

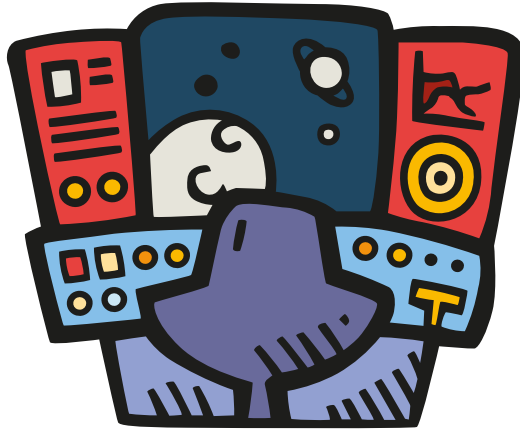
Assist AI

Context

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

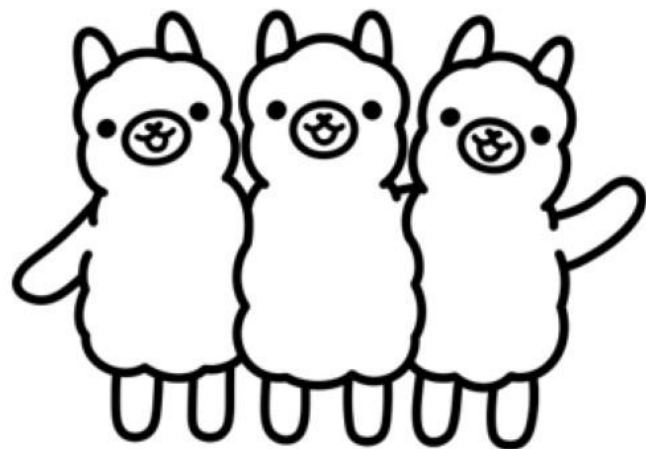


The mission



Local AI

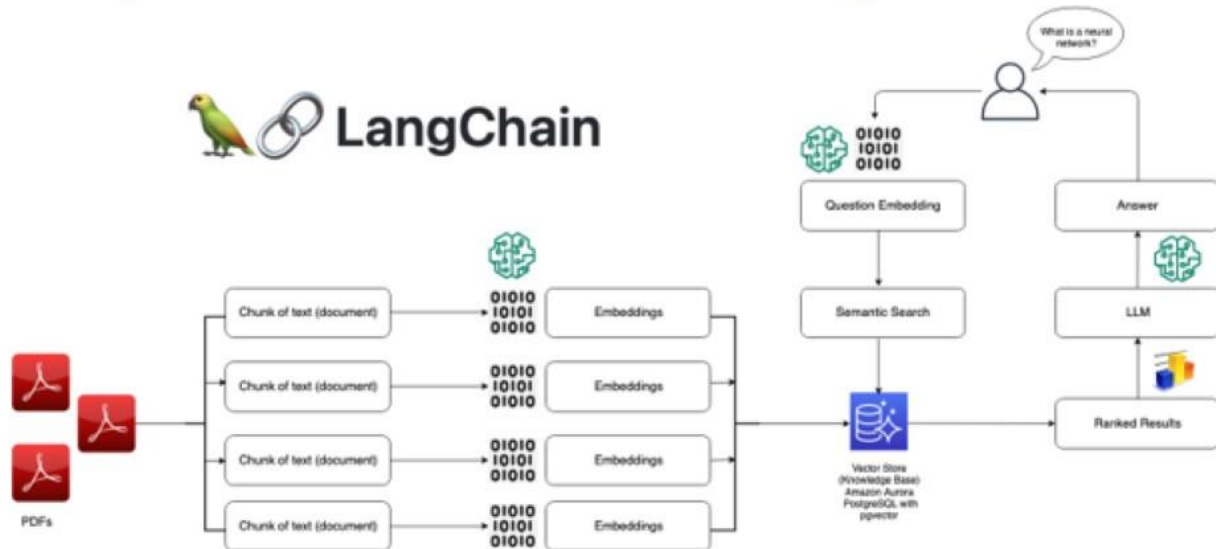
- Full control
- No files shared



```
ollama run llama3
```

File reader

- Easily add files
- Answer questions based on your files



Personal assistant

- Increased work efficiency
- Increased memory recollection

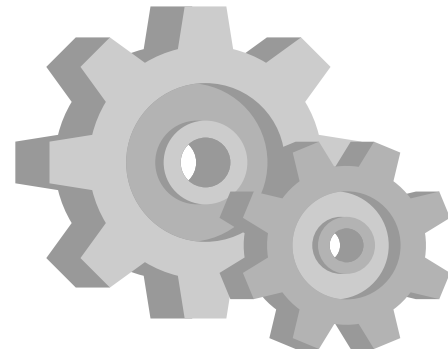




How It's Made

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

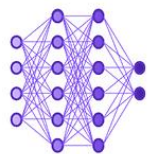


Natural language generation

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

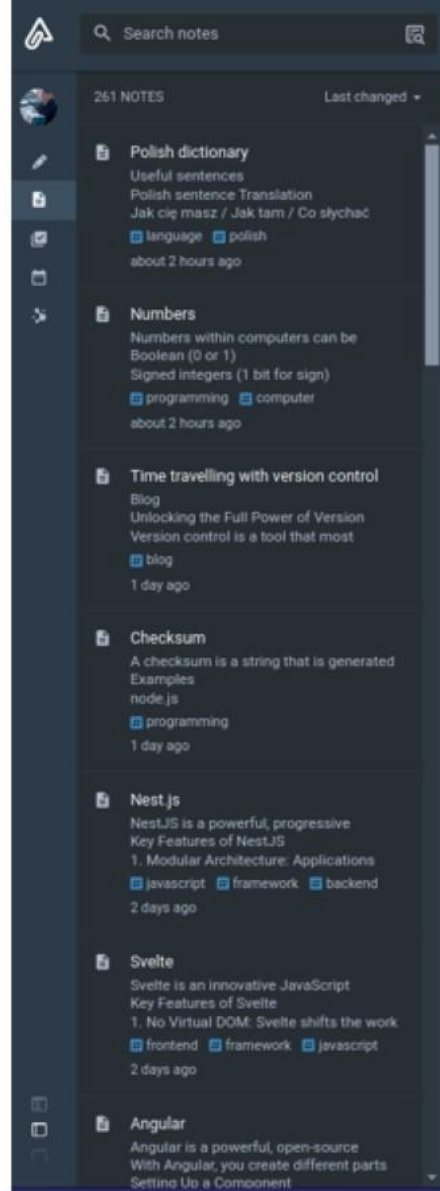
Word	Probability
ability	0.002
text	0.084
coherent	0.085
...	...
ideas	0.041

Output at step 2

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

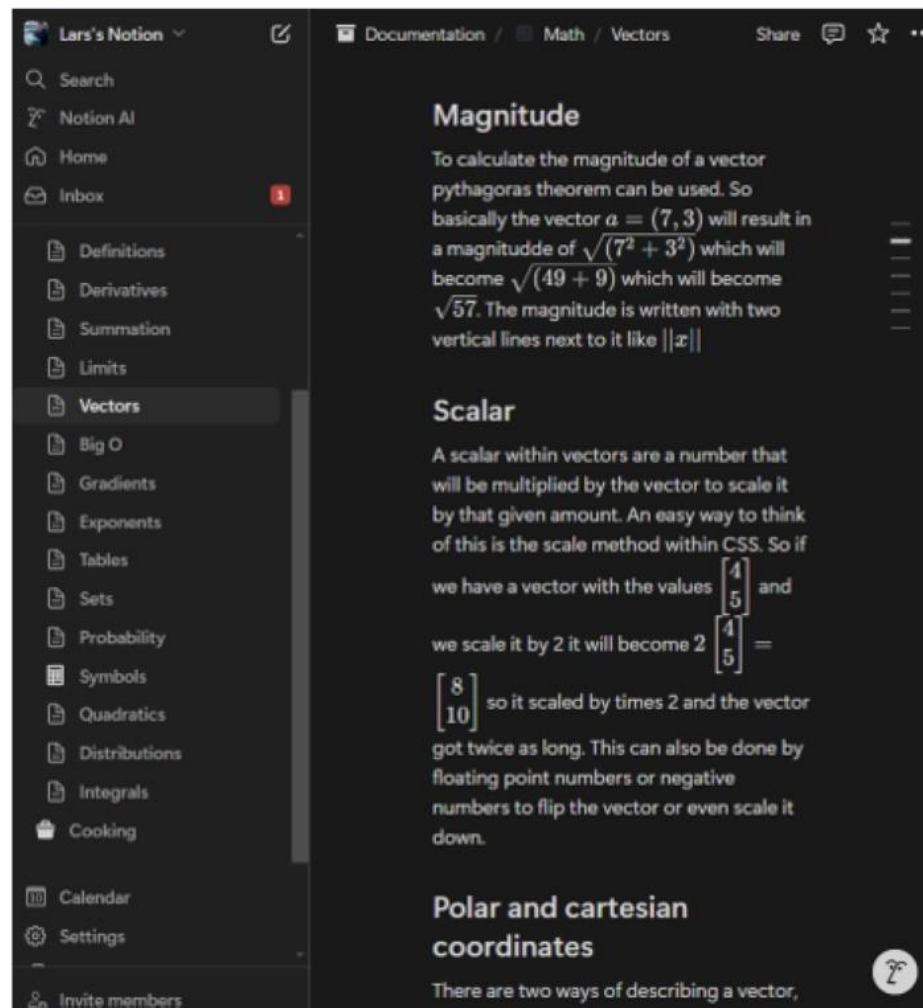
Source: Second brain

- 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



The screenshot shows the Notion application interface. On the left is a sidebar titled 'Lars's Notion' with a search bar and a list of topics: Definitions, Derivatives, Summation, Limits, Vectors (highlighted), Big O, Gradients, Exponents, Tables, Sets, Probability, Symbols, Quadratics, Distributions, Integrals, Cooking, Calendar, Settings, and Invite members. The main content area on the right is titled 'Documentation / Math / Vectors' and contains two sections: 'Magnitude' and 'Scalar'. The 'Magnitude' section explains how to calculate the magnitude of a vector using the Pythagorean theorem, with an example vector $a = (7, 3)$ resulting in a magnitude of $\sqrt{57}$. The 'Scalar' section explains that a scalar is a number that scales a vector, using an example where a vector $\begin{bmatrix} 4 \\ 5 \end{bmatrix}$ is scaled by 2 to become $\begin{bmatrix} 8 \\ 10 \end{bmatrix}$. The bottom right corner features a small circular icon with a stylized 'L'.

Lars's Notion

Search

Notion AI

Home

Inbox

Definitions

Derivatives

Summation

Limits

Vectors

Big O

Gradients

Exponents

Tables

Sets

Probability

Symbols

Quadratics

Distributions

Integrals

Cooking

Calendar

Settings

Invite members

Documentation / Math / Vectors

Share

Magnitude

To calculate the magnitude of a vector pythagoras theorem can be used. So basically the vector $a = (7, 3)$ will result in a magnitude of $\sqrt{7^2 + 3^2}$ which will become $\sqrt{49 + 9}$ which will become $\sqrt{57}$. The magnitude is written with two vertical lines next to it like $||x||$

Scalar

A scalar within vectors are a number that will be multiplied by the vector to scale it by that given amount. An easy way to think of this is the scale method within CSS. So if we have a vector with the values $\begin{bmatrix} 4 \\ 5 \end{bmatrix}$ and we scale it by 2 it will become $2 \begin{bmatrix} 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 8 \\ 10 \end{bmatrix}$ so it scaled by times 2 and the vector got twice as long. This can also be done by floating point numbers or negative numbers to flip the vector or even scale it down.

Polar and cartesian coordinates

There are two ways of describing a vector,

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

All

Embedding

Vision

Tools

Popular

llama3.3

New state of the art 70B model. Llama 3.3 70B offers similar performance compared to the Llama 3.1 405B model.

tools

70b

↓ 654.6K Pulls

🏷 14 Tags

🕒 Updated 4 weeks ago

phi4

Phi 4 is a 14B parameter, state-of-the-art open model from Microsoft.

14b

↓ 31.1K Pulls

🏷 5 Tags

🕒 Updated yesterday

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.
""")
```

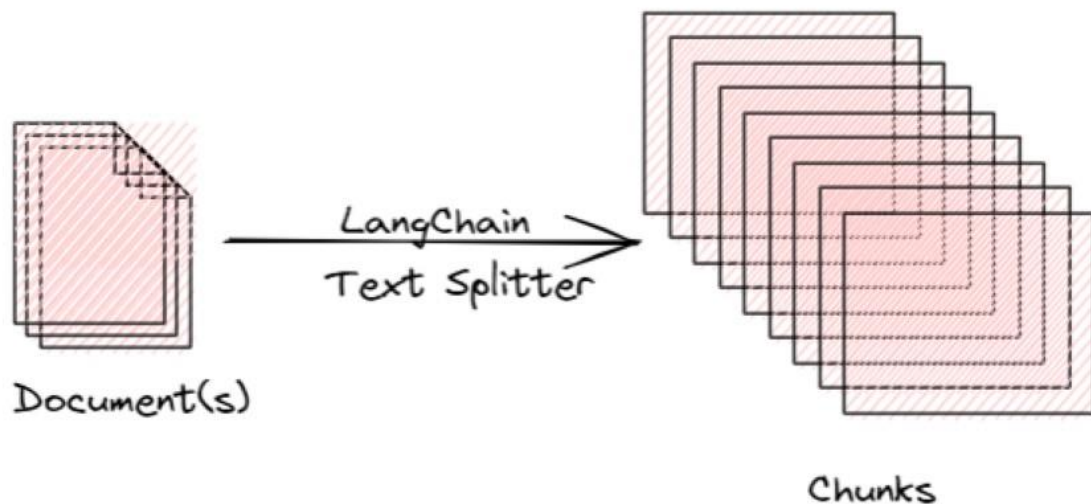
Step 3: Load documents

- 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

All

Embedding

Vision

Tools

Popular

nomic-embed-text

A high-performing open embedding model with a large token context window.

embedding

↓ 10.8M Pulls 3 Tags Updated 10 months ago

mxbai-embed-large

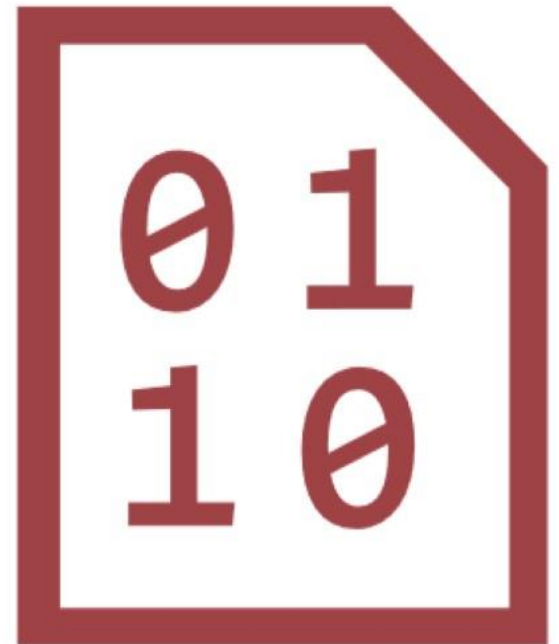
State-of-the-art large embedding model from mixedbread.ai

embedding 335m

↓ 947.4K Pulls 4 Tags Updated 8 months ago

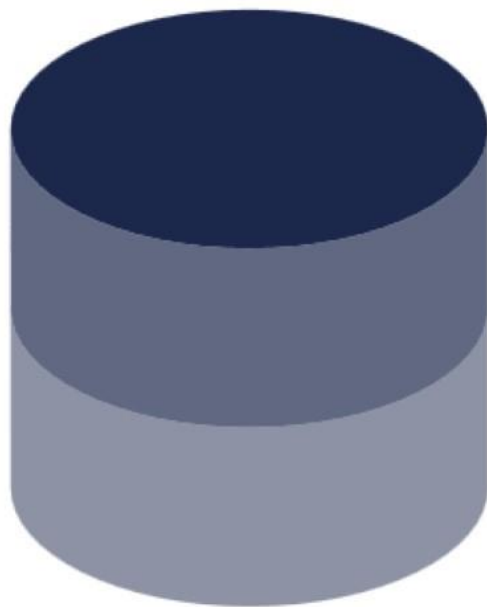
Step 6: Vectorise text chunks

- 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

AssistAI

Welcome to AssistAI! This client aims to provide a simple interface for interacting with the AssistAI chatbot and connect it to your documents.

```
controllers: [AppController],  
providers: [AppService],  
})  
export class AppModule {}
```

This code defines an `AppModule` that imports the `AppController` and `AppService`, which are then provided to Nest.js.

6. Using a module

```
import { Module } from '@nestjs/common';  
import { AppController } from './app.controller';  
import { AppService } from './app.service';  
  
@Module({  
  controllers: [AppController],  
  providers: [AppService],  
})  
export class AppModule {}
```

This code defines an `AppModule` that imports the `AppController` and `AppService`, which are then provided to Nest.js.

7. Using a module with multiple controllers

```
import { Module } from '@nestjs/common';  
import { AppController } from './app.controller1';  
import { AppController2 } from './app.controller2';  
  
@Module({  
  controllers: [AppController, AppController2],  
})  
export class AppModule {}
```

This code defines an `AppModule` that imports two separate controllers, `AppController1` and `AppController2`.

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 the ability of these tools to analyze code and generate new code based on patterns and structures learned from existing code. This can be particularly useful for developers who write redundant or complex code, or when working with new programming languages that have unfamiliar syntax."  
larsvonk@LarsHackbook ➤
```

Demo