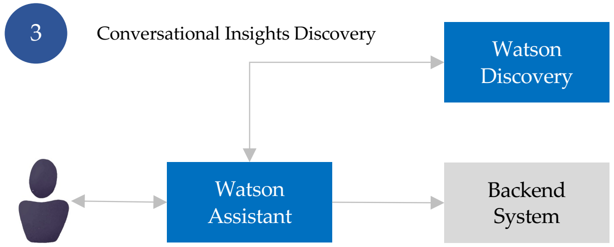


Watson / watsonx.ai Strawman Patterns

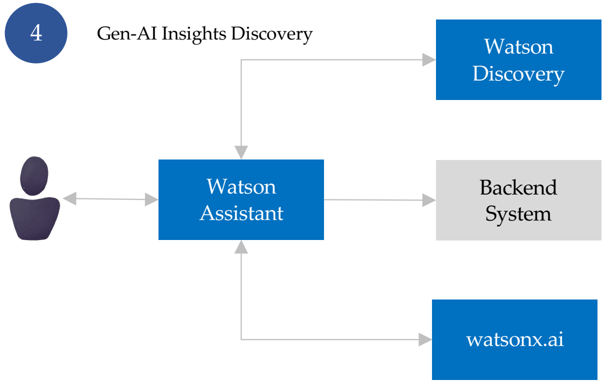
1 Classic Conversational AI

2 Classic Insights Discovery

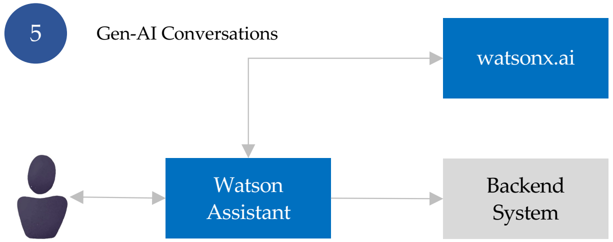
3 Conversational Insights Discovery



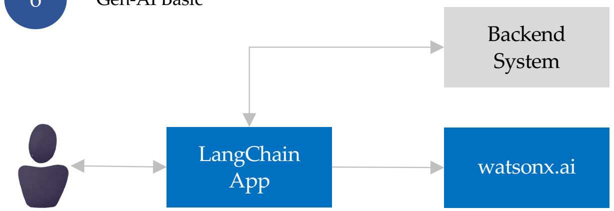
4 Gen-AI Insights Discovery



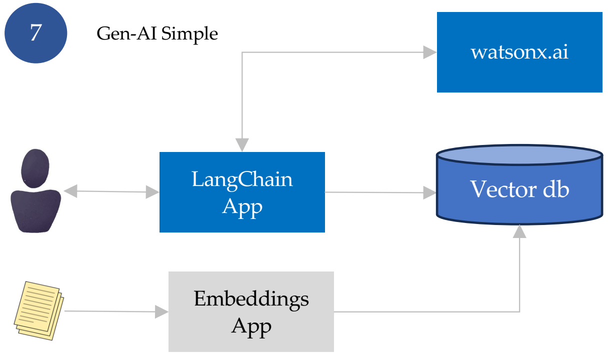
5 Gen-AI Conversations



6 Gen-AI Basic



7 Gen-AI Simple



Tasks

1. Summarization
2. Q & A
3. Extraction
4. Classification
5. Generation
6. Transformation
7. Listing
8. Comparison

IBM Tech

1. Foundation Models
2. Prompt Lab
3. Watson Assistant
4. Watson Discovery
5. Watson Speech
6. IBM Cloud

Sample Technology Stack (non-IBM)

1. Python
2. LangChain
3. ChromaDB, Pinecone…
4. Flask, FastAPI…
5. MySQL

Watsonx.ai using Python SDK

Demo

Thank You