Vector Stores and PostgreSQL

Welcome and Introductions

Introductions

- Kevin Cunningham
  - Lived, taught, and developed in Brighton for 20 years
  - Relocated back to Northern Ireland 3 years ago
  - Dad to two kids
  - Love learning new things (currently 3D printing and local first development)
  - You can find me on Bluesky (@dolearning.dev) or on my website (https://kevincunningham.co.uk)

1. Overview of AI Workflows	2. Why Postgres as a vector store	3. Storing and managing vectors	4. Querying the vector store
• Look at high-level architecture - LLMs, vector stores and JSON • Look at key vocabulary and concepts (embeddings, vectors, hybrid queries, etc.)	• What is a vector store? Key concepts and use cases. • Why Postgres and how does it compare with other market tools • Setting up Postgres with vector capabilities (pgvector) • **Lab**: Install and configure Postgres using Docker	• Generating embeddings: Overview of tools and workflows • Storing and organizing embeddings in Postgres • Strategies for handling large datasets including chunking • Dense and sparse vectors • **Lab:** Generate embeddings for a dataset and store them	• Techniques for similarity search: k-NN, cosine similarity • Using indexes to optimize vector queries • Reranking results • **Lab:** Query stored vectors to retrieve similar items (document/image search)
5. Querying LLMs with retrieved data	**6. NoSQL with JSON in Postgres**	**7. Integrating Vector, Relational and JSON Data**	**8. Putting it all together**
• Recap on querying LLMs vis APIs • Best practices for combining vector retrieval with LLM prompts • Prompt configuration parameters (temperature, top-k, etc) • **Lab:** Build a pipeline where vector store results enhance LLM responses (context-aware Q&A, etc)	• Overview of JSON/JSONB support in Postgres • Querying JSONB data with SQL • Indexing JSONB data for performance • **Lab:** Design a schema mixing vector, relational and JSONB data for a sample project	• Building hybrid queries to power advanced workflows • **Case study:** Combining embeddings, metadata (relational) and configurations (JSON) • **Lab:** Implement a hybrid query to support a sample AI use case	• Full stack pipeline demo: Retrieve data, query the LLM and return results • Debugging and optimising the workflow • Spotlight on LLM frameworks • **Lab:** Build a working application combining all elements

Who are you?

- Name
- Role
- LLM/PostgreSQL/Vector Experience
- What will make these two days feel useful?

Timings (or when are we getting coffee?)

09.30 – 11.00 – Session 1
11.00 – 11.15 – Coffee
11.15 – 12.45 – Session 2
12.45 – 13.45 – Lunch
13.45 – 15.15 – Session 3
15.15 – 15.30 – Tea
15.30 – 16.30 – Session 4


Questions
?

Overview of AI Workflows

Plan for the session

- Recap on key components of AI workflows
- Understand key terminology
- Explore real-world use cases
- Visualise high-level AI architecture
- Integration Challenges and Trade-offs

Modern AI workflows combine structured and unstructured data, embeddings and vector search to enable intelligent systems.

Generate responses, process queries and interact intelligently.

What examples of it are there?

ChatGPT, bp, BERT

What

Enabl
conte

What

Searcl
kitten

Key Components of AI Workflows

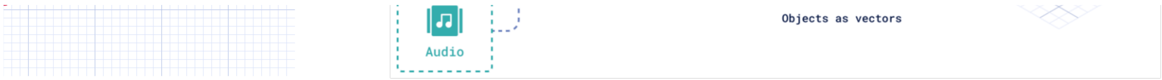
11

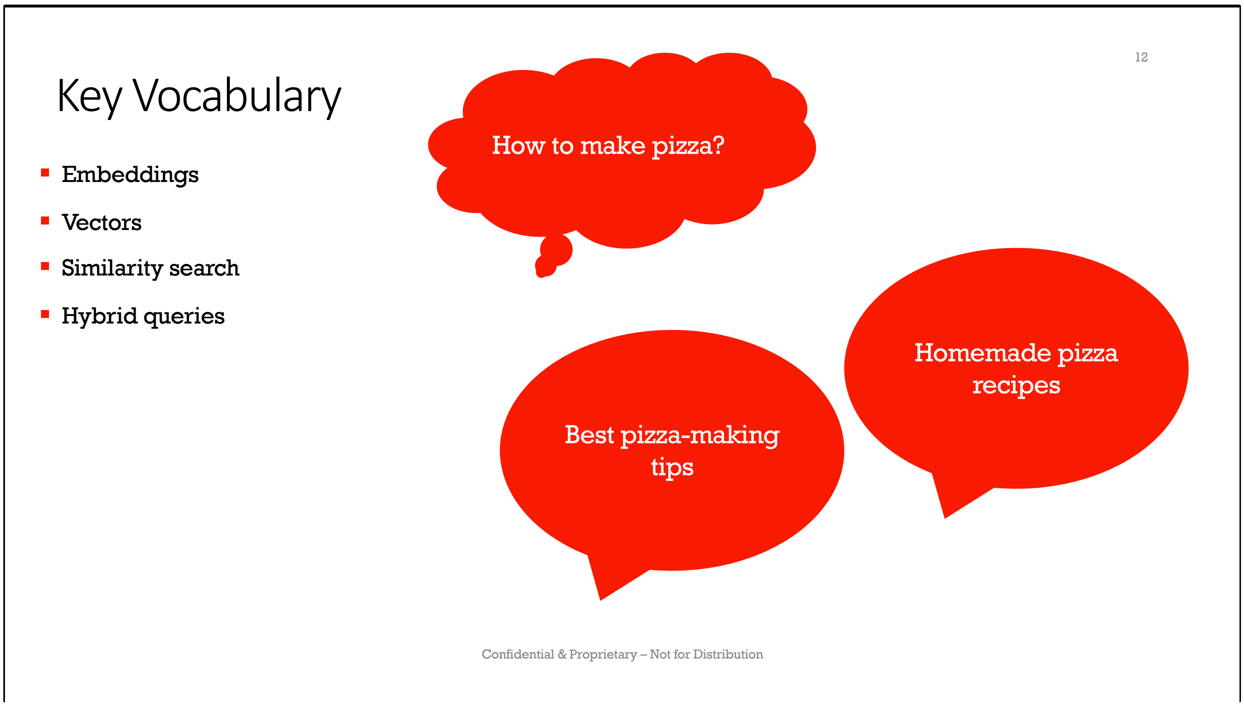
- Large Language Models (LLMs)
- Embeddings
- Vector Stores
- JSON (and Relational Data)

- Can you think of an example of an application you've used recently that likely relies on an LLM?
- A fun way to think about embeddings is as GPS coordinates for concepts in a multi-dimensional space. Words, images, or sentences with similar meanings are 'closer together' in this space.
- Why not just store embeddings in a traditional database? This is a common question, and the answer lies in the computational overhead. Vector stores are specifically optimized for operations like nearest-neighbor searches, which are critical in similarity-based applications.
- Think of an e-commerce site. The vector store retrieves similar products based on embeddings, while the relational data fills in the details you see on the page.

What is JSON?
A lightweight data format used for storing and exchanging structured information in a human-readable and machine-parsable way.
What does it do?
Facilitates flexible data storage, serialization, and transmission, often used in APIs, NoSQL databases, and config files.
What examples of it are there?
PostgreSQL (JSONB), MongoDB, MySQL (JSON), CouchDB, Elasticsearch
What is Relational Data?
A structured data model that organizes information into tables with predefined schemas and relationships.
What does it do?
Ensures data integrity, supports complex queries, and enables efficient storage and retrieval using SQL.
What examples of it are there?
PostgreSQL, MySQL, SQLite, SQL Server, Oracle Database

- These components—LLMs, embeddings, vector stores, and JSON/relational data—are the building blocks of modern AI workflows. In the next sections, we'll dive deeper into their roles, starting with the terminology and vocabulary you need to fully grasp how these pieces fit together.





The **nuances** of natural language or the hidden **meaning** in large datasets of images, sounds, or user interactions are hard to fit into a table.
At their core, vector embeddings are about semantics. They take the idea that "a word is known by the company it keeps" and apply it on a grand scale.

Embeddings are the numerical representations of data—whether it's text, images, or audio—that capture their semantic meaning. Think of embeddings as a way to translate messy, unstructured data into something machines can understand and compare. For instance, the phrase 'cute puppies' and the word 'adorable dogs' might have embeddings that are very close to each other in a multi-dimensional space.

Different models produce embeddings with vector spaces of varying dimensions, depending on how the model was designed and trained.

Embeddings are represented as vectors—essentially, multi-dimensional arrays of numbers. These vectors allow us to perform mathematical operations, such as calculating similarity between two data points. It's through vectors that we can move beyond simple keyword matches to understanding meaning.

Most but not all normalise these to 1.

Similarity Search
This is where the magic happens. Similarity search allows us to compare embeddings to find the closest matches. For example, in a document database, we could search for text that's most 'semantically similar' to a given query. This is much more powerful than keyword search, especially for natural language.

Hybrid Queries
Hybrid queries combine the power of vector-based similarity search with structured database queries. For instance, you could retrieve all products similar to 'wireless earbuds' but filter them by price or stock availability using a traditional SQL query.

Any other terms we want to get out on the table?

Now that we have a shared understanding of these key terms, we're ready to explore how they're applied in practical workflows, starting with embeddings and vector stores.

Real-World Applications

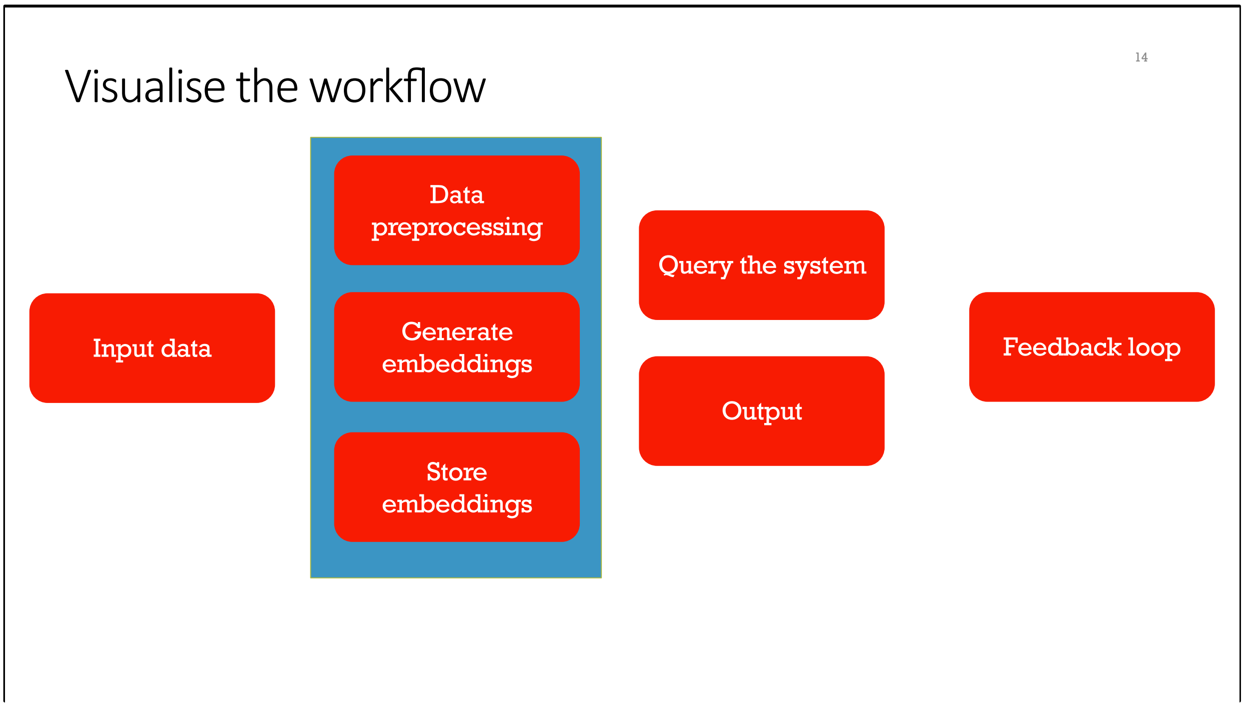
- Semantic Search
- Recommendation Systems
- Q&A
- Image/Multimedia Search
- Hybrid Applications

Activity

- Split in groups (2/3)
- Pick one of the application types
- Discuss and brainstorm a **new or unique use case** for the selected application type.
- Think about a specific domain (e.g., healthcare, education, e-commerce, etc.) where this application could solve a real-world problem.
- Summarize your idea in 1-2 sentences. Be ready to share your group's idea with everyone in the room.

Optional Prompts to Guide Thinking:

•"What problem does this solve?"
•"Who would benefit from this application?"
•"What kind of data would you need to make it work?"



Input Data:
"Everything starts with raw data. This could be a user query, an uploaded image, or audio input. For example, think of a user typing 'best books on Python' into a search bar or uploading an image for reverse image search. The type of input determines the preprocessing and embedding approach."
Data Preprocessing:
"Once we have the raw input, preprocessing ensures the data is clean and ready for embedding generation. For text, this might mean tokenizing words, removing stopwords, or normalizing casing. For images, it might mean resizing or cropping. Preprocessing ensures that we're feeding consistent, high-quality data into the next step."
Prompt to Audience:
•"What challenges do you think preprocessing might introduce when working with large, unstructured datasets?"
Generate Embeddings:
"After preprocessing, the data is transformed into embeddings using a model like BERT, GPT, or a custom-trained model. These embeddings are essentially numerical representations that capture the meaning or features of the data. For instance, the phrase 'learn Python' might generate an embedding close to

'Python tutorials' in the vector space."

Example:

•"Imagine embeddings as coordinates in a multi-dimensional space where similar inputs are located closer together."

Challenging Question:

•"How do you decide which model to use for generating embeddings? What trade-offs might you need to consider?"

Store Embeddings:

"These embeddings are then stored in a vector store, which is optimized for tasks like similarity search and fast retrieval. Examples of vector stores include pgvector for Postgres, FAISS for local memory, and Pinecone for cloud-based scalability."

Key Point:

•"This stage is critical for systems like semantic search or recommendation engines, where speed and efficiency matter."

Query the System:

"When a user interacts with the system—like searching for 'best books' or 'similar images'—their input is converted into an embedding and matched against stored embeddings. The system uses similarity measures like cosine similarity to find the closest matches. In more advanced setups, hybrid queries combine this semantic search with traditional SQL filters, like filtering products by price or location."

Prompt to Audience:

•"Can you think of scenarios where hybrid queries might outperform purely semantic ones?"

Output:

"The results are presented to the user in a meaningful way—like a ranked list of documents, product recommendations, or a set of related images. This stage bridges the technical output with the end-user experience."

Example:

•"For a Q&A system, the output might be a direct answer generated by an LLM, while for a search engine, it might be a ranked list of articles."

Feedback Loop:

"This step is often overlooked but is critical for improving the system over time. Feedback loops involve learning from user interactions—such as clicks, likes, or corrections—and using that data to retrain embeddings, optimize search rankings, or refine the model. For instance, if a user consistently ignores certain recommendations, the system can adjust to avoid similar results in the future."

Challenging Question:

•"How do you think feedback loops could be integrated into an AI workflow without overwhelming the system with noise or bad data?"

Closing Statement:

"This workflow outlines the journey data takes through an AI system. Over the course of this session, we'll explore each of these stages in more detail, focusing on how they're implemented and optimized in real-world scenarios."

15

Area	Challenge	Trade-off
Data preprocessing		
Embedding generation		
Vector Stores		
Querying the System		
Feedback Loop		

Data Preprocessing:

•"The quality of your input data determines the effectiveness of the entire system. However, cleaning and preprocessing data takes time and resources, especially with unstructured formats like text or images."

Embedding Generation:

•"Choosing the right embedding model is a critical decision. Larger models offer more accuracy but come with a trade-off in speed and computational requirements. For instance, GPT-based embeddings might capture richer meaning but may be too slow for real-time applications."

Vector Stores:

•"Storing and retrieving embeddings at scale requires significant infrastructure. Vector stores optimized for performance can be expensive and may introduce challenges in distributed systems."

Querying the System:

•"Hybrid queries, while powerful, require careful integration of semantic and structured search. Balancing complexity with performance is an ongoing challenge."

Feedback Loop:

•"Incorporating user feedback can improve system accuracy, but not all feedback is useful. For example, biased or noisy feedback can lead to unintended consequences, such as reinforcing stereotypes or degrading search quality."

Modern AI workflows combine structured and unstructured data, embeddings and vector search to enable intelligent systems.


Questions
?