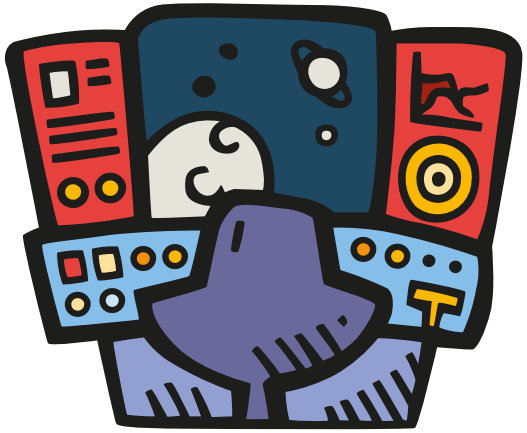
Assist AI

Context

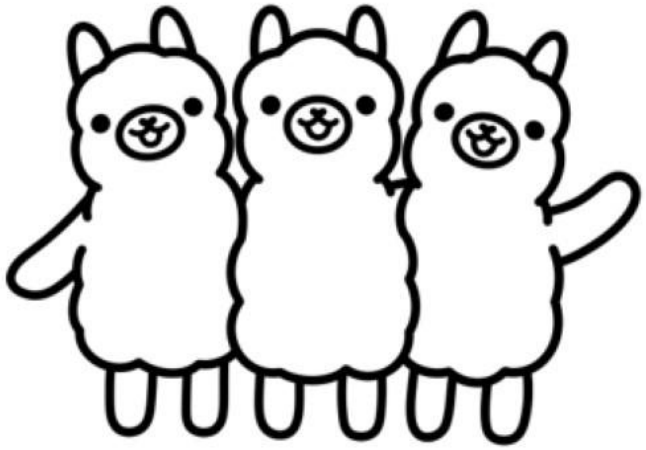
- Development of generative AI
- Usage of ChatGPT
- Desire for custom model
- Second brain applications
- Desire for knowledge

The mission



Local AI

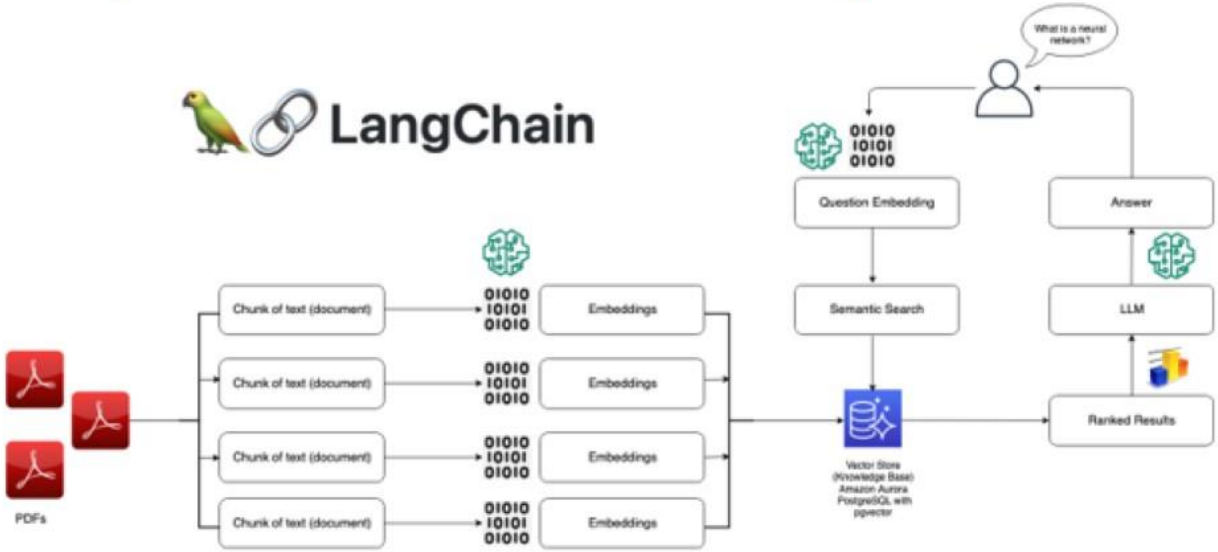
- Full control
- No files shared



```
ollama run llama3
```

File reader

- Easily add files
- Answer questions based on your files



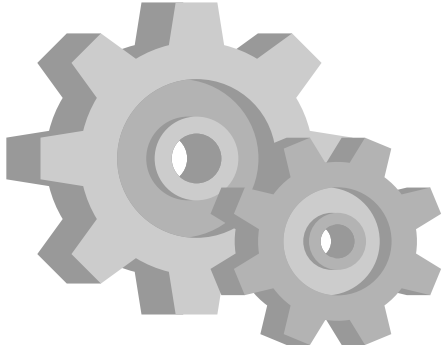
Personal assistant

- Increased work efficiency
- Increased memory recollection

How It's Made

How does a large language model work?

- Training data: information used to teach the model
- Training: find dependencies (patterns) between text sequences using transformers
- Input encoding: converting textual input to vectorised inputs
- Pattern matching: matching vectors and patterns to data
- Response decoding: converting resulting vectorised output into text, word by word

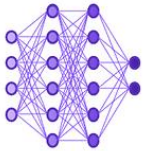


Natural language generation

After training: We can generate text by predicting one word at a time

A trained language model can

Input



LLM

Word	Probability
speak	0.065
generate	**0.072**
politics	0.001
...	...
walk	0.003

Output at step 1

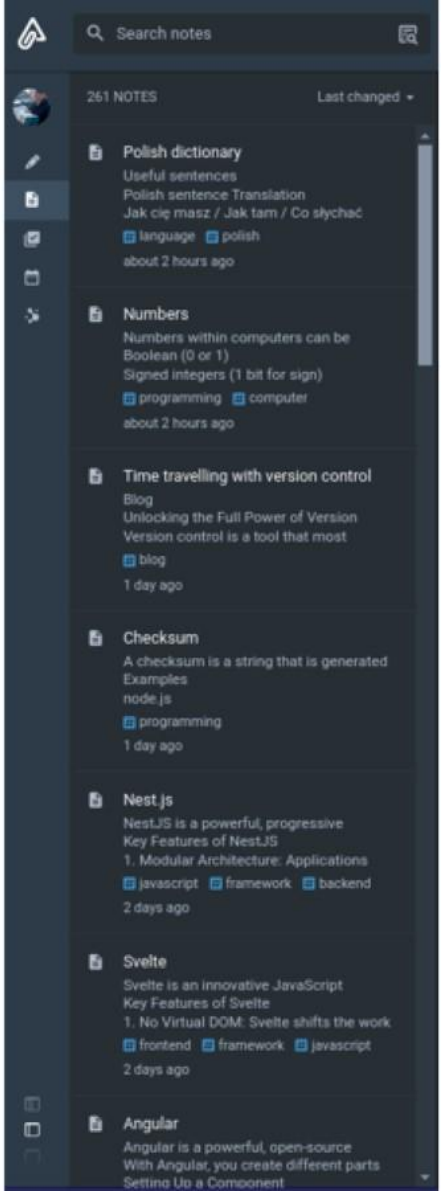
Word	Probability
ability	0.002
text	0.084
coherent	**0.085**
...	...
ideas	0.041

Output at step 2

LLMs are an example of what's called "Generative AI"

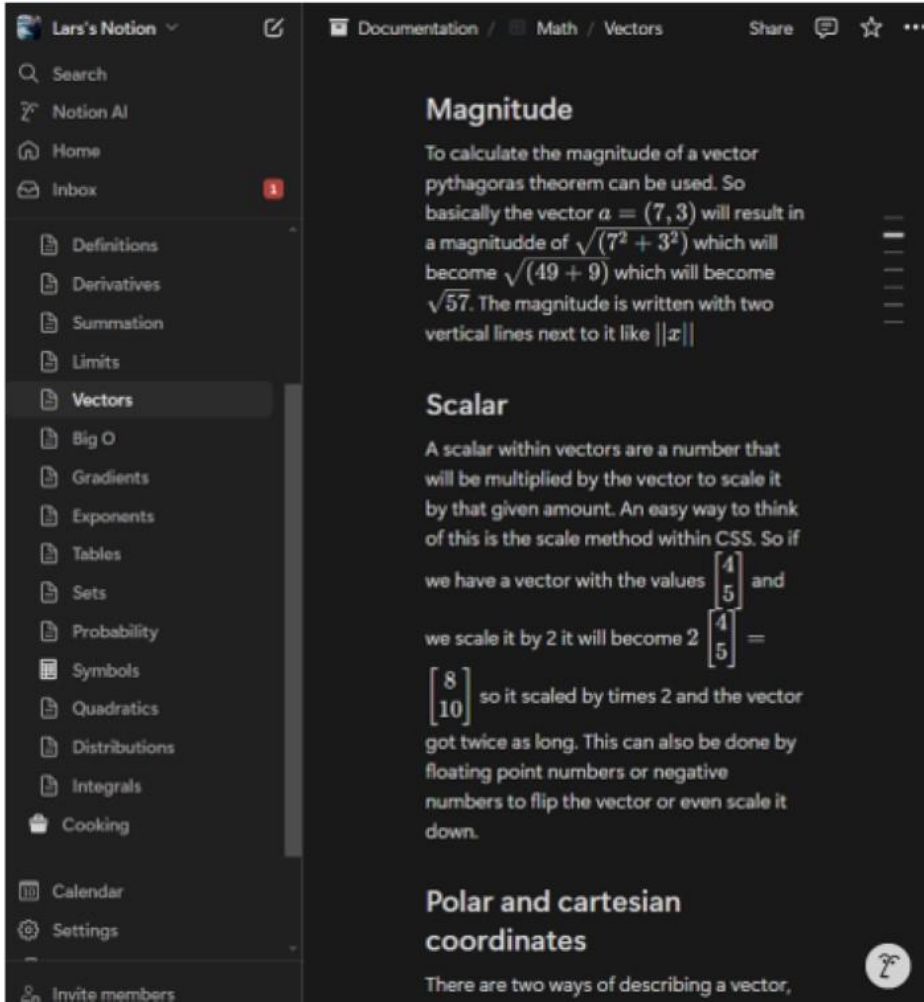
Source: Second brain

- Human memory
- Note taking apps vs paper
  - Longevity
  - Scalability
- Wiki



Note taking superpower

- Quick jots
- Advanced note taking
- Growth over time
- A library of your knowledge
- Over 250 notes taken in Amplenote and Notion



Model: How to make (with langchain)

1. Select model

2. Read documents

3. Split documents into small parts with overlap (strings of about 200 characters long)

4. Convert split documents into a vector representation using an embedding (for fast semantic search based on input)

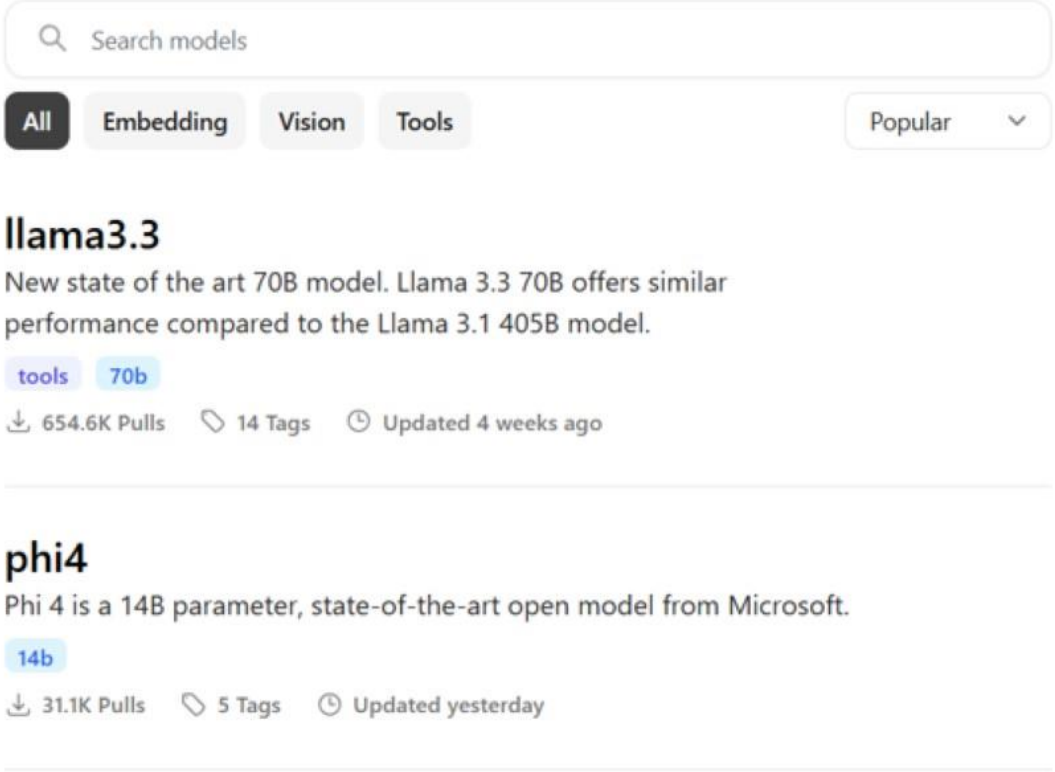
5. Store vector representations of split documents in vector store (local database)

6. Query an input, the input is also converted into a vector representation using the embedding so corresponding pieces of text can be found for context to the model which then answers based on this context


Step 1: Select chat model
Step 2: Write prompt template
Step 3: Load documents
Step 4: Split documents
Step 5: Select embedding model
Step 6: Vectorise text chunks
Step 7: Store vectorised chunks
Step 8: Create retrieval chain

Step 1: Select chat model

- Ollama
- Open-source models
  - Chat
  - Embedding



Step 2: Write prompt template

- Wrapping content
  - Input
  - History
  - Context

```
# Create basic prompt
self.prompt = ChatPromptTemplate.from_template("""
    You are an expert assistant specialized in answering questions based on provided documents. Use the **context** as your primary source of
    information, and if applicable, draw from the **chat history** for continuity.

    - Always prioritize the **context** for your answers.
    - If the **context** is unclear or insufficient, use the **chat history** to provide continuity.
    - Respond concisely and descriptively, ensuring clarity and relevance.
    - For technical questions or code-related inquiries, provide detailed examples where appropriate.

    **Context**: {context}
    **Chat History** (if relevant): {chat_history}
    **Question**: {input}

    Your response should only answer the question and be directly based on the context or history.
""")
```

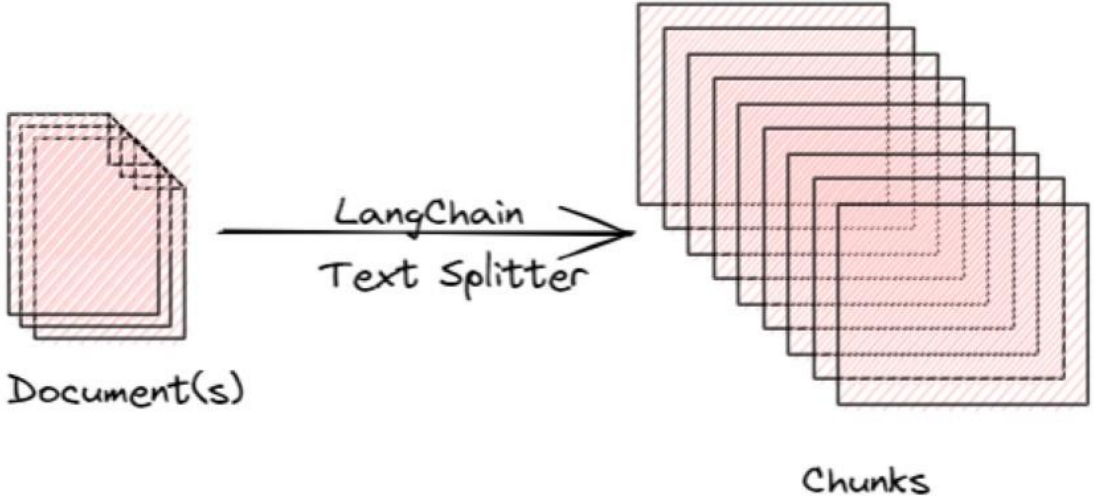
Step 3: Load documents

- Export notes from note taking app to textual form (.txt or .md)
- Download resources that you want to access

Arduino.md	12/11/2024 14:04	Markdown Source ...	1
Assembly.md	12/11/2024 14:04	Markdown Source ...	6
Astro.md	12/11/2024 14:04	Markdown Source ...	2
Authentication.md	12/11/2024 14:04	Markdown Source ...	2
Automate life.md	24/11/2024 20:23	Markdown Source ...	1
Back propagation.md	12/11/2024 14:04	Markdown Source ...	1
Backend.md	12/11/2024 14:04	Markdown Source ...	3
Best practises.md	12/11/2024 14:04	Markdown Source ...	4
Big O.md	12/11/2024 14:04	Markdown Source ...	3

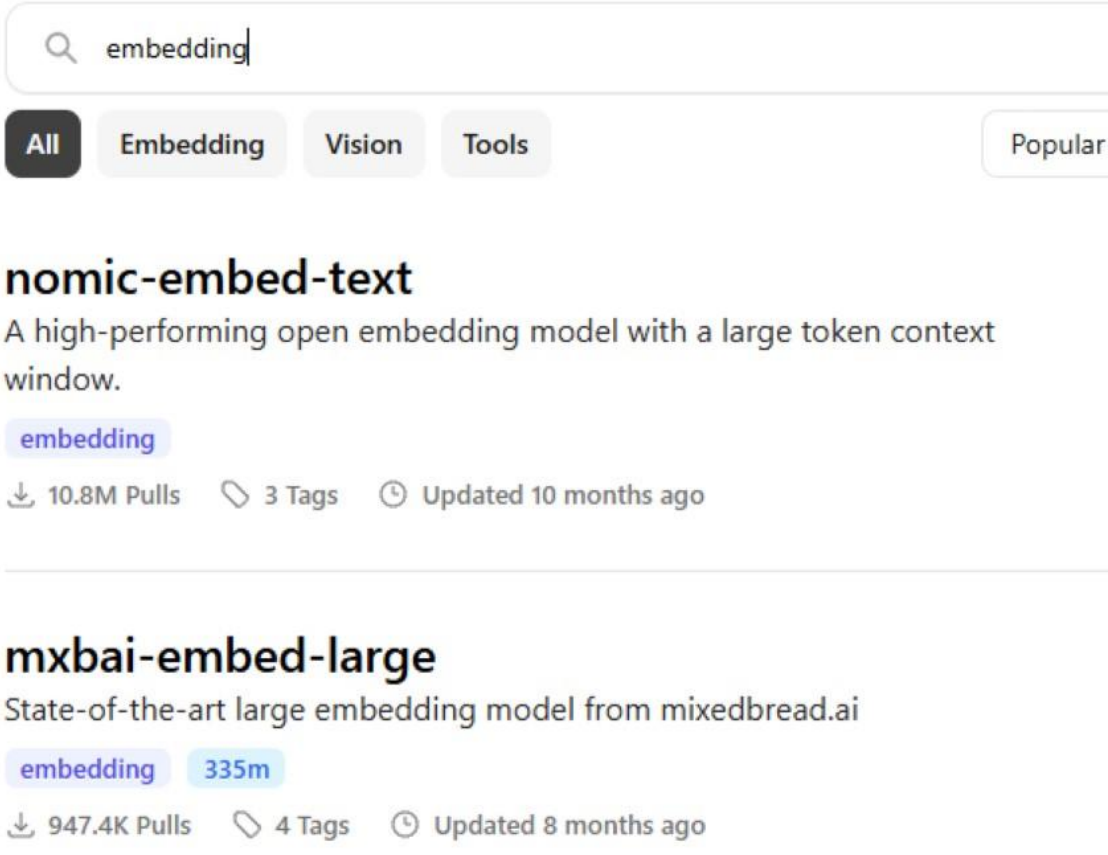
Step 4: Splitting documents

- Context size window
- Multiple documents
- Overlap



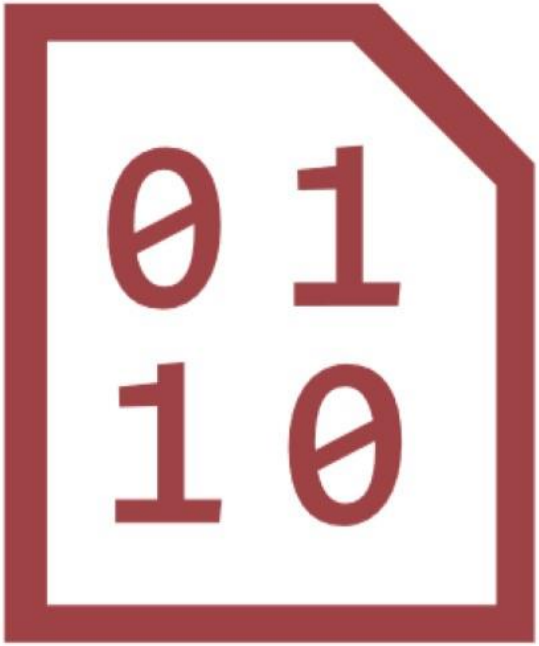
Step 5: Select embedding model

- Ollama
- Vectorising document chunks



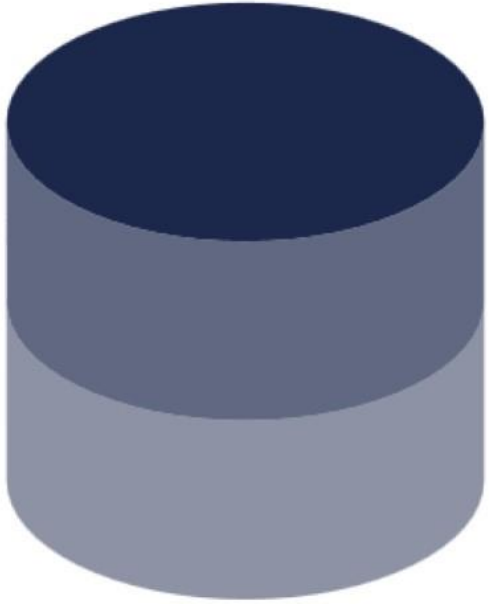
Step 6: Vectorise text chunks

- Text embedding / vectorisation
  - Convert in batches (OOM error)
- Vector store
  - Efficient storage
  - Retrieval chain



Step 7: Store vectorised chunks

- Vector store
  - Efficient storage
  - Retrieval chain



Step 8: Create retrieval chain

- Pattern matching
- Vector search in store

Results

- Technology used
- Langchain – For configuration
- Ollama – Running local model
- Llama – Chat model
- Mxbai-embed-large – Embedding model
- Faiss – vector store

Step 1: Invoke model

Step 2: Vectorise input

Step 3: Match input to vectorised chunks

Step 4: Inject text chunks as context

Step 5: Generate response

Invocation

- Preferred form
  - Graphical user interface
  - Command line tool
  - Online tool

GUI

- Simple electron application

CLI

- Using python and a shell script

```
larsvonk@LarsHackbook  ~  assist-ai "What is AI?"
"In the context of code generation tools like AI assistants, AI refers to th
e ability of these tools to analyze code and generate new code based on patt
erns and structures learned from existing code. This can be particularly use
ful for developers who write redundant or complex code, or when working with
 new programming languages that have unfamiliar syntax."
larsvonk@LarsHackbook  ~
```

Demo