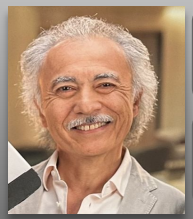
Semantic Search from the Command Line

Learning about Embeddings/Similarity for Java Developers

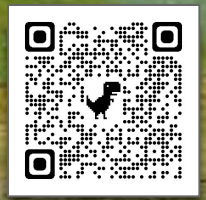
Gran Sasso

NYJavaSIG



Frank Greco
@frankgreco
linkedin.com/in/frankdgreco

Chair
NYJavaSIG - NY Java User Group
@nyjavasig



Topics

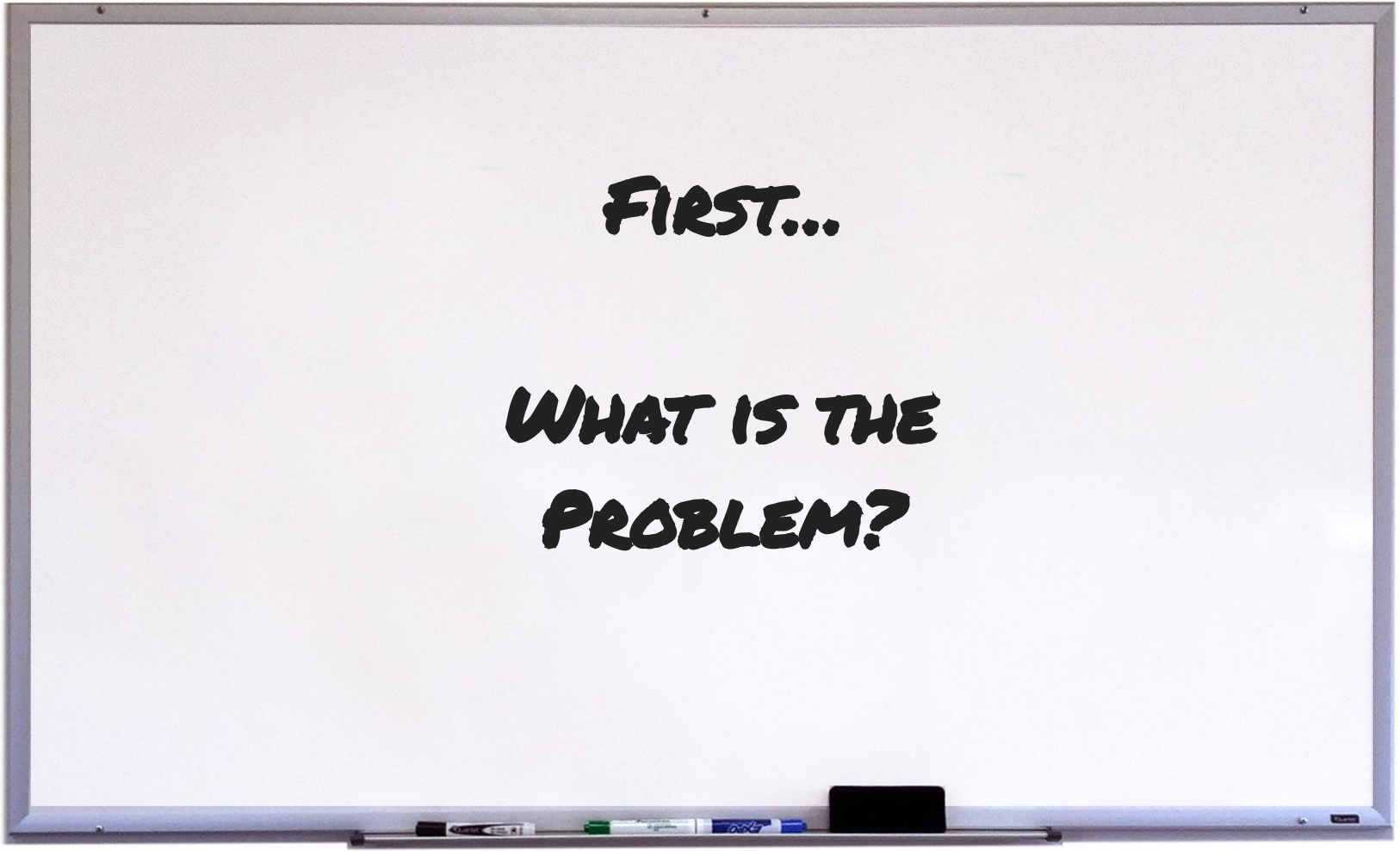
- (5) Problem Description
- (20) What are Embeddings and What is Similarity
- (5) LangChain4j Overview
- (5) Get Embeddings - demo
- (5) Compare Embeddings - demo
- (10) Vector Stores and Chunking
- (10) LangChain4j Ingestion - demo
- (5) Vector Search and Filters
- (5) Semantic Search - demo
- (5) Q/A

Goals

Search for similar text content on your filesystem.

- Need info, but can't remember filename or exact text
- Can't come up with a regexp-able string as search target

- Need to understand embeddings / "similarity" for GenAI project
  - Venkat did emphasize learning at D2N
- You are a CLI person
- Desperately need a distraction from following the news...


First...
What is the Problem?

Problem to solve

You need to recall some important information.

You recall that information in a document on your filesystem,
but you don't remember the filename, the exact text.
And you can't come up with a useful regexp as a target.

```
grep - Standard GNU grep utility
egrep - Extended grep (equivalent to grep -E)
fgrep - Fast grep for fixed strings (equivalent to grep -F)
rgrep - Recursive grep (equivalent to grep -r)
pgrep - Process grep for searching processes
zgrep - Grep for gzip-compressed files
bzgrep - Grep for bzip2-compressed files
xzgrep - Grep for xz-compressed files
ngrep - Network packet grep
agrep - Approximate grep for fuzzy matching
sgrep - Structured grep for XML/JSON
ugrep - Ultra grep with TUI and fuzzy search
ripgrep (rg) - Fast grep respecting gitignore
ggrep - GNU grep on non-GNU systems
vgrep - Visual grep with interactive TUI
hgrep - Grep with syntax highlighting
cgrep - Context-aware grep for source code
gitgrep - Grep in git repositories
lcgrep - Grep with line context
tgrep - Grep with directory tree view
mgrep - Multi-pattern grep
wgrep - Word grep
jgrep - JSON-aware grep
xmlgrep - XML-aware grep
pygrep - Python-specific grep
jsgrep - JavaScript-specific grep
phpgrep - PHP-specific grep
gogrep - Go-specific grep
rggrep - Regex grep with advanced features
fngrep - Filename grep
lgrep - Line-oriented grep
dgrep - Disk grep for searching file systems
igrep - Interactive grep
kgrep - Kernel grep for searching Linux kernel source
webgrep - Web page grep
csvgrep - CSV-aware grep
sqlgrep - SQL-aware grep
bingrep - Binary file grep
```

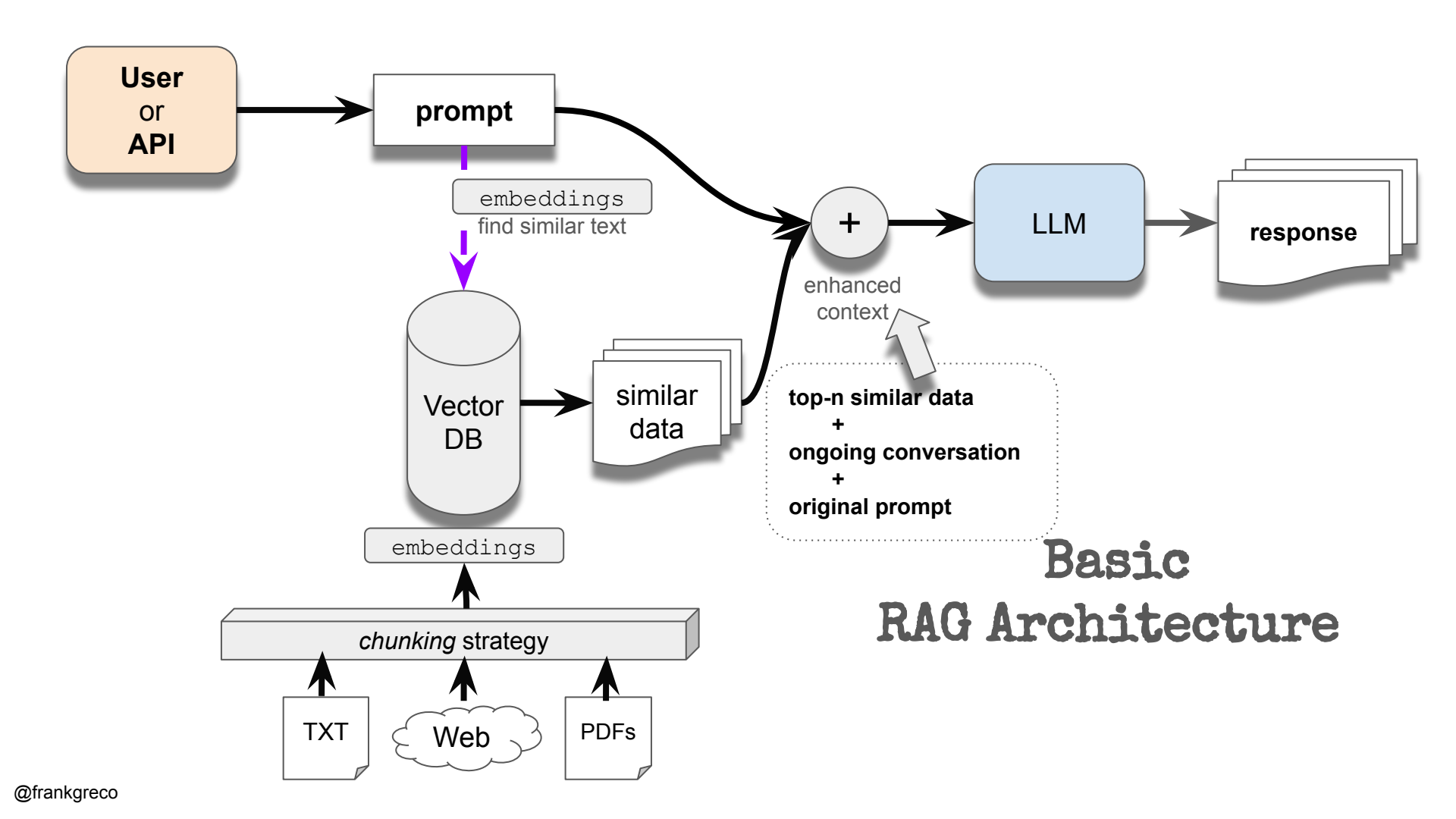
The Grep Family

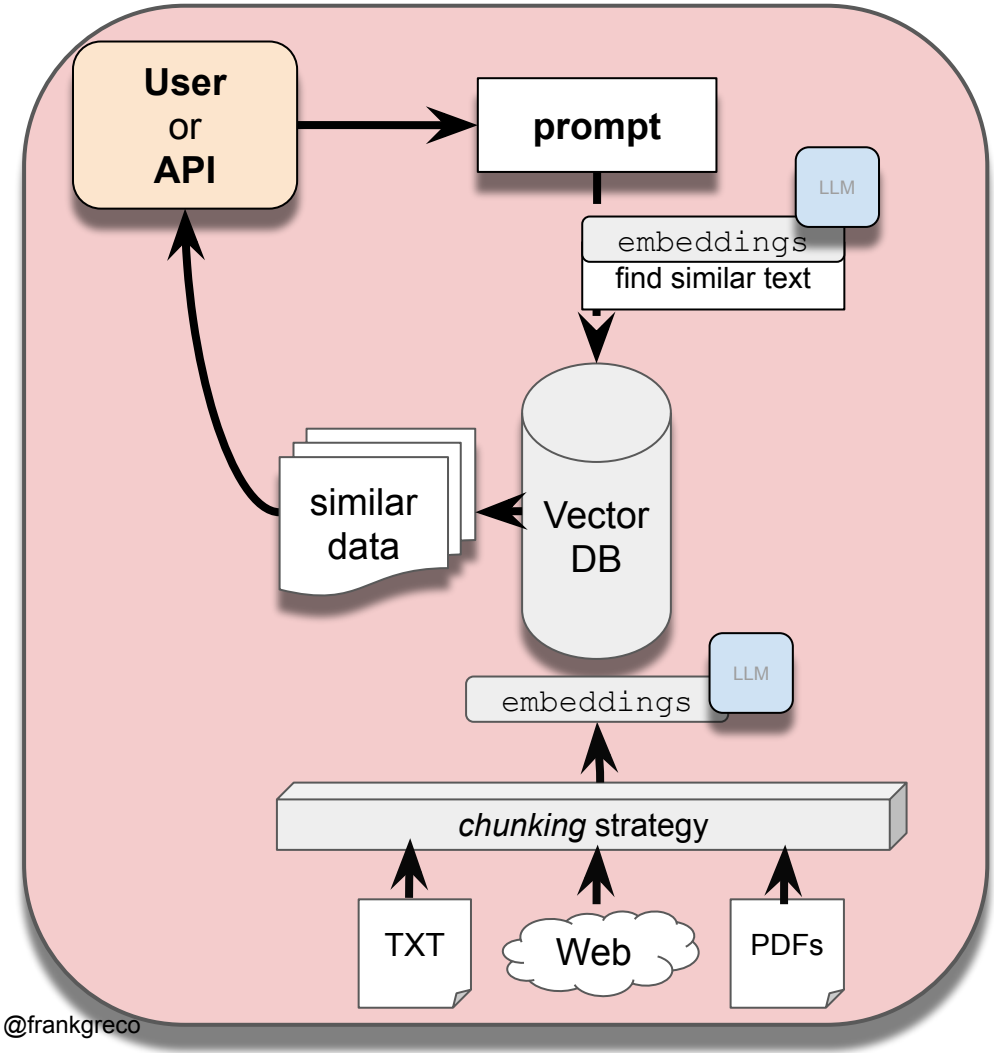
Some have regexp and “fuzzy” matching.

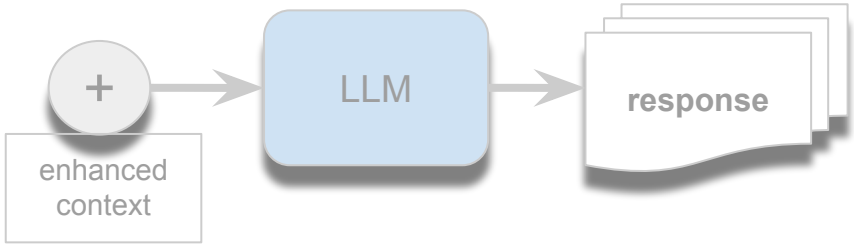
Regexp doesn’t account for ***similar*** matches.
Fuzzy matches have high false positives.

Arguable whether all of these are actually necessary.
But I digress…

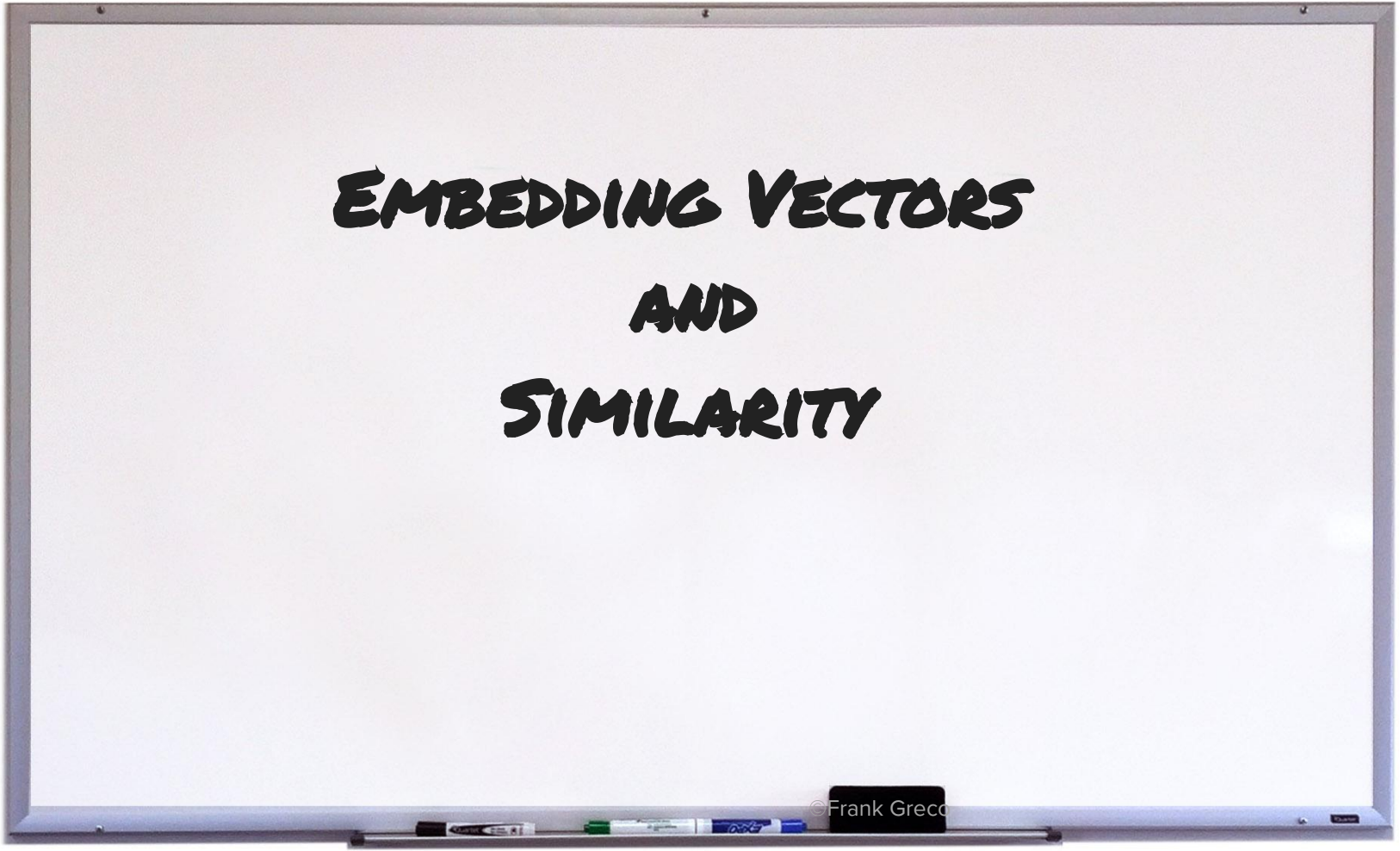
Wasn't there something about a basic GenAI RAG system that determined similarity?


User
or
API
prompt
embeddings
find similar text
+
LLM
response
enhanced
context
Vector
DB
similar
data
top-n similar data
+
ongoing conversation
+
original prompt
embeddings
chunking strategy
TXT
Web
PDFs
Basic
RAG Architecture

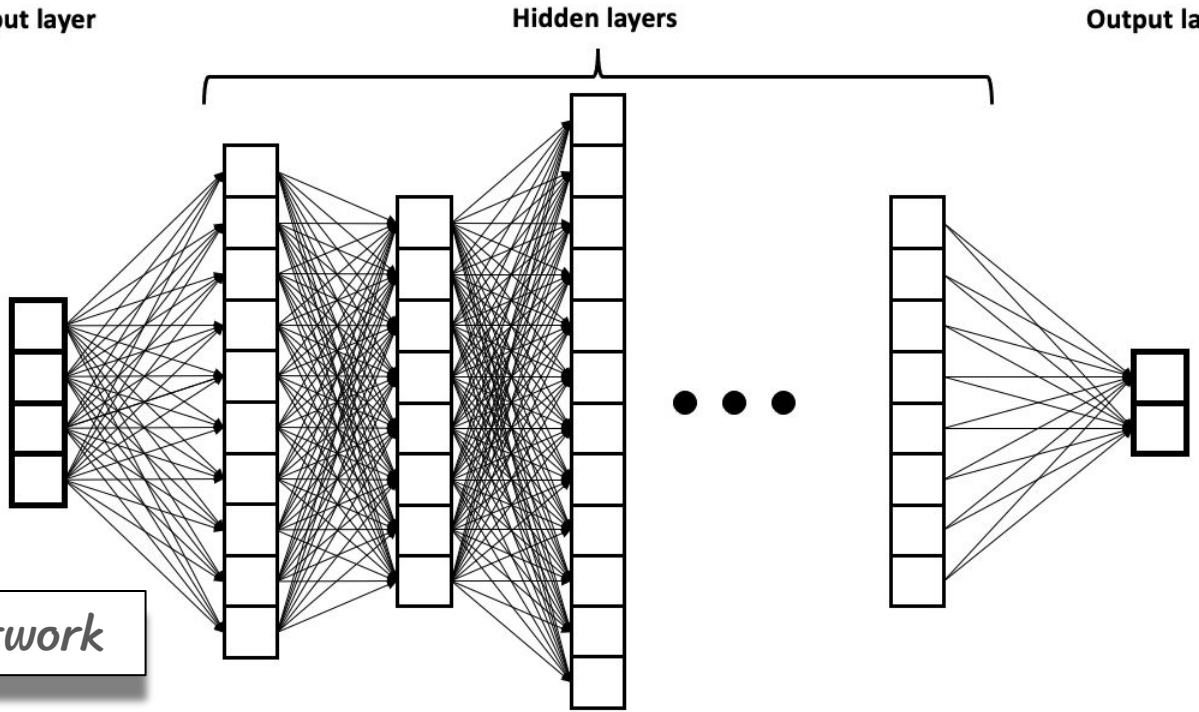




Basic RAG Architecture


Embedding Vectors
and
Similarity
©Frank Greco

In AI applications,
Embeddings are used to determine ***Similarity*** *.*



Neural Network

http://commons.wikimedia.org/wiki/File:Example_of_a_deep_neural_network.png

Similarity Applications

Text

Semantic search, sentiment analysis, language understanding, anomaly detection, grouping support issues

Images

Similar images, style detection, clustering, visual search, classify video scenes

Audio

Music recommendation, sound pattern detection


Google Arts & Culture | Experiments | t-SNE Map
GUIDED TOUR
https://experiments.withgoogle.com/t-sne-map
3D
SHOWING ARTWORKS FROM 623 PARTNER COLLECTIONS

@frankgreco

*Let's talk about **Text** Embeddings*

Text Embeddings

Fixed-length, numerical vectors representing words, sentences, paragraphs, or even entire documents.

"Embedding vector"

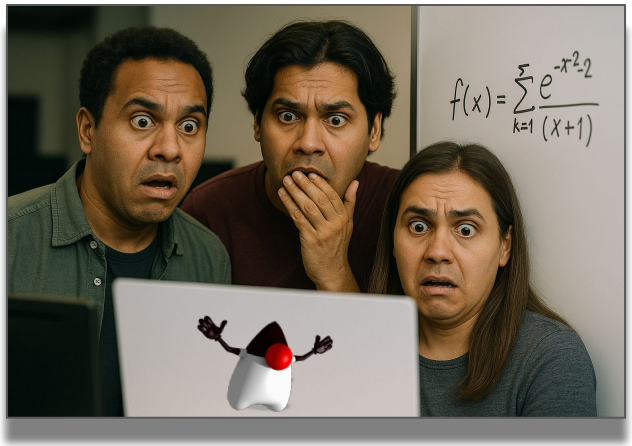
Using Java/LangChain4j

```
Embedding vector = [-0.005175813, -0.013680333, 0.018862674,
0.024423573, 0.024097228, -0.0039813947, -0.037333734, -0.003062737,
-0.033208746, 3.4674033E-4, 0.0653732, -0.04566203, 0.002189767,
0.013823925, 0.0059884093, 0.010958626, -0.0035310402, -0.037960313,
-0.021460371, 0.051118497, 0.01725706, -0.064015605, 0.015742827,
0.022517724, 0.016826289, 0.03213834, …
```

Ewww… I'm afraid of math!!
There are no semi-colons or curly braces!

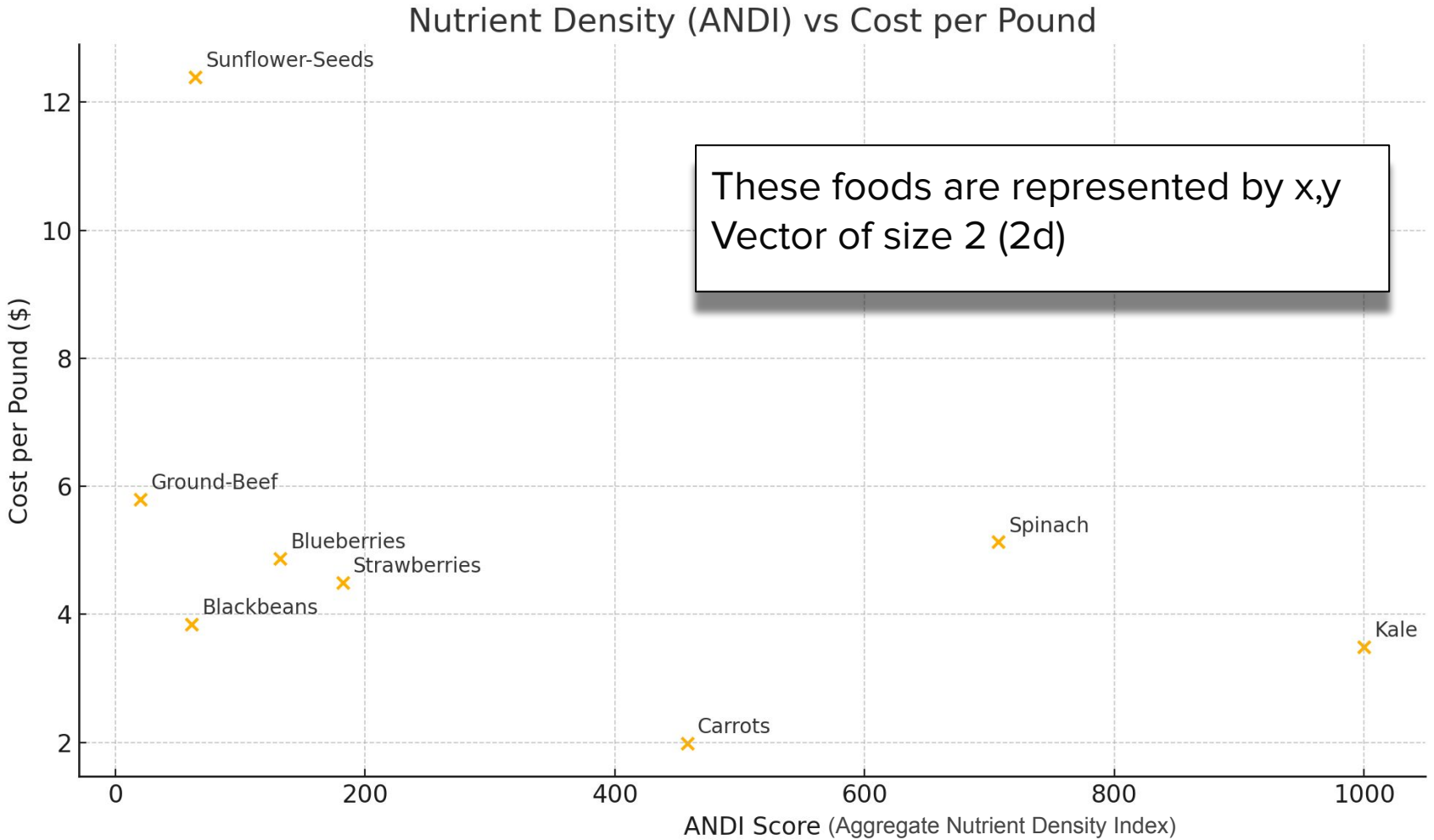
Text Embeddings are Useful!

Text embeddings are used to extract semantic meaning and calculate **similarity**.

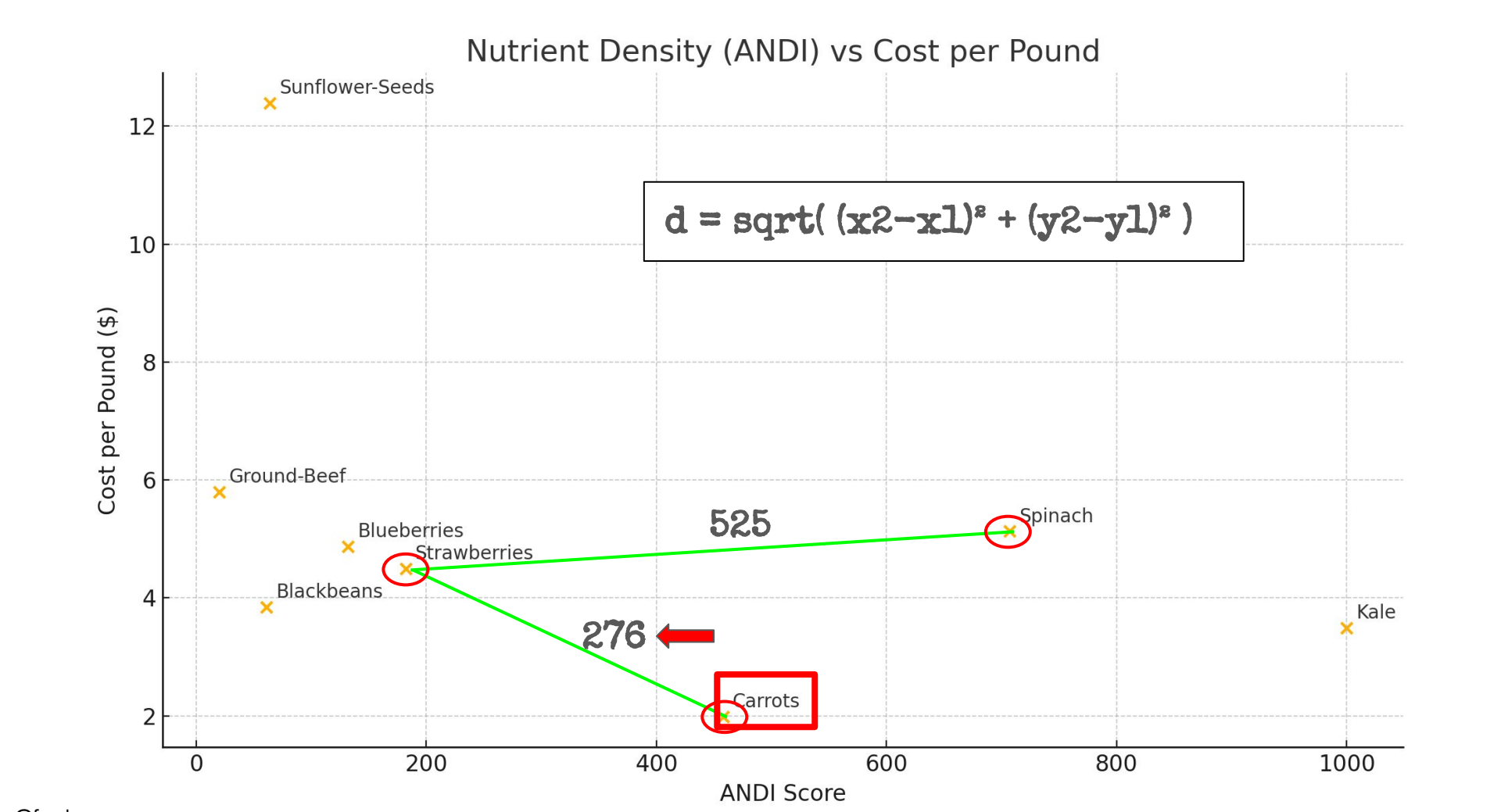


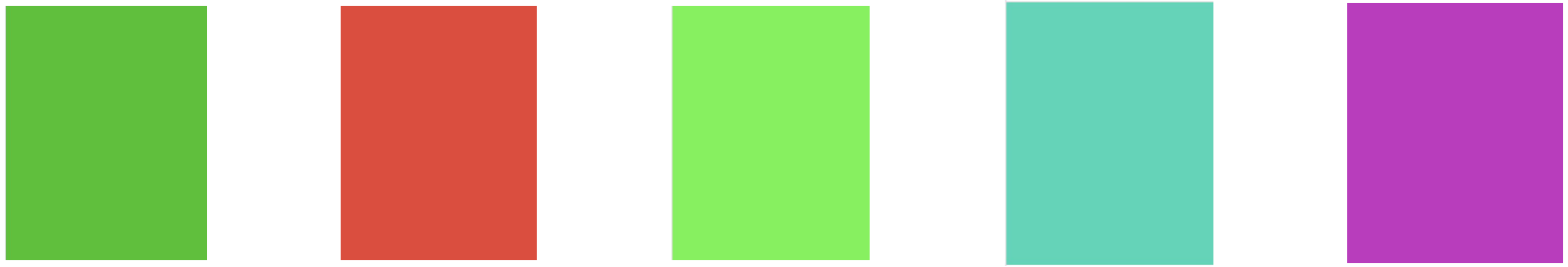
Wait a minute,

What exactly is “Similarity”?

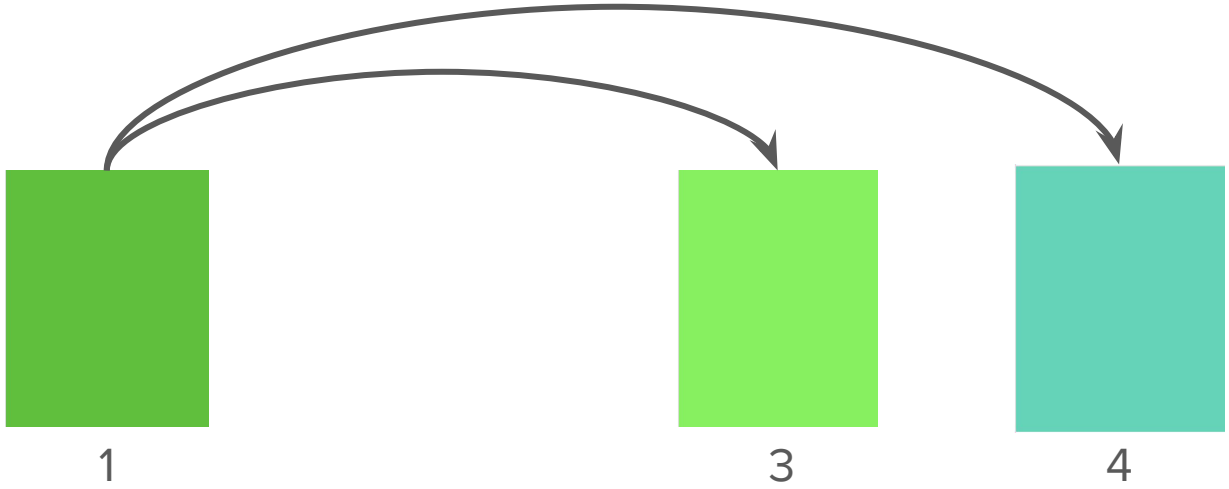


aanmc.org/wp-content/uploads/2022/08/ANDI-Guide-Gly-Index.pdf


Nutrient Density (ANDI) vs Cost per Pound
Sunflower-Seeds
Ground-Beef
Blueberries
Strawberries
Blackbeans
Spinach
Kale
Carrots
525
276
d = sqrt((x2-x1)² + (y2-y1)²)
Cost per Pound ($)
ANDI Score
0
200
400
600
800
1000
2
4
6
8
10
12



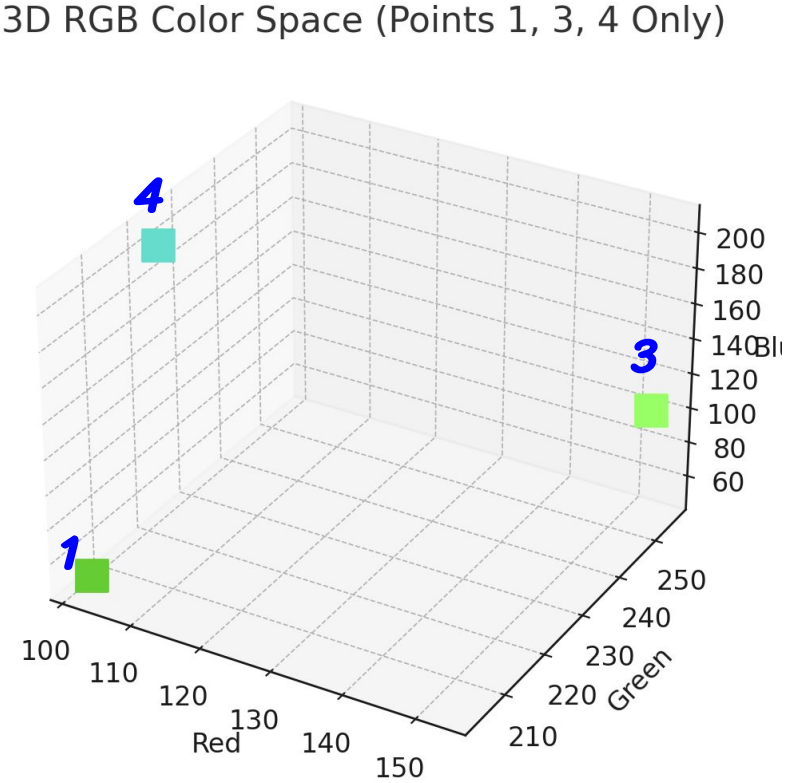
Color similarity



Is Color-3 or Color-4 more similar to Color-1?

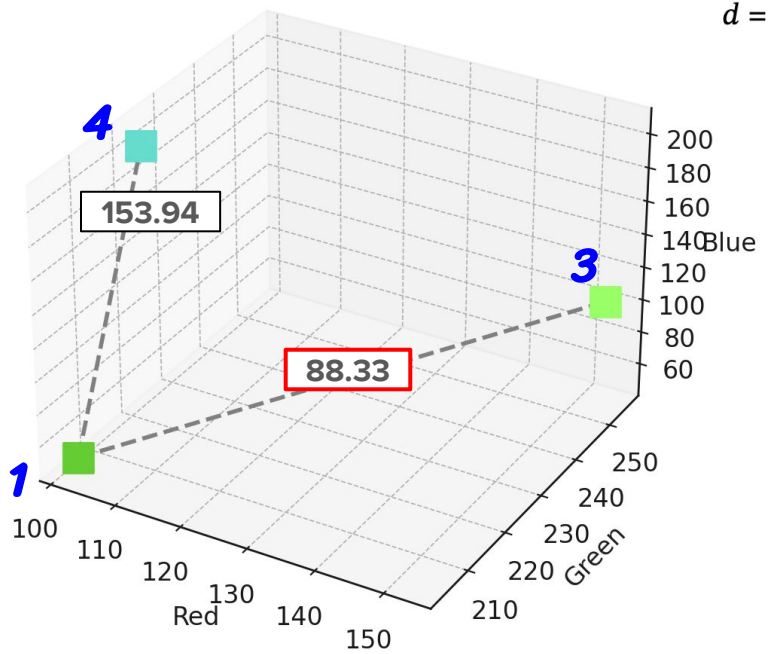
RGB, CMYK, HSV, pantone, LAB, Indexed color, grayscale, bitmap, CIELAB

These colors are represented by R,G,B
Vector of size 3 (3d)



These colors are represented by R,G,B
Vector of size 3 (3d)

3D RGB Color Space with Connecting Lines (Points 1, 3, 4)



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

One way of calculating similarity.
“Euclidean” similarity

Interesting!

But what about text similarity?

Instead of 2 or 3 dimensions,
Text (String) characteristics have a lot!

Num of dimensions related to size of LLM's internal neural network.

Typical # dimensions for vector: 384, 768, 1536, 2048, etc.

Since neural networks are not all the same size,
embedding vectors are not all the same size.

"Embedding" originates from mathematics.

Mapping from one space to another.

Food -> (2d) nutrition and cost
Color -> (3d) R, G and B
Text Strings -> (1,536d) many characteristics...

There are dozens of embedding services and libraries to use.

Btw, don't mix embedding vectors from different sources!

Embeddings Service

```
import …        // LangChain4j imports


public class GetEmbedding {

   public static void main(String[] argv) {
     String apiKey = System.getenv("OPENAI_API_KEY");

     EmbeddingModel model = OpenAiEmbeddingModel.builder()
       .apiKey(apiKey)
       .modelName(TEXT_EMBEDDING_3_SMALL)
       .build();

     Response<Embedding> response = model.embed("One fish, two fish, red fish, blue fish");
     Embedding embedding = response.content();

     System.out.println(embedding);
   }
}
```

```
1536 elements: Embedding { vector = [-0.005175813, -0.013680333, 0.018862674, 0.024423573,
0.024097228, -0.0039813947, -0.037333734, -0.003062737, -0.033208746, 3.4674033E-4,
0.0653732, -0.04566203, 0.002189767, 0.013823925, 0.0059884093, 0.010958626,
-0.0035310402, -0.037960313, -0.021460371, 0.051118497, 0.01725706, -0.064015605,
0.015742827, 0.022517724, 0.016826289, 0.03213834, 6.065916E-4, 0.03098961, 0.021512585,
0.018875727, 0.073779814, -0.016943771, 0.0035473574, -0.016434675, -0.057227653,
0.004944109, 0.0071665104, -0.024397464, 0.015181515, -0.0010826456, 0.018901834,
0.048690498, 0.054825764, -0.013641172, 0.06960261, 0.020794628, -0.043286245,
-0.028666042, -0.005857872, 0.06552984, -0.040127236, 0.08552819, -0.054877978,
0.037620917, -0.0023317267, -0.008654638, -0.049499832, 0.037829775, 0.024045013,
-0.058271956, 0.054773547, 0.032477736, 0.0038214861, 0.010025282, -0.020246372,
0.040988784, -0.047698416, -0.017792266, 0.052084476, 0.018458007, -0.016056117,
0.020481339, 0.014554935, 0.013099441, 0.044878803, 0.025063206, -0.043703966,
-0.034096405, -6.1026297E-4, 0.054773547, -0.01366728, 0.017374545, 0.018027233,
-0.05540013, -0.023849208, -0.03161619, 0.0140197305, -0.0344358, -0.025520088,
-0.013758656, -0.0048527326, -0.04532263, -0.088400014, -0.03955287, -0.001202577,
-0.022857122, 0.024084175, 0.015077085, -0.034383588, 0.025781162, 0.019658953,
-0.04106711, -0.041328184, -0.020559661, 0.0057208072, 0.011500357, 0.03310432,
-0.008367456, -0.0011772853, -0.005985146, -0.01847106, 0.005883979, 0.015155408,
-9.129469E-4, -0.00326507, 0.054721333, 0.008387037, -0.029266514, -0.0054303613,
-0.023314003, 0.022491617, -0.027047377, 0.046314716, -0.031433437, -0.029240407,
0.0093987025, -0.010423422, -0.032608274, 0.0029240407, 0.00829566, -0.001980907,
0.021930305, -0.017178738, -0.008256499, -0.005567426, 0.024567163, -0.042842418, ...
```

```
1536 elements: Embedding { vector = [-0.005175813, -0.013680333, 0.018862674, 0.024423573,
0.024097228, -0.0039813947, -0.037333734, -0.003062737, -0.033208746, 3.4674033E-4,
0.0653732, -0.04566203, 0.002189767, 0.013823925, 0.0059884093, 0.010958626,
-0.0035310402, -0.037960313, -0.021460371, 0.051118497, 0.01725706, -0.064015605,
0.015742827, 0.022517724, 0.016826289, 0.03213834, 6.065916E-4, 0.03098961, 0.021512585,
0.018875727, 0.073779814, -0.016943771, 0.0035473574, -0.016434675, -0.057227653,
0.004944109, 0.0071665104, -0.024397464, 0
0.048690498, 0.054825764, -0.013641172, 0.
-0.028666042, -0.005857872, 0.06552984, -0
0.037620917, -0.0023317267, -0.008654638,
-0.058271956, 0.054773547, 0.032477736, 0.
0.040988784, -0.047698416, -0.017792266, 0
0.020481339, 0.014554935, 0.013099441, 0.0
-0.034096405, -6.1026297E-4, 0.054773547,
-0.05540013, -0.023849208, -0.03161619, 0.0140197305, -0.0344358, -0.025520088,
-0.013758656, -0.0048527326, -0.04532263, -0.088400014, -0.03955287, -0.001202577,
-0.022857122, 0.024084175, 0.015077085, -0.034383588, 0.025781162, 0.019658953,
-0.04106711, -0.041328184, -0.020559661, 0.0057208072, 0.011500357, 0.03310432,
-0.008367456, -0.0011772853, -0.005985146, -0.01847106, 0.005883979, 0.015155408,
-9.129469E-4, -0.00326507, 0.054721333, 0.008387037, -0.029266514, -0.0054303613,
-0.023314003, 0.022491617, -0.027047377, 0.046314716, -0.031433437, -0.029240407,
0.0093987025, -0.010423422, -0.032608274, 0.0029240407, 0.00829566, -0.001980907,
0.021930305, -0.017178738, -0.008256499, -0.005567426, 0.024567163, -0.042842418, ...
```

You rarely look at these values.

Just understand that these values represent how one embedding algorithm views the characteristics of a string.

Each individual element isn't really meaningful by itself, but the vector represents the semantics of the string.

(Very) loosely speaking, each element is a string attribute.

HOW DO I CALCULATE TEXT SIMILARITY??

There are several ways of calculating similarity of 2 points in N-dimensional space.

Probably more than a dozen...

Each algorithm has its advantages and disadvantages for different types of data.

Euclidean Similarity

$$\text{distance (b,a)} = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2}$$

Euclidean distance between two vectors

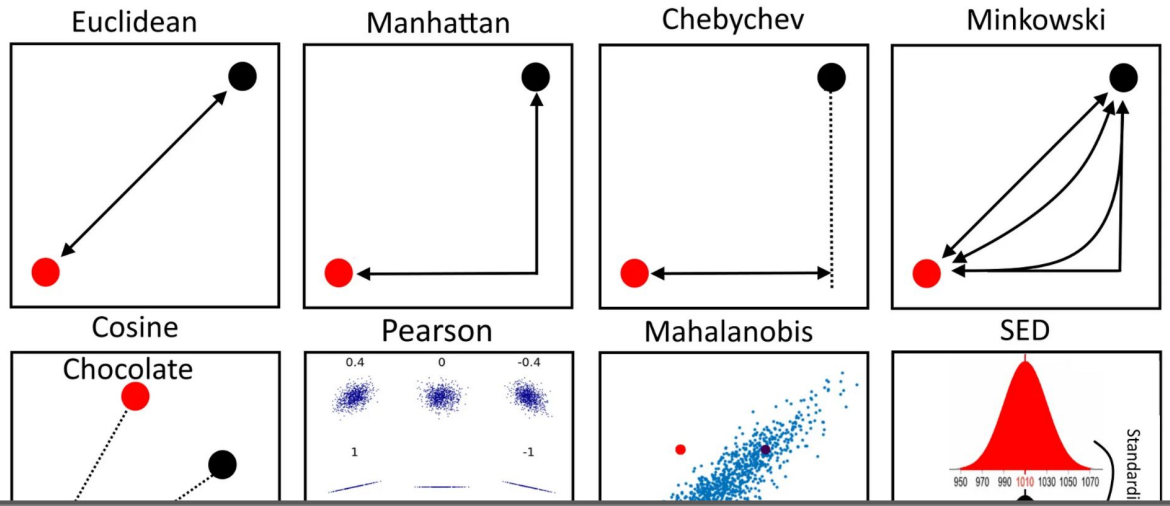
Similarity of two vectors
Closer to 0 implies similarity

https://towardsdatascience.com/calculating-document-similarities-using-bert-and-other-models-b2c1a29c9630

17 types of similarity and dissimilarity measures used in data science.

The following article explains various methods for computing distances and showing their instances in our daily lives. Additionally, it will introduce you to the pydist2 package.

Mahmoud Harmouch · Follow
Published in Towards Data Science · 25 min read · Mar 13, 2021



https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681

Cosine Similarity

Machine friendly since it uses matrices!

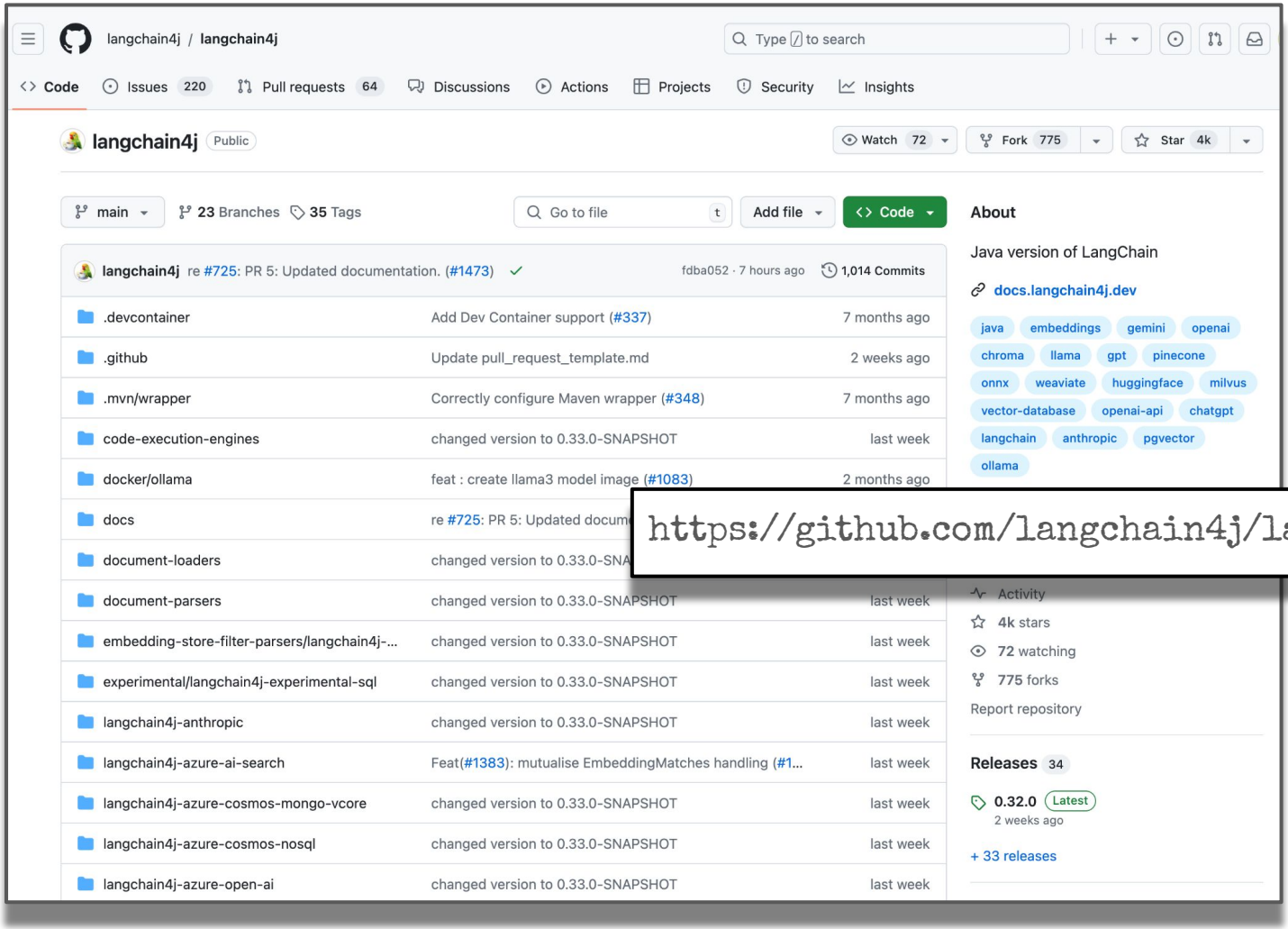
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Cosine of the angle between two vectors

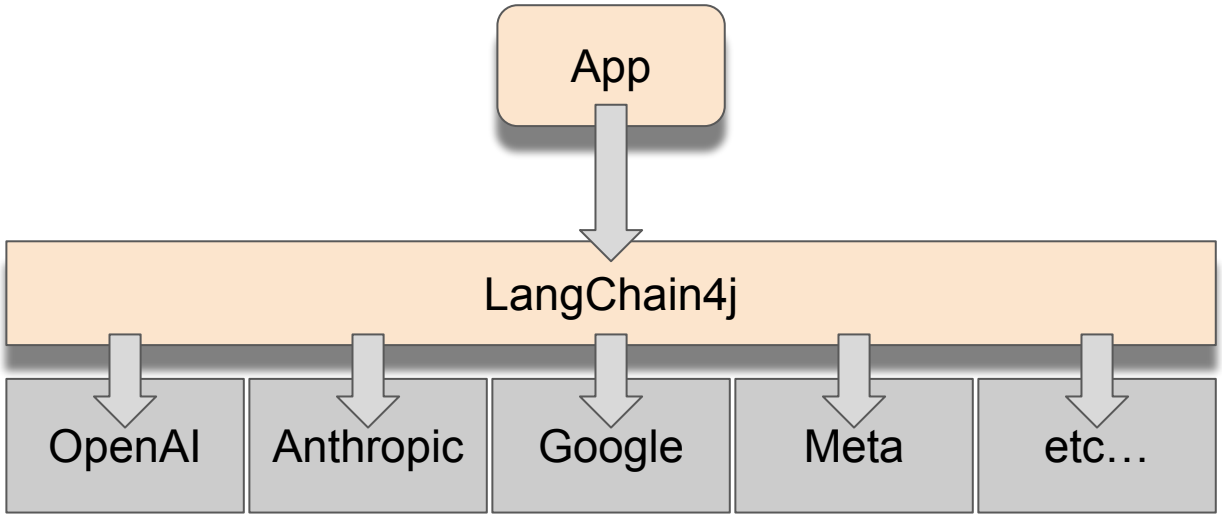
Similarity of two vectors [-1, +1]

https://towardsdatascience.com/calculating-document-similarities-using-bert-and-other-models-b2c1a29c9630


LangChain4j
Overview
©Frank Greco


langchain4j / langchain4j
Type / to search
Code
Issues 220
Pull requests 64
Discussions
Actions
Projects
Security
Insights
langchain4j Public
Watch 72
Fork 775
Star 4k
main
23 Branches
35 Tags
Go to file t
Add file
Code
About
Java version of LangChain
docs.langchain4j.dev
java embeddings gemini openai
chroma llama gpt pinecone
onnx weaviate huggingface milvus
vector-database openai-api chatgpt
langchain anthropic pgvector
ollama
langchain4j re #725: PR 5: Updated documentation. (#1473)
fdba052 · 7 hours ago
1,014 Commits
.devcontainer Add Dev Container support (#337) 7 months ago
.github Update pull_request_template.md 2 weeks ago
.mvn/wrapper Correctly configure Maven wrapper (#348) 7 months ago
code-execution-engines changed version to 0.33.0-SNAPSHOT last week
docker/ollama feat : create llama3 model image (#1083) 2 months ago
docs re #725: PR 5: Updated docum
document-loaders changed version to 0.33.0-SNA
document-parsers changed version to 0.33.0-SNAPSHOT last week
embedding-store-filter-parsers/langchain4j-... changed version to 0.33.0-SNAPSHOT last week
experimental/langchain4j-experimental-sql changed version to 0.33.0-SNAPSHOT last week
langchain4j-anthropic changed version to 0.33.0-SNAPSHOT last week
langchain4j-azure-ai-search Feat(#1383): mutualise EmbeddingMatches handling (#1... last week
langchain4j-azure-cosmos-mongo-vcore changed version to 0.33.0-SNAPSHOT last week
langchain4j-azure-cosmos-nosql changed version to 0.33.0-SNAPSHOT last week
langchain4j-azure-open-ai changed version to 0.33.0-SNAPSHOT last week
https://github.com/langchain4j/langchain4j
Activity
4k stars
72 watching
775 forks
Report repository
Releases 34
0.32.0 Latest
2 weeks ago
+ 33 releases

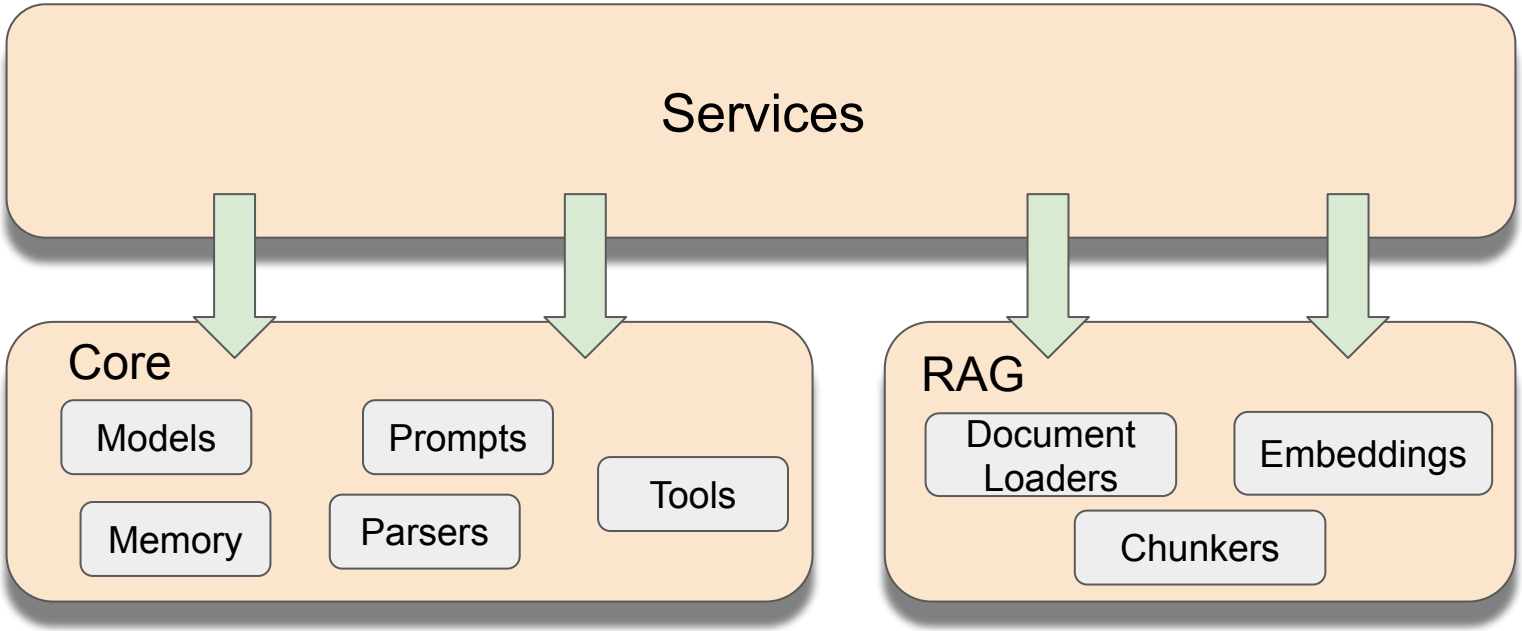
@frankgreco



Ollama
HuggingFace
Bedrock
LocalAI
…

Provides an abstraction for LLMs and associated tools.

LangChain4J Components



Simple Completion

```java
public class SimpleChatCompletion {
    public static void main(String[] args) throws IOException {

        ChatModel cmodel = OpenAiChatModel.builder()
                .apiKey(System.getenv("OPENAI_API_KEY"))
                .modelName(OpenAiChatModelName.GPT_4_O)
                .timeout(Duration.ofSeconds(30))
                .maxTokens(1024)
                .build();

        List<ChatMessage> messages = new ArrayList<>();

        UserMessage usermsg = new UserMessage("How should I learn Java?");
        messages.add(usermsg);

        Response<AiMessage> answer = cmodel.chat(messages);

        System.out.println(answer.content().text());
    }
}
```

Simple Completion

```java
public class HelloWorldGoogle {
    public static void main(String[] argv) {
        String apiKey = System.getenv("GOOGLE_API_KEY");

        ChatModel model = GoogleAiGeminiChatModel.builder()
                .apiKey(apiKey)
                .modelName("gemini-1.5-flash")
                .build();

        List<ChatMessage> messages = new ArrayList<>();
        UserMessage usrmsg = UserMessage.from("Why should I learn Java.");
        messages.add(usrmsg);

        var answer = model.chat(messages);

        System.out.println(answer.aiMessage().text());
    }
}
```

Simple Completion with System Message

```java
public class SimpleChatCompletion {
    public static void main(String[] argv) {

        ChatModel cmodel = AnthropicChatModel.builder()
            .apiKey(System.getenv("ANTHROPIC_API_KEY"))
            .modelName(CLAUDE_3_5_SONNET_20241022)
            .timeout(Duration.ofSeconds(120))
            .maxTokens(256)
            .build();

        List<ChatMessage> messages = new ArrayList<>();

        SystemMessage sysmsg = new SystemMessage("""
                You are a polite Java expert explaining concepts to a grammar school child.
            """);
        messages.add(sysmsg);

        UserMessage usrmsg = UserMessage.from("Why should I learn Java.");
        messages.add(usrmsg);

        var answer = cmodel.chat(messages);

        System.out.println(answer.aiMessage().text());
    }
}
```

Check version!

```
def lc4jVersion = "1.4.0"
...
implementation "dev.langchain4j:langchain4j:${lc4jVersion}"
implementation "dev.langchain4j:langchain4j-open-ai:${lc4jVersion}"
implementation "dev.langchain4j:langchain4j-google-ai-gemini:${lc4jVersion}"
implementation "dev.langchain4j:langchain4j-anthropic:${lc4jVersion}"
implementation "dev.langchain4j:langchain4j-mistral-ai:${lc4jVersion}"
...
```

Gradle dependencies
`build.gradle`

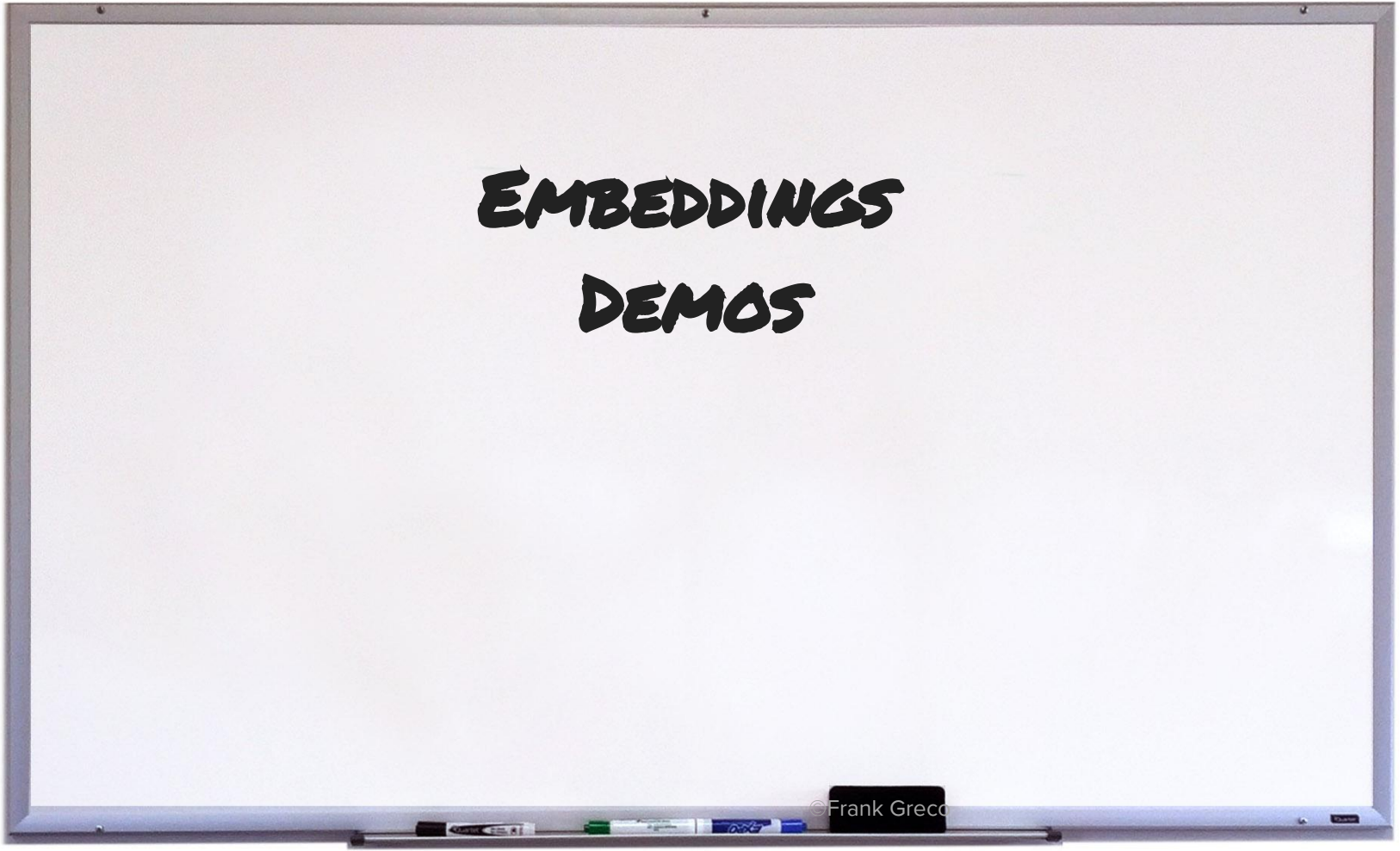
Use the LC4J libraries that you are using in your app

```
<dependency>
    <groupId>dev.langchain4j</groupId>
    <artifactId>langchain4j-core</artifactId>
    <version>1.4.0</version>
</dependency>

<dependency>
    <groupId>dev.langchain4j</groupId>
    <artifactId>langchain4j-open-ai</artifactId>
    <version>1.4.0</version>
</dependency>
```

Check version!

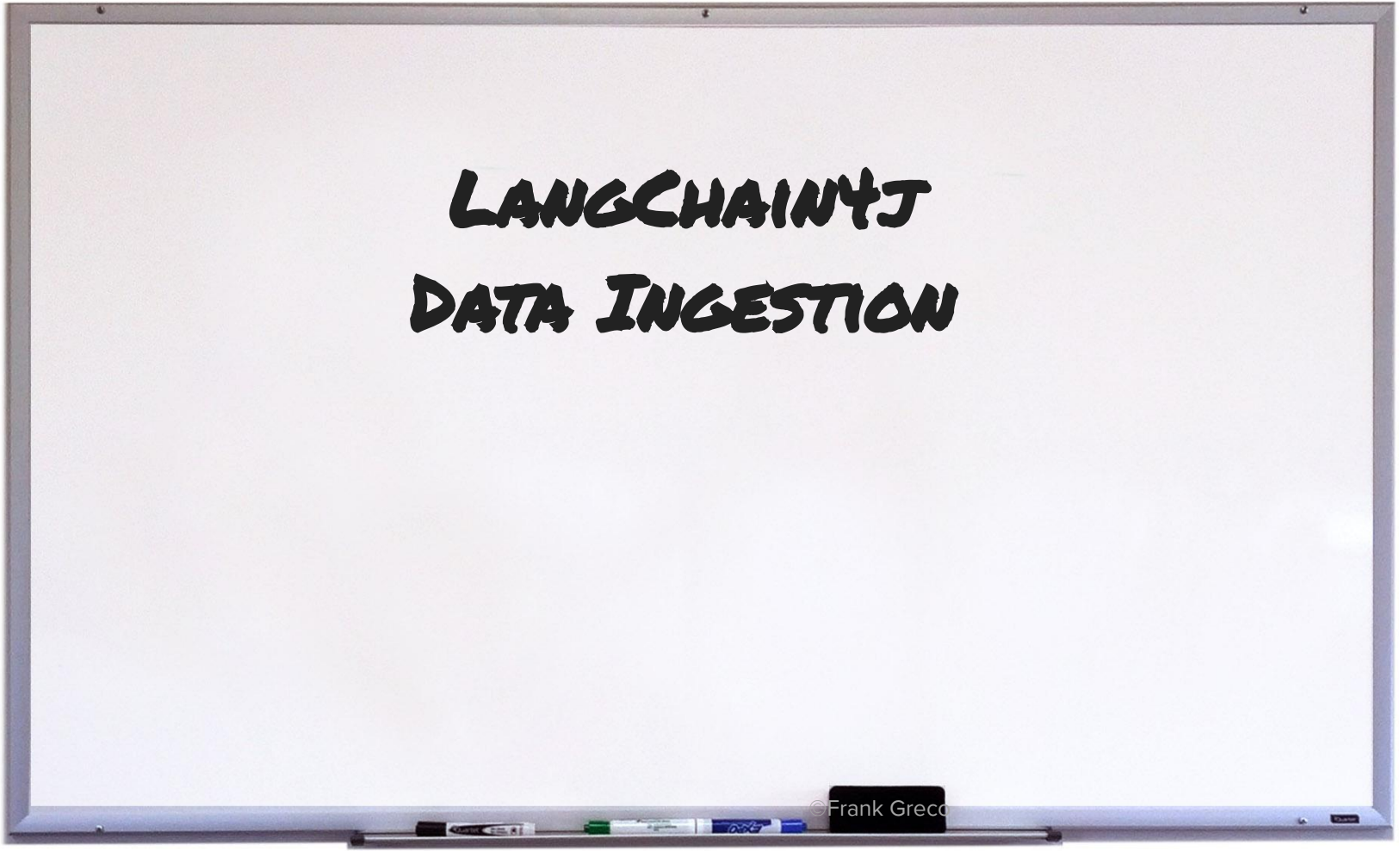
Maven dependencies
`pom.xml`

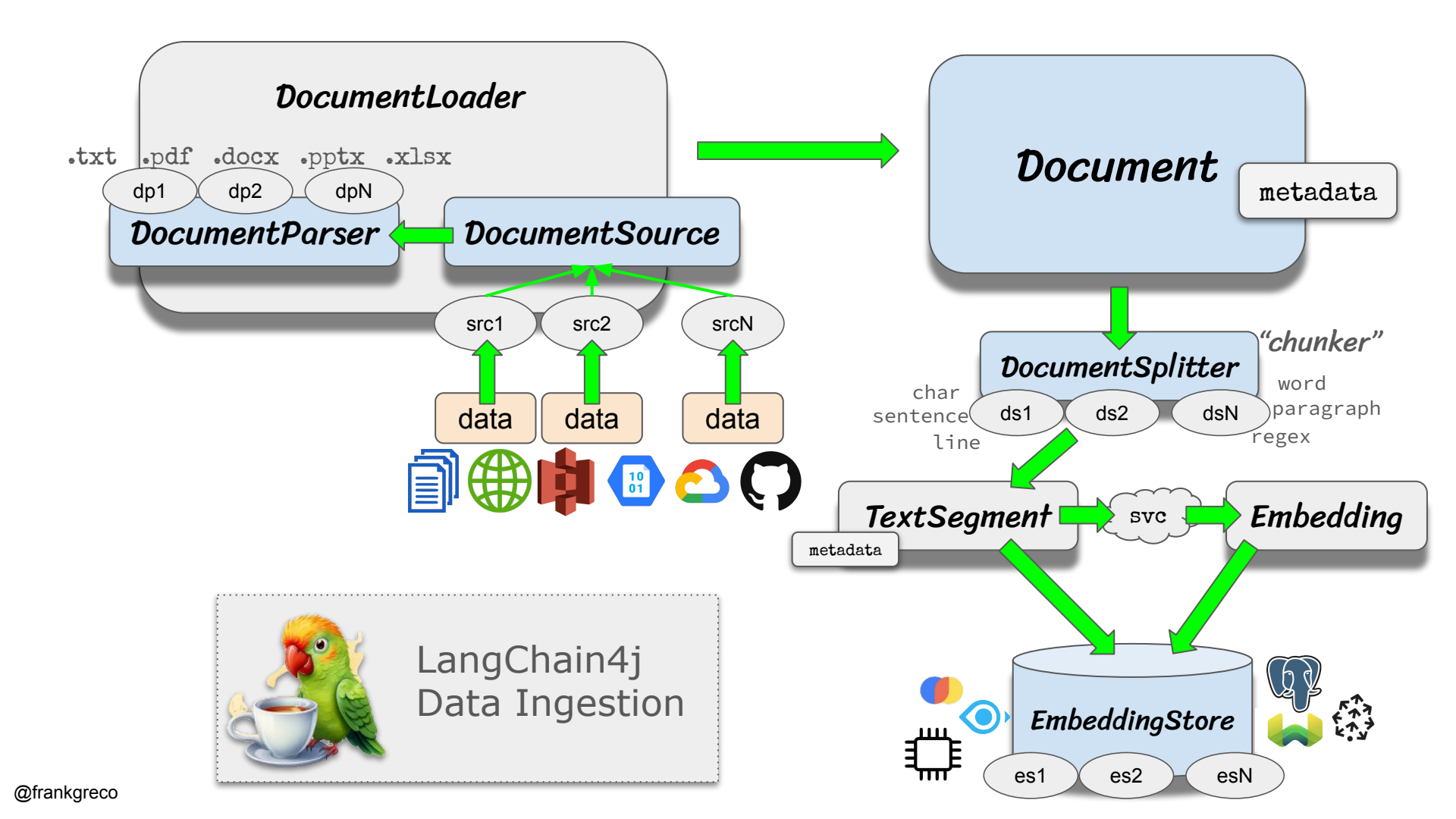
Use the LC4J libraries that you are using in your app


Embeddings
Demos
©Frank Greco

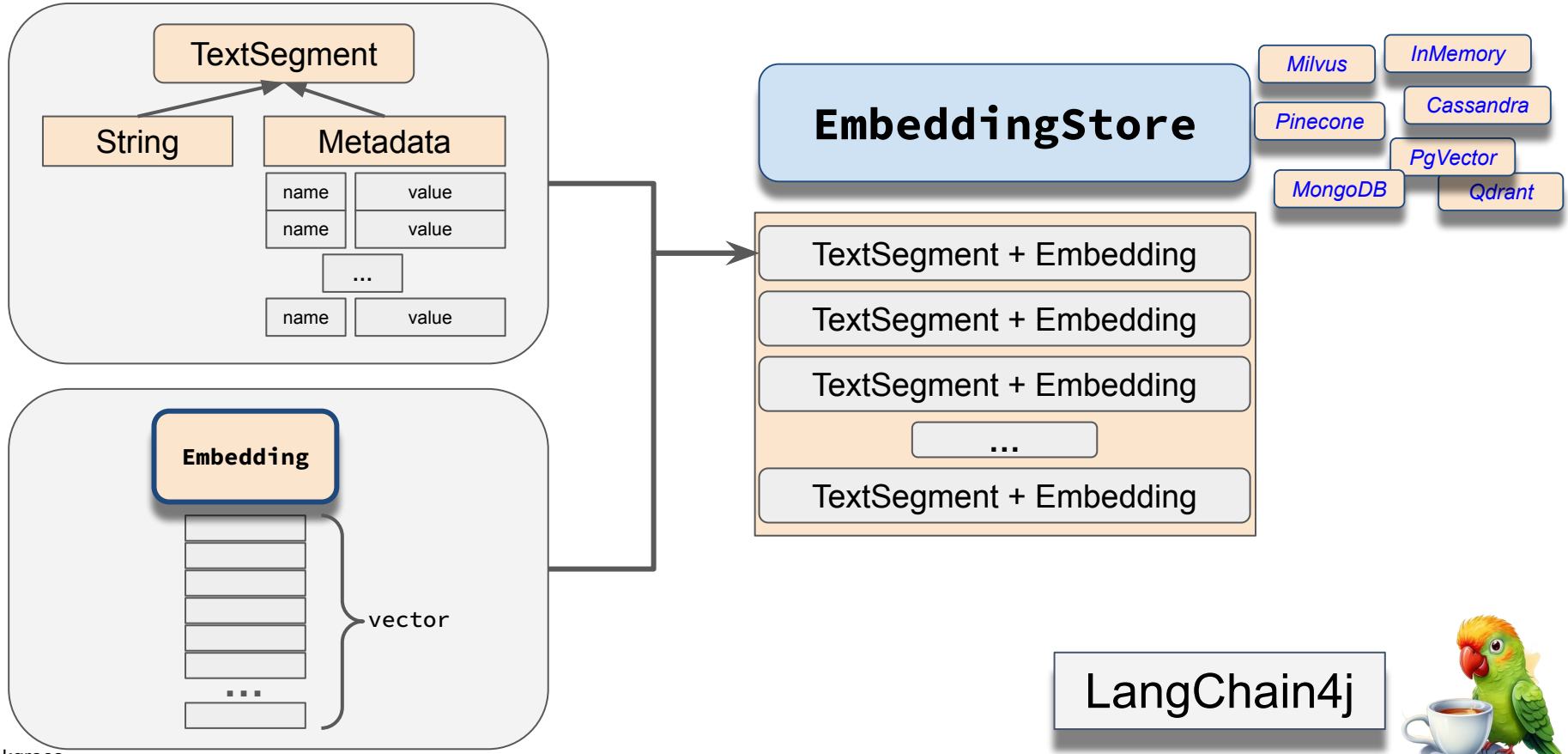
```java
public class GetEmbedding {
 public static void main(String[] argv) {
   String apiKey = System.getenv("OPENAI_API_KEY");

   EmbeddingModel model = OpenAiEmbeddingModel.builder(
         .apiKey(apiKey)
         .modelName(TEXT_EMBEDDING_3_SMALL)
         .build();

   Response<Embedding> response = model.embed(
                           "One fish, two fish, red fish, blue fish");
   Embedding embedding = response.content();

   System.out.println(embedding.dimension() + " elements: " + embedding);
  }
}
```

```
1536 elements: Embedding { vector = [-0.0052022343, -0.013655049,
0.018811593, 0.024425045, 0.024098681, -0.0039979527, -0.037362095,
-0.0030776078, -0.03321075, 3.4002998E-4, 0.06532492, -0.04566478,
0.002183372, 0.0138378125, 0.005965925, 0.010939704, -0.0035443075,
-0.0379626, -0.021461664, 0.051121578, 0.017245047, -0.06391503, 0.01575683,
0.022492973, 0.016827302, 0.032166388, 5.6379294E-4, 0.031043695,
0.021526936, 0.018876866, 0.07373204, -0.016970903, 0.0035475714,
-0.01642261, -0.057231102, 0.0049770433, 0.007206106, -0.02438588,
0.01518243, -0.0010810791, 0.018902974, 0.048641216, 0.05482907,
-0.013641994, 0.06960681, 0.020769773, -0.043314964, -0.028693879,
-0.005858225, 0.06558601, -0.040103547, 0.085585564, -0.054881286,
0.037623186, -0.0023171809, -0.008648633, -0.049528927, 0.03780595,
0.024059517, -0.058223248, 0.05477685, 0.032531913, 0.003802135,
0.0100128325, -0.020247592, 0.040991254, -0.047701288, -0.0178325,
0.052061506, 0.018459119, -0.016096247, 0.020456465, 0.014568866,
0.013093703, 0.04490762, …
```

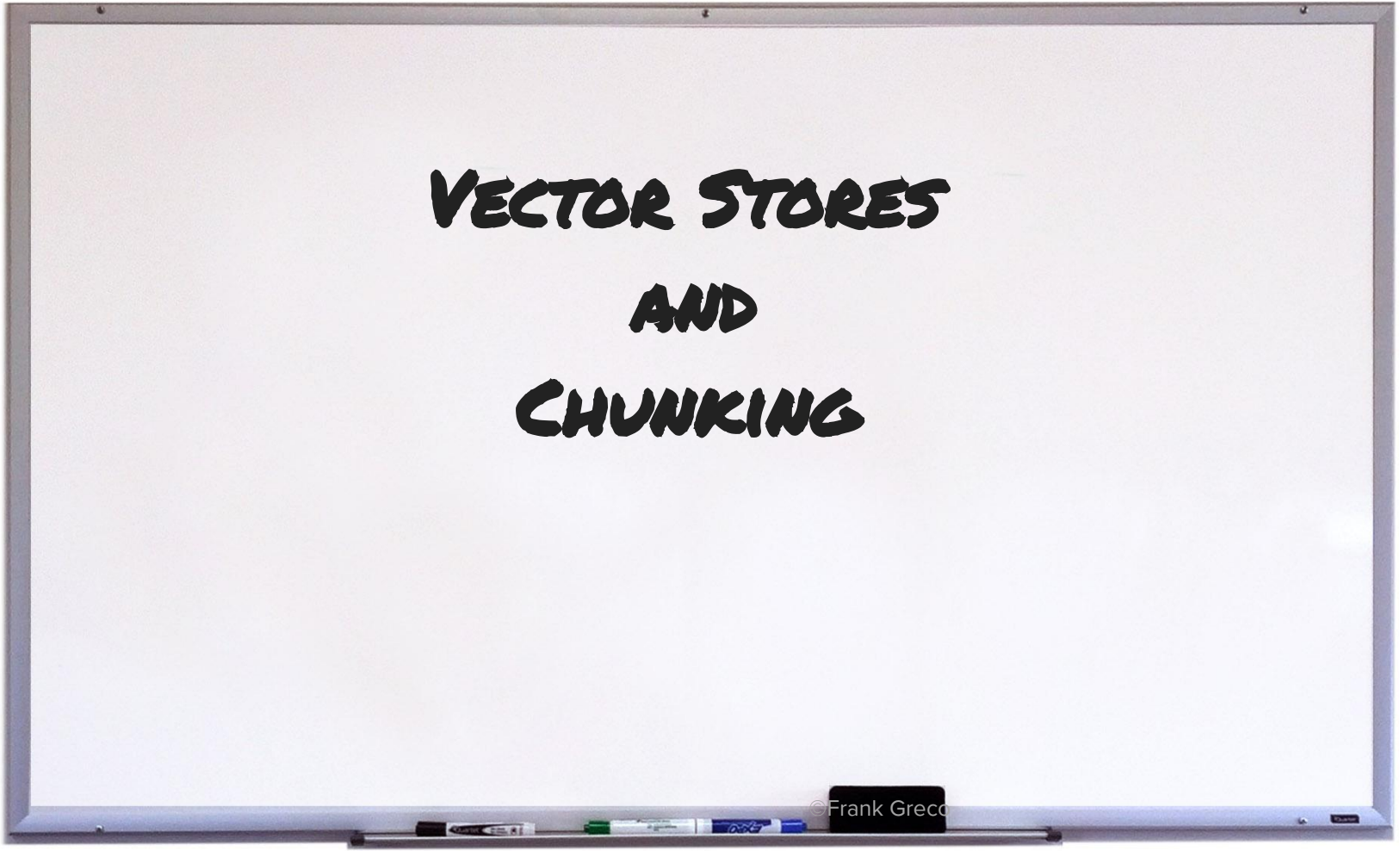

LangChain4j
Data Ingestion
©Frank Greco


DocumentLoader
.txt .pdf .docx .pptx .xlsx
dp1
dp2
dpN
DocumentParser
DocumentSource
src1
src2
srcN
data
data
data
Document
metadata
DocumentSplitter
"chunker"
char
sentence
line
word
paragraph
regex
ds1
ds2
dsN
TextSegment
metadata
svc
Embedding
EmbeddingStore
es1
es2
esN
LangChain4j
Data Ingestion
@frankgreco

Storing Embedding Vectors, Strings, and Metadata



LangChain4j


Vector Stores
and
Chunking
©Frank Greco

While you can implement this yourself
for educational purposes, use a library.

Most (embedding) vector databases offer a variety of
high-performance similarity functions.

Vector Database (VDB)

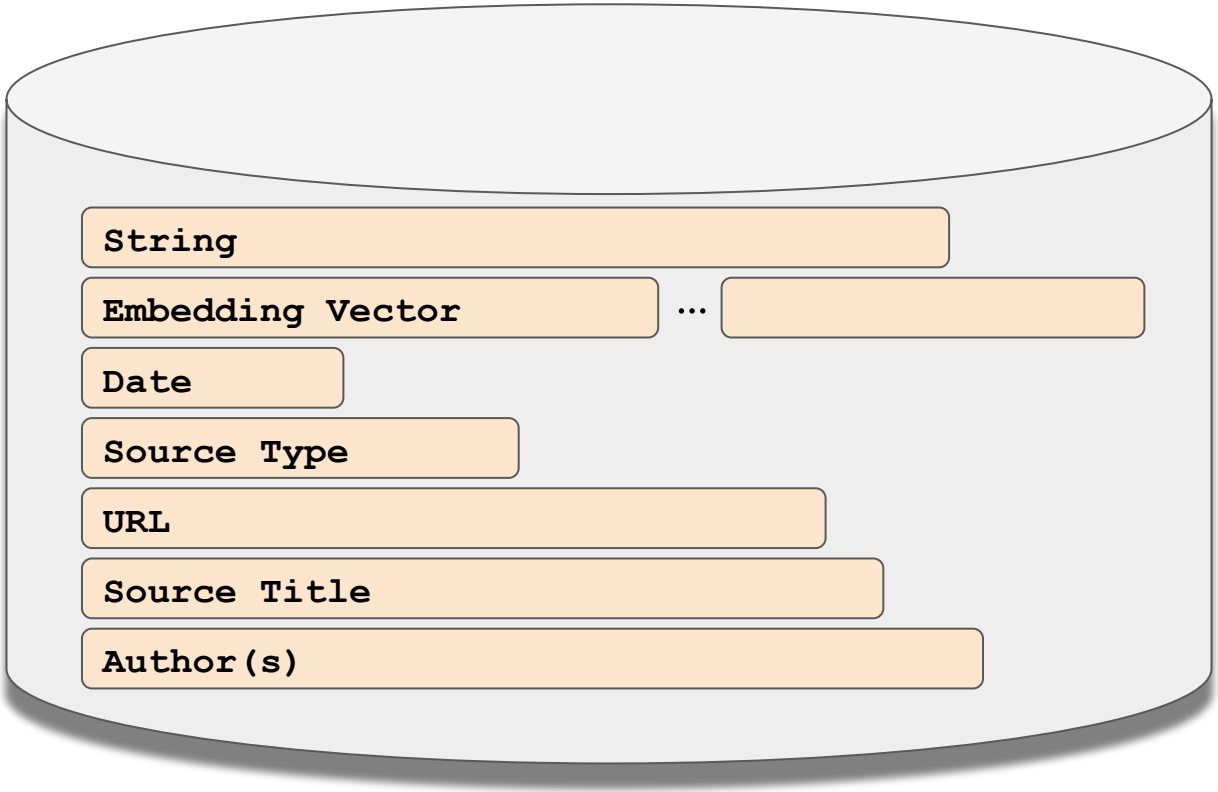
Stores the embedding vector and the original data.

Provides fast similarity search across all the vectors.

Can store metadata (the source URL of the original data, insert date, author, etc) for citations and other purposes.

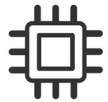
Used for similarity searches, recommendation engines, anomaly detection, categorization, forecasting, and… RAG!

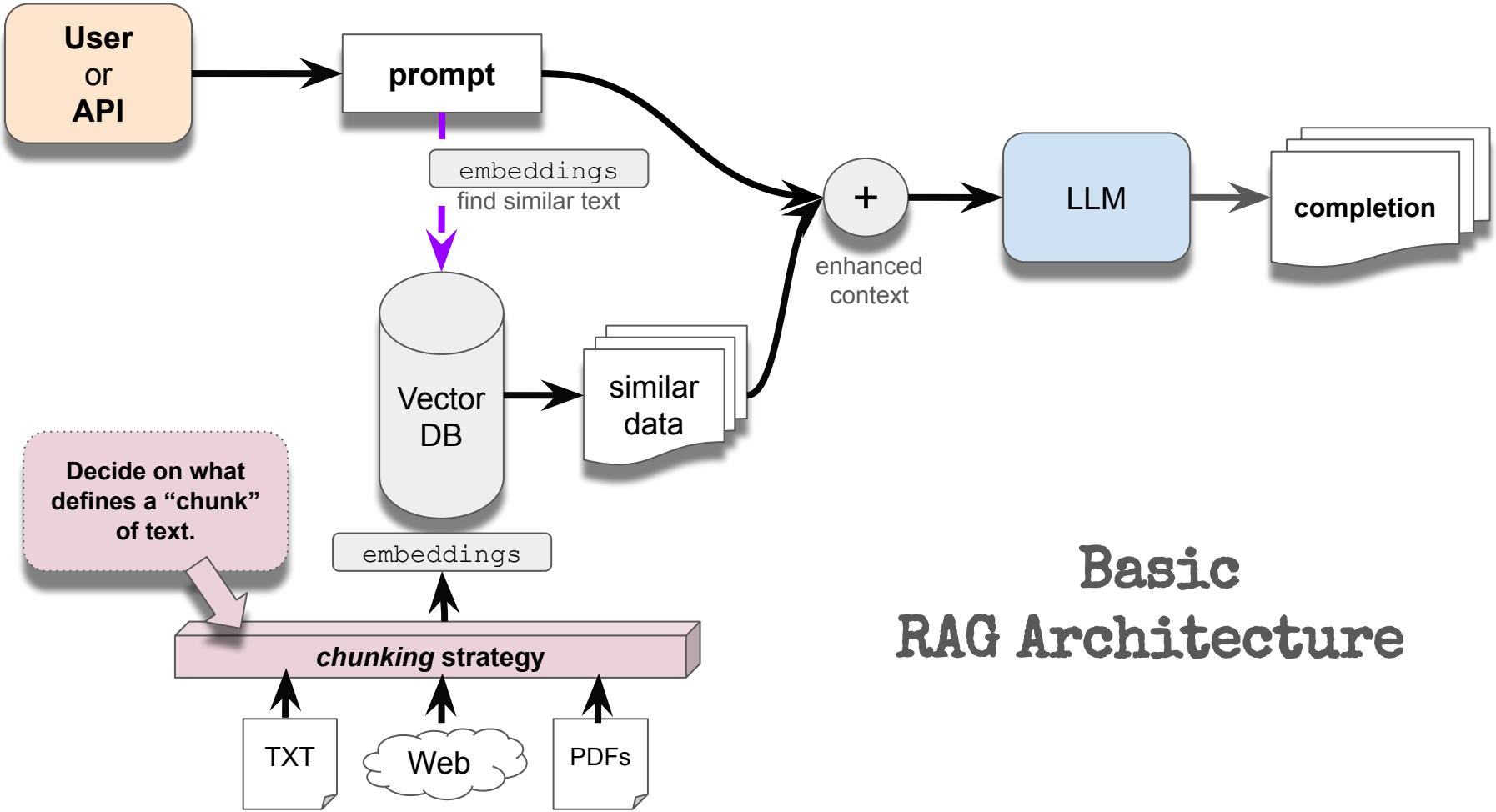
Vector Database (VDB)



Lots of VDBs

Chroma





Basic RAG Architecture

We need a "Chunking" Strategy

A RAG system needs to know about **relevant** text to provide context in the prompt sent to the LLM.

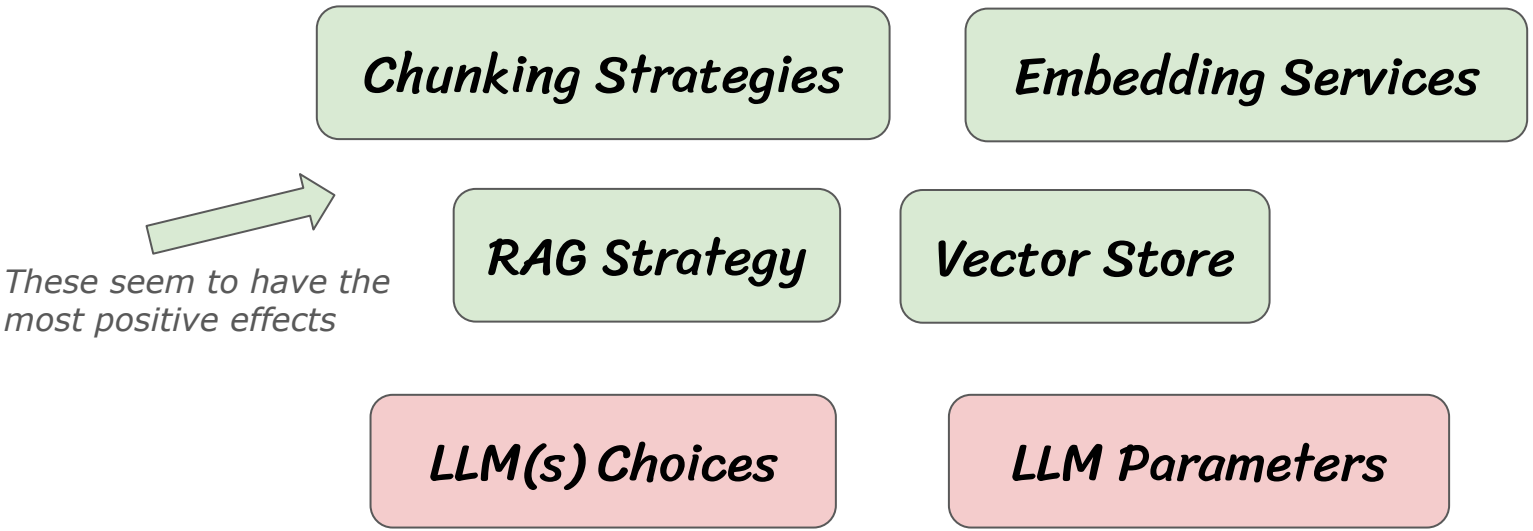
This helps to improve **accuracy** .

Due to context-window constraints and costs, it is not always feasible to send huge "chunks" of text.

Character limit?
Entire documents?
Sentences?
Paragraphs?
Phrases?
Markdown? HTML? JSON?
Recursive doc summarization chunkers?

Smaller chunks may be good for granular queries.
Larger chunks may be good for more in-depth queries.
Need to monitor/evaluate!

Performance of LLMs




Searching an
EmbeddingStore
©Frank Greco

Searching the EmbeddingStore

LOW-LEVEL - more control, more complex

1. Get embedding vector for target String
2. Create a EmbeddingSearchRequest with that vector
3. search(EmbeddingSearchRequest)
4. Retrieve matches

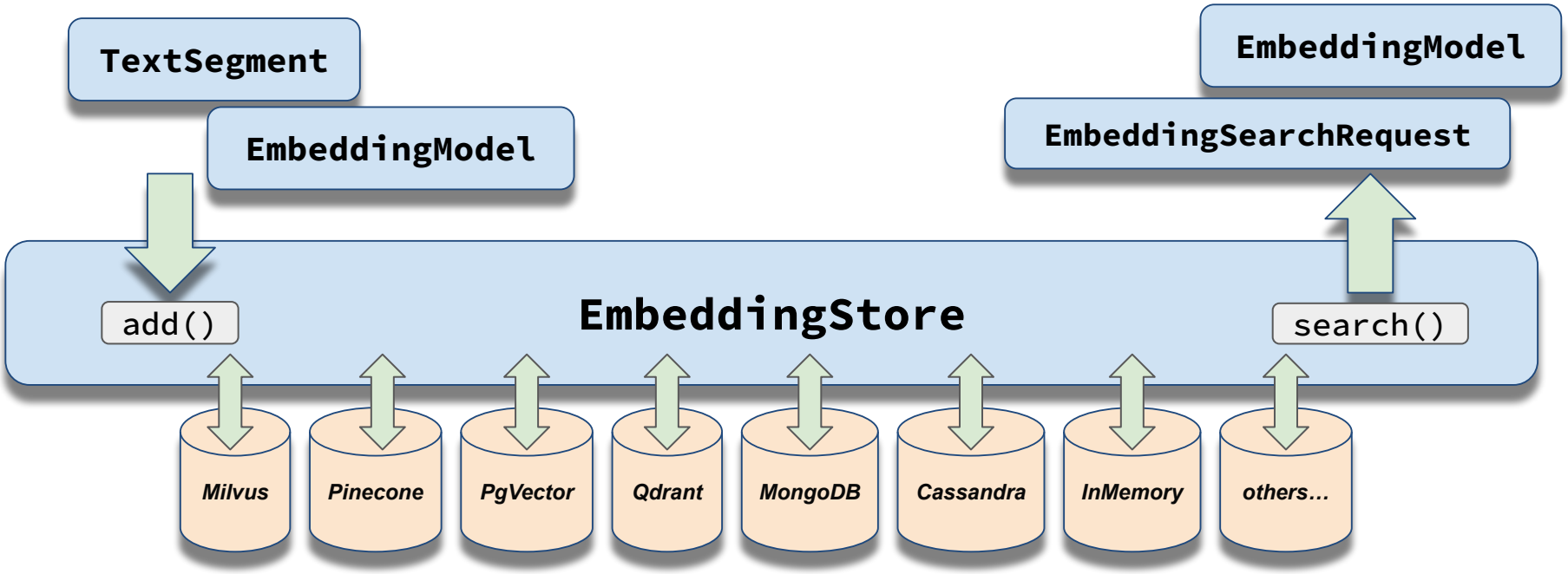
HIGH-LEVEL - easier

1. Create EmbeddingStoreContentRetriever
2. Associate with AiService

Using an EmbeddingStore

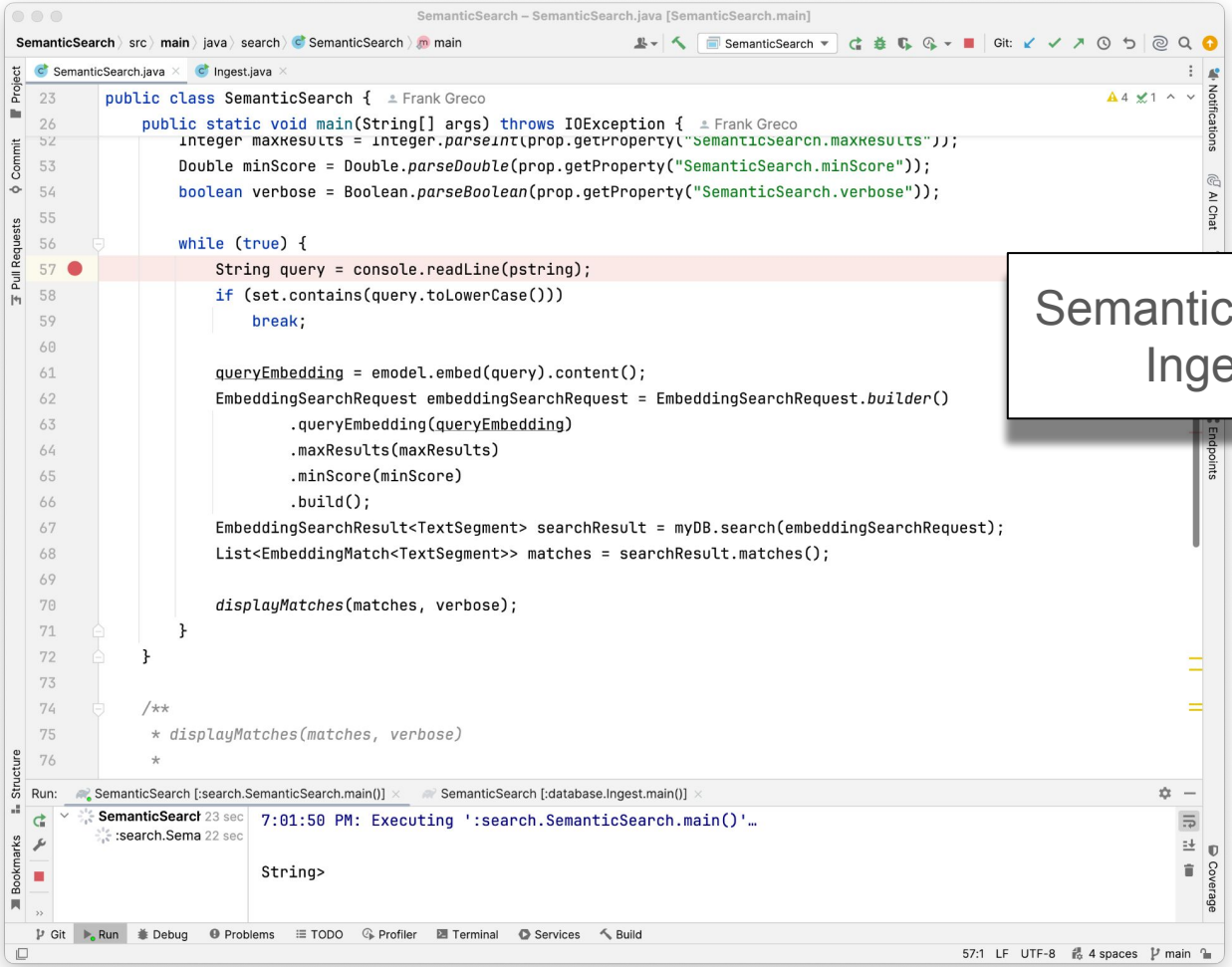
Low-Level

EmbeddingStore - Low-Level



LangChain4j


Semantic Search
Demo
©Frank Greco



SemanticSearch.java
Ingest.java

Thanks!

Questions / Comments?

LinkedIn Learning

Foundations of AI/ML for Java Developers

Introduction to Building GenAI Java Applications using LangChain4j

Pearson/OReilly

GenAI for Busy Java Developers

@frankgreco