



From ChatGPT User to RAG Implementer: A Developer's Journey

I'm Pasha

And this is my story

I'm Pasha

I used to be a ChatGPT user

I'm Pasha

I used to be a ChatGPT user

And I didn't really care about its internals

I'm Pasha

I used to be a ChatGPT user

And I didn't ~~really~~ care about its internals

I'm Pasha

I used to be a ChatGPT user

And I didn't ~~really~~ care about its internals

I didn't need to know much about LLMs

I'm Pasha

I used to be a ChatGPT user

And I didn't ~~really~~ care about its internals

I didn't need to know much about LLMs

I didn't hear about RAG

I'm Pasha

I used to be a ChatGPT user

And I didn't ~~really~~ care about its internals

I didn't need to know much about LLMs

I didn't hear about RAG (Retrieval-augmented generation)

Who is like me here?

Who is like me here?

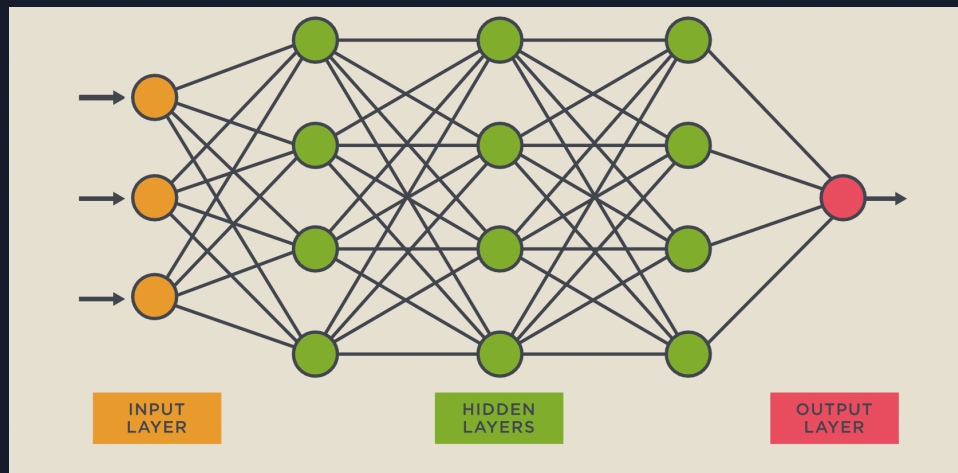
I will cross some T's in this talk

Of course...

I heard about neural networks

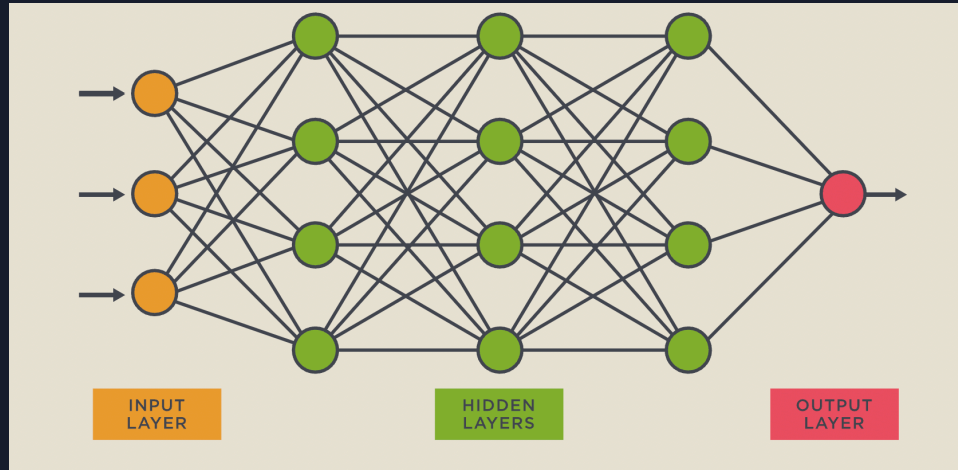
Of course...

I heard about neural networks



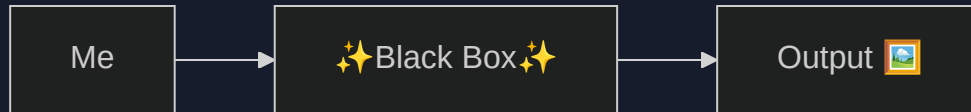
Of course...

I heard about neural networks

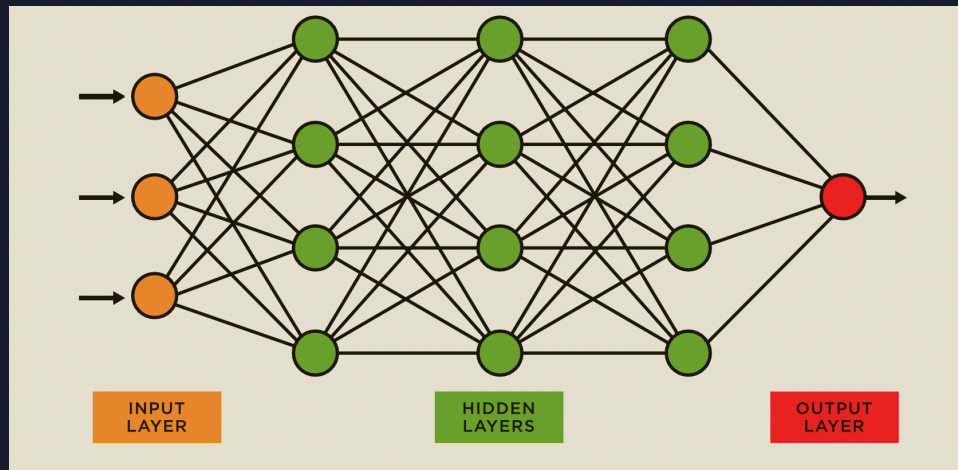
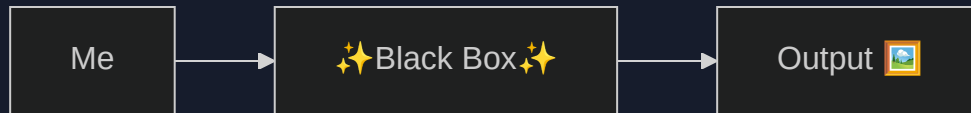


But what the heck is this?

How I perceived LLMs



How I perceived LLMs



Time to tell about me

1. I work with JVM languages for almost 15 years
2. I used to be a
 1. Backend developer
 2. Data engineer
 3. Engineering manager
3. Worked with data scientists, product managers, and other stakeholders
4. Usually did performance optimizations
5. Worked with Spring and without it

And now...


1. Developer Advocate at BellSoft
2. Have a lot of time to experiment
3. Need to produce content for the community
4. Still love to code

And now...

1. Developer Advocate at BellSoft
2. Have a lot of time to experiment
3. Need to produce content for the community
4. Still love to code

Follow me, BTW!

 asm0dey

 asm0di0

 @asm0dey@fosstodon.org

One day...

I founds that we have A LOT of documents:

1. Blog posts
2. Documentation
3. Whitepapers

And I just can't find my way through them

One day...

I founds that we have A LOT of documents:

1. Blog posts
2. Documentation
3. Whitepapers

And I just can't find my way through them

So I decided to do something about it

Looking for solution

Looking for solution

- Elasticsearch?

Looking for solution

- Elasticsearch?
- Google Desktop?

Looking for solution

- Elasticsearch?
- ~~Google Desktop?~~

Looking for solution

- Elasticsearch?
- ~~Google Desktop?~~
- ``grep``?

WHO ARE WE DEVELOPERS!



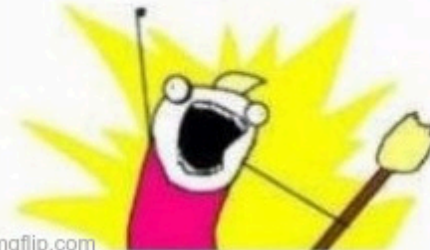
WHAT SHOULD WE DO?

SOLVE PROBLEMS!



WHAT WILL WE DO?

YAK SHAVING!



Let's use AI!

But how?

Luckily I have a friend

Who I could ask stupid questions (thanks @shrimpsizemoose)

Luckily I have a friend

Who I could ask stupid questions (thanks @shrimpsizemoose)

What did I ask?

Luckily I have a friend

Who I could ask stupid questions (thanks @shrimpsizemoose)

What did I ask?

How do I search thru my documents? You do this neural network magic for years, right?

Luckily I have a friend

Who I could ask stupid questions (thanks @shrimpsizemoose)

What did I ask?

How do I search thru my documents? You do this neural network magic for years, right?

He's a good friend, so he didn't answer, but asked what I know

Luckily I have a friend

Who I could ask stupid questions (thanks @shrimpsizemoose)

What did I ask?

How do I search thru my documents? You do this neural network magic for years, right?

He's a good friend, so he didn't answer, but asked what I know

I:

Well, word2vec, layers, embedding!

Luckily I have a friend

Who I could ask stupid questions (thanks @shrimpsizemoose)

What did I ask?

How do I search thru my documents? You do this neural network magic for years, right?

He's a good friend, so he didn't answer, but asked what I know

I:

Well, word2vec, layers, embedding!

He:

Explain me embeddings then

Luckily I have a friend

Who I could ask stupid questions (thanks @shrimpsizemoose)

What did I ask?

How do I search thru my documents? You do this neural network magic for years, right?

He's a good friend, so he didn't answer, but asked what I know

I:

Well, word2vec, layers, embedding!

He:

Explain me embeddings then

I couldn't

Embeddings

Here's an embedding

```
^[0.123, -0.456, 0.789, -0.234, 0.567, -0.890, 0.345, -0.678, 0.901, -0.432]^
```

And one more:

```
^[0.234, -0.567, 0.890, -0.123, 0.456, -0.789, 0.321, -0.654, 0.987, -0.345]^
```

What do they mean?

We do not know. Nothing in this context

What is an embedding?

An embedding is a way to represent something (like a word, image, or document) as a list of numbers

What is an embedding?

An embedding is a way to represent something (like a word, image, or document) as a list of numbers

Think of it like GPS coordinates:

- "New York" → (40.7128° N, 74.0060° W)
- "Tokyo" → (35.6762° N, 139.6503° E)

What is an embedding?

An embedding is a way to represent something (like a word, image, or document) as a list of numbers

Think of it like GPS coordinates:

- "New York" → (40.7128° N, 74.0060° W)
- "Tokyo" → (35.6762° N, 139.6503° E)

Just like coordinates tell us where cities are in physical space...

Embeddings tell us where things are in "meaning space" 🤖

What is an embedding?

An embedding is a way to represent something (like a word, image, or document) as a list of numbers

Think of it like GPS coordinates:

- "New York" → (40.7128° N, 74.0060° W)
- "Tokyo" → (35.6762° N, 139.6503° E)

Just like coordinates tell us where cities are in physical space...

Embeddings tell us where things are in "meaning space" 🤖

Similar things should have similar coordinates:

- "cat" and "kitten" would be close together
- "cat" and "rocket" would be far apart

Example: Family relationships in meaning space

Example: Family relationships in meaning space

Let's look at a famous example:

"king" - "man" + "woman" \approx "queen"

Example: Family relationships in meaning space

Let's look at a famous example:

"king" - "man" + "woman" \approx "queen"

Similarly:

"father" - "man" + "woman" \approx "mother"

Example: Family relationships in meaning space

Let's look at a famous example:

"king" - "man" + "woman" \approx "queen"

Similarly:

"father" - "man" + "woman" \approx "mother"

This shows that embeddings capture relationships:

- The difference between "father" and "mother" is similar to the difference between "man" and "woman"
- The "parent" concept stays constant while the gender changes

Example: Family relationships in meaning space

Let's look at a famous example:

"king" - "man" + "woman" \approx "queen"

Similarly:

"father" - "man" + "woman" \approx "mother"

This shows that embeddings capture relationships:

- The difference between "father" and "mother" is similar to the difference between "man" and "woman"
- The "parent" concept stays constant while the gender changes

These relationships emerge naturally when AI models learn from text!

Word2Vec: A Breakthrough in Word Embeddings

Word2Vec: A Breakthrough in Word Embeddings

Word2Vec was introduced by Google researchers in 2013:

- First major breakthrough in creating meaningful word embeddings
- Made it practical to capture word relationships in vector space

How Word2Vec Works

How Word2Vec Works

The core idea is learning from context:

- Predicts words that appear near each other
- If words often appear in similar contexts, they get similar embeddings

How Word2Vec Works

The core idea is learning from context:

- Predicts words that appear near each other
- If words often appear in similar contexts, they get similar embeddings

For example:

- "cat" and "dog" often appear near words like:
 - "pet"
 - "food"
 - "vet"

Word2Vec's Key Innovations

Word2Vec's Key Innovations

Technical breakthroughs:

- Much faster training than previous methods
- Produced higher quality embeddings

Word2Vec's Key Innovations

Technical breakthroughs:

- Much faster training than previous methods
- Produced higher quality embeddings

Conceptual breakthrough:

- Showed that simple neural networks could capture complex meaning

Word2Vec's Impact

Word2Vec's Impact

Changed the field of NLP:

- Sparked a revolution in natural language processing
- Laid groundwork for modern language models
- Still used today in many applications

Word2Vec Results

Word2Vec Results

The output is word embeddings:

- Each word becomes a dense vector of numbers
- Similar words have similar vectors
- Vector math captures semantic relationships

Word2Vec Results

The output is word embeddings:

- Each word becomes a ~~dense vector of numbers~~
- Similar words have similar vectors
- Vector math captures semantic relationships

Properties of the vectors:

- Typically 100-300 dimensions
- Enable measuring word similarity
- Can be visualized in lower dimensions

Word2Vec's Limitations

Technical limitations:

- Fixed context window misses broader document meaning
- Cannot handle polysemy (same word with different meanings)
- Requires pre-trained embeddings for each word

Modern Alternatives

Superseded by newer architectures:

- BERT and other transformers learn contextual embeddings
- Large language models capture richer relationships
- Modern models handle multiple word meanings

Text Embeddings

Similar concept, but for chunks of text:

Instead of: ``"cat"`` → ``[0.1, 0.2, -0.3, ...]``

We get: ``"The cat sat on the mat"`` → ``[0.4, -0.2, 0.1, ...]``

Benefits

- Captures meaning of entire passages
- Similar texts get similar vectors
- Can compare documents, paragraphs, or sentences

Use Cases

- Semantic search (find similar documents)
- Document clustering
- Question answering
- Text classification

How Text Embeddings Work

How Text Embeddings Work

- Split text into tokens

How Text Embeddings Work

- Split text into tokens
- Process through neural network layers:
 - Embedding layer converts tokens to vectors
 - Attention layers capture relationships
 - Feed-forward layers transform data
 - Pooling layers combine information

How Text Embeddings Work

- Split text into tokens
- Process through neural network layers:
 - Embedding layer converts tokens to vectors
 - Attention layers capture relationships
 - Feed-forward layers transform data
 - Pooling layers combine information
- Combine token representations

How Text Embeddings Work

- Split text into tokens
- Process through neural network layers:
 - Embedding layer converts tokens to vectors
 - Attention layers capture relationships
 - Feed-forward layers transform data
 - Pooling layers combine information
- Combine token representations
- Output fixed-size vector

Popular models

- OpenAI's `text-embedding-ada-002`
- Sentence-BERT
- Universal Sentence Encoder

Why do we need embeddings?

Why do we need embeddings?

- Embeddings convert text into numbers that capture meaning

Why do we need embeddings?

- Embeddings convert text into numbers that capture meaning
- Similar texts get similar vectors, enabling semantic search

Why do we need embeddings?

- Embeddings convert text into numbers that capture meaning
- Similar texts get similar vectors, enabling semantic search
- RAG workflow:
 1. Convert documents to embeddings and store them
 2. Convert user query to embedding
 3. Find most similar document embeddings
 4. Feed relevant documents to LLM as context

Why do we need embeddings?

- Embeddings convert text into numbers that capture meaning
- Similar texts get similar vectors, enabling semantic search
- RAG workflow:
 1. Convert documents to embeddings and store them
 2. Convert user query to embedding
 3. Find most similar document embeddings
 4. Feed relevant documents to LLM as context
- This is not only a search, but also summarization!

But can't we just...

save data in a database and ask LLM to access it?

But can't we just...

save data in a database and ask LLM to access it?

No!

- The whole NN is a huge bunch of hardcode

But can't we just...

save data in a database and ask LLM to access it?

No!

- The whole NN is a huge bunch of hardcoded weights

But can't we just...

save data in a database and ask LLM to access it?

No!

- The whole NN is a huge bunch of hardcoded weights

But can't we just...

save data in a database and ask LLM to access it?

No!

- The whole NN is a huge bunch of hardcoded weights
- NN can't access anything directly, it's not a program, more of a data structure

And here I hear the magic word

RAG (Retrieval Augmented Generation)

Essentially, the idea is to mix relevant data into prompt and make LLM use it

How to implement RAG?

1. Convert documents to embeddings
2. Store them in a vector database
3. Convert user query to embedding
4. Find most similar document embeddings
5. Feed relevant documents to LLM as context

1. Convert documents to embeddings

- Not all documents can be converted to embeddings directly
 - Text needs to be split into chunks
 - Each chunk has a maximum token limit:
 - OpenAI ada-002: 8,191 tokens
 - Mistral-7B: 8,192 tokens
 - Claude: 8,000 tokens
 - Need to balance chunk size:
 - Too small → loses context
 - Too large → less precise matches
- Some information may be lost in the conversion process

2. Store them in a vector database

Where?

Vector database is a specialized database for storing and searching vectors

Key features:

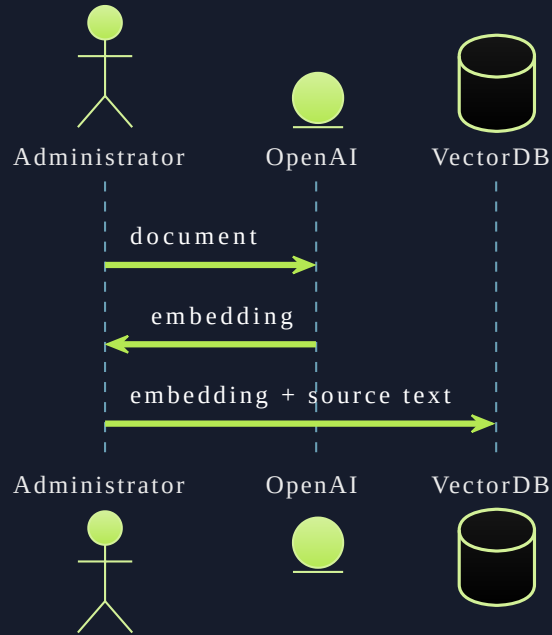
- Efficient similarity search:
 - Cosine similarity: $\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$
 - Euclidean distance: $\sqrt{\sum (a_i - b_i)^2}$
 - Dot product: $\sum_i a_i b_i$
- Optimized for high-dimensional data
- Can store metadata alongside vectors

2. Store them in a vector database

For example

- Chroma
- Weaviate
- Pinecone
- Milvus
- pgvector (PostgreSQL extension)

This is how it works



3. Convert user query to embedding

Same as with document.

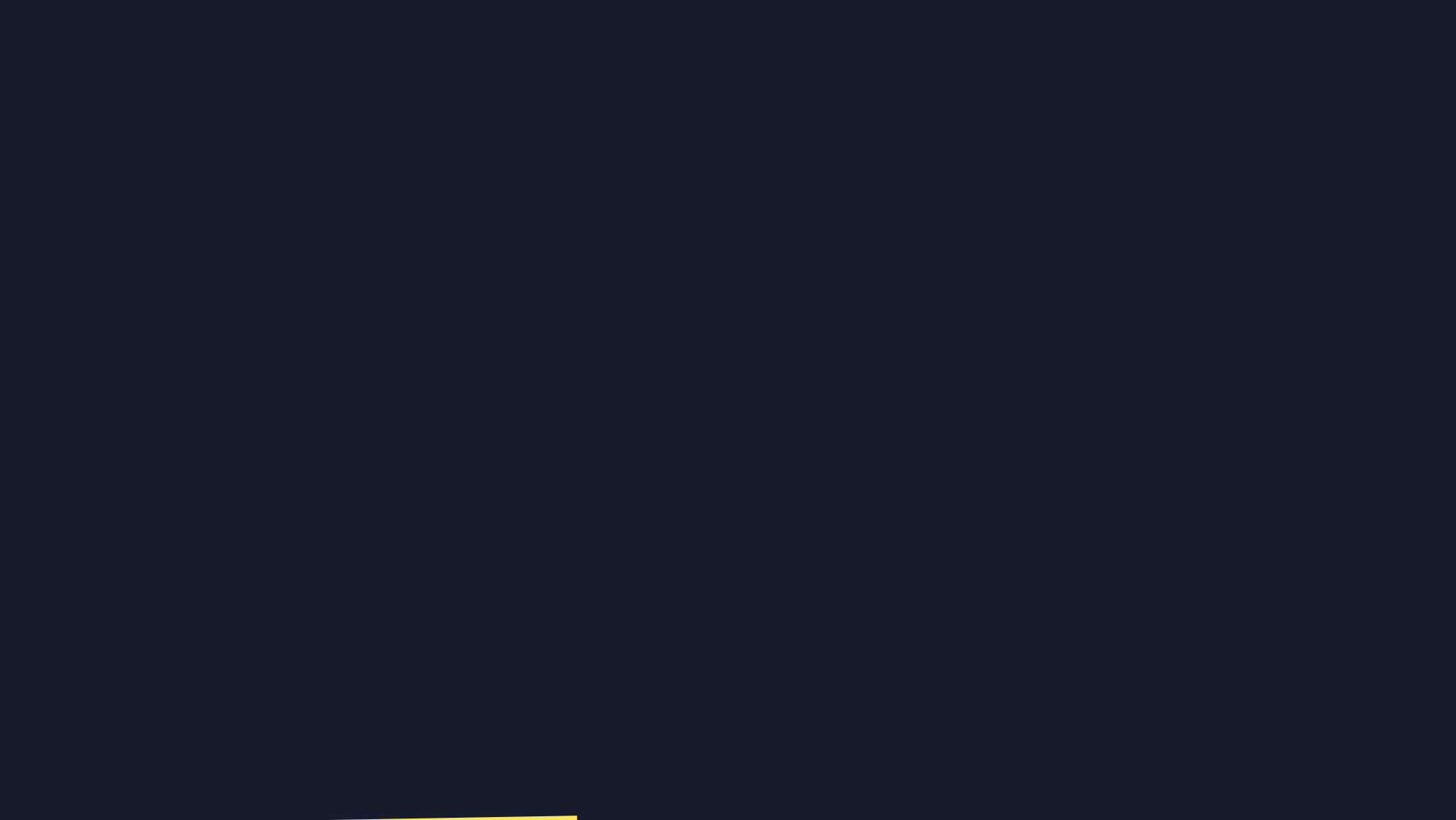
Probably return an error if the query is too big.

4. Find most similar document embeddings

- Use vector database to find closest embeddings to query embedding
- Usually returns:
 - Distance/similarity score
 - Original text
 - Optional metadata
- Can limit number of results (e.g. top-k)
- Can filter by metadata (e.g. only search specific document types)

5. Feed relevant documents to LLM as context

- Take relevant documents from vector search
- Add them as context to LLM prompt
- Ask LLM to answer based on provided context
- LLM generates response using only provided context
- Response is more accurate and grounded in your documents





Demo!

Summary

- RAG is simple
- You need to know embeddings to understand RAG
- Embeddings are not magic, they are just vectors
- You don't need to know how LLM works to use RAG

Questions?

 asm0dey

 asm0di0

 @asm0dey@fosstodon.org

Blog:



Source code:



