Databases in the AI Trenches

Bruce Momjian

As a follow up to the presentation, *Postgres and the Artificial Intelligence Landscape*, this talk covers explains recent AI developments like generative AI and Rag.

https://momjian.us/presentations

Last updated: May 2025

Further Study

My previous talk, *Postgres and the Artificial Intelligence Landscape*, covered

- What is artificial intelligence?
- Machine learning and deep learning
- Machine learning demo and examples
- Why use a database?

https://momjian.us/main/presentations/extended.html#landscape

Outline

1. AI Explosion

OpenAI's ChatGPT was released on November 30, 2022. From ChatGPT4:

> *The release of ChatGPT was important because it made **advanced AI accessible** to the public, enabling natural, **human-like conversations**. It became a tool for boosting **productivity**, creativity, and learning. Additionally, it sparked **ethical discussions** about AI's societal impact and set the stage for future AI applications across industries.*

The best video series about generative AI is at https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi.

https://medium.com/@chloeng_22909/how-has-interest-and-coverage-around-ai-changed-since-chatgpt-502d415a3126

Discriminative/Predictive AI

Before OpenAI's GPT-3, most public AI was discriminative:

- Classification, e.g., fraudulent credit card charges, image recognition
- Recommendations, e.g., web pages, videos
- Predictive, e.g., trends
- Language translation

https://www.turing.com/kb/generative-models-vs-discriminative-models-for-deep-learning
https://solutionshub.epam.com/blog/post/generative-ai-vs-predictive-ai

Generative AI

OpenAI revolutionized the public's perception of what was possible with AI by releasing ChatGPT (for text) and Dall-E (for images) which showed the AI creation of *content*.

Generative AI Use Cases

- Text
  - chatbots
  - semantic/vector search
  - summarization
  - language translation
- Image/audio/video creation and search
- Software
  - code creation
  - code analysis
  - data analysis
  - programming language conversion *
  - natural language interface for tasks previously requiring specialized software

https://www.turing.com/resources/generative-ai-applications

https://www.youtube.com/watch?v=zjkBMFhNj_g
* https://www.codeconvert.ai/

Discriminative and Generative AI Compared

Discriminative	Generative
supervised learning requires training data with known outcomes	outcomes determined from massive training data
often problem-domain specific	usually general
output is a determination/prediction	output is new content

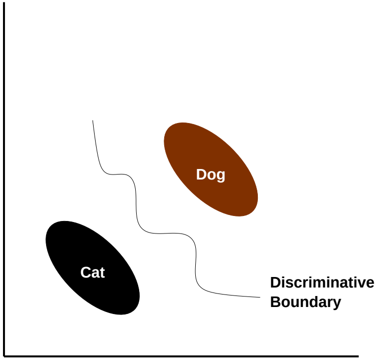
Discriminative unsupervised learning is similar to generative because it does not include known outcomes.

Discriminative and Generative AI Compared

*A generative algorithm models how the data was "**generated**", so you ask it "what's the likelihood this or that class generated this instance?" and pick the one with the better probability. A discriminative algorithm uses the data to create a **decision boundary**, so you ask it "what side of the decision boundary is this instance on?" So it doesn't create a model of how the data was generated, it makes a model of what it thinks the boundary between classes looks like. – Anthony, Pittsburgh*

https://stackoverflow.com/questions/879432/what-is-the-difference-between-a-generative-and-a-discriminative-algorithm

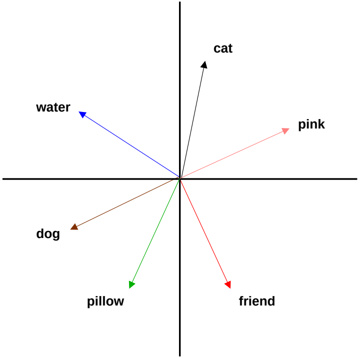
Discriminative Boundary



They focus on boundaries between groups.

https://solutionshub.epam.com/blog/post/generative-ai-vs-predictive-ai

Generative Vectors



The color of the arrows is meant to represent an aspect of the word, or an aspect of how the word appeared in the training data. It focuses on relationships between items.

2. Hyper-Dimensional Vectors: Math

Mathematics allows the representation of concepts that can't be represented in the real world:

- Numbers greater than the number of atoms in the universe (10^{80})
- Infinity (∞)
- Function derivatives calculate slopes using infinitesimally small adjustments
- Function integration involves the infinite sum of infinitesimally small areas

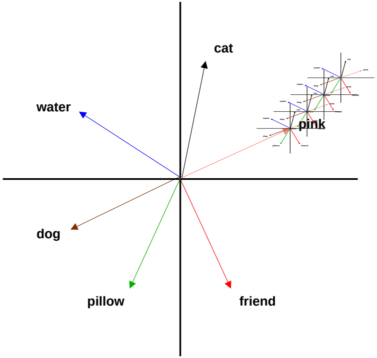
Duality

Mathematics allows concepts to be represented in multiple forms. For example, vectors can be thought of as representing the

- Physical world
- Computer science arrays
- Abstract representations

https://en.wikipedia.org/wiki/Duality_(mathematics)

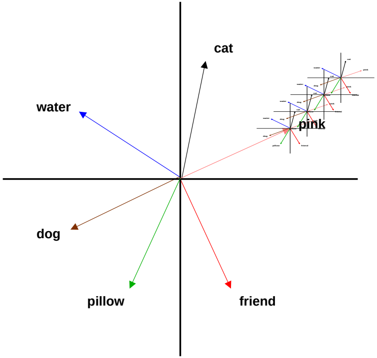
Hyper-Dimensional Vectors



Mathematics allows vectors in dimensions beyond the three-dimensional world to be used for AI. This is a two-dimensional image represents 10 dimensions. ChatGPT3's text embedding vectors can use 1024 to 12288 dimensions, meaning the image above would require 100-1,200 times more nesting.

https://community.openai.com/t/what-version-of-gpt-is-text-embedding-ada-002-based-on/404462

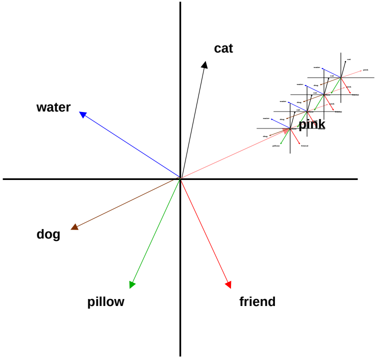
Hyper-Dimensional Vector Range



Each axis is typically a 4-byte floating point number. With 1k-12k dimensions, that yields a total range of $10^{9k}$to 10^{118k}values, *far* more than the number of atoms in the universe. (Even if we use only 4-byte floats between ± 1, and adjusting the the limited use of bits by floating point numbers, the minimum range is still 10^{8k}.)

https://stackoverflow.com/questions/7744016/how-many-distinct-values-can-be-stored-in-floating-point-formats

Hyper-Dimensional Vector Magnitude



Notice that each vector has the same length or magnitude.

https://postgresml.org/docs/open-source/pgml/guides/embeddings/vector-normalization
https://stackoverflow.com/questions/10002918/what-is-the-need-for-normalizing-a-vector

3. Training Text Embeddings

To train generative AI

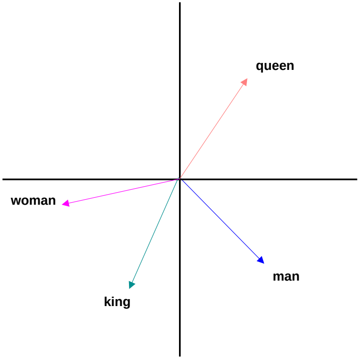
1. Create vectors
   - one for each English word/token (30-50k)
     - words can be added automatically
     - tokenizers can also be made up by letter combinations (byte-pair based, ideal for rare words and proper nouns)
   - assign each vector the same fixed length/magnitude (normalized vectors)
   - assign each vector a random direction in the 1k-12k dimensional space
2. Adjust the direction of vectors using a massive number of training documents (e.g., word2vec) by either
   - for each word, adjust its vector to be closer to the vectors of surrounding words, e.g., bag of words
   - for each word, adjust the vectors of surrounding words to be closer its vector, e.g., skip-gram

https://neptune.ai/blog/vectorization-techniques-in-nlp-guide

https://jalammar.github.io/illustrated-word2vec/

https://neptune.ai/blog/vectorization-techniques-in-nlp-guide
https://medium.com/@aidant0001/unraveling-the-magic-of-word-embeddings-1f6fac66c647

Text Embeddings with Random Directions



https://stackoverflow.blog/2023/11/09/an-intuitive-introduction-to-text-embeddings/

Training Example

For the training text:

The king is a tall man.

we move each red word closer to the blue words, or the blue words closer to the red word (depending on the training method):

1. The king is a tall man.
2. The king is a tall man.
3. The king is a tall man.
4. The king is a tall man.
5. The king is a tall man.
6. The king is a tall man.

https://towardsdatascience.com/word2vec-out-of-the-black-box-a404b4119681

Training Example

For the training text:

> *The king is a tall man. The queen is a beautiful woman. They sit together in the throne room of the castle.*

we have

- “king” getting closer to “man”
- “tall” getting closer to “man” and “king”
- “queen” getting closer to “woman”
- “beautiful” getting closer to “woman” and “queen”
- “throne” getting closer to “castle”
- “king”, “man” “queen”, “woman” getting closer to “throne” and “castle” (spans sentences)

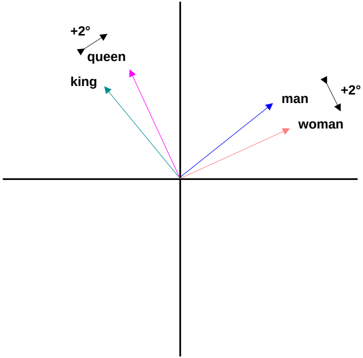
Training Example

- phrases and minor words also move closer to each other, e.g.,
  - “The” is closer to “king”
  - “throne” is closer to “room”
  - “throne” and “room” are closer to “castle”

  Consider there are 1-12 thousand dimensions, so moving words closer in one dimension might not affect closeness in other dimensions.

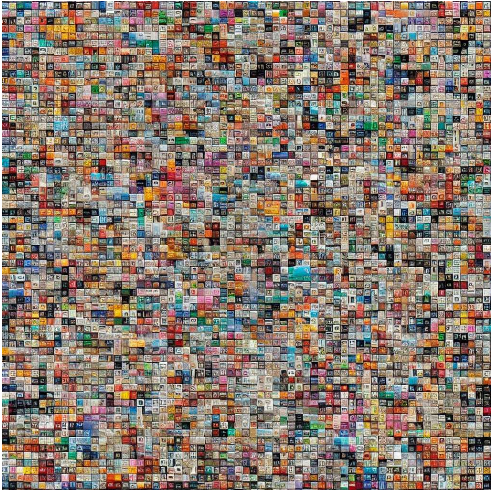
Text Embeddings with Learned Directions

King - Man + Woman = Queen
Woman - Man + King = Queen



https://jalammar.github.io/illustrated-word2vec/
https://www.elastic.co/search-labs/blog/generative-ai-transformers-explained

Text Embeddings



4. Semantic/Vector Search

Postgres has supported full text/phrase search since 2003, but that only finds base/stemmed words. It has no concept of synonyms (except those explicitly configured) or word relationships. As you can imagine from our previous slides, semantic/vector search promises much richer search capabilities. Full text/phrase search is better for precise queries and handles proper nouns better. It is possible to use a hybrid search which combines full text/phrase search and semantic/vectors search.

https://blog.meilisearch.com/full-text-search-vs-vector-search/
https://www.youtube.com/watch?v=P5VpaUyh8Iw

Semantic/Vector Search Setup

1. Download a pre-trained text embedding, or create your own
2. Choose a chunk size (sentence, paragraph, section, page, document)
3. Find text embedding vectors of all words (or byte pairs) in the chunk
4. Average the vectors
5. Store them in the database along with the chunk

https://qdrant.tech/articles/what-is-rag-in-ai/

https://zilliz.com/learn/guide-to-chunking-strategies-for-rag
https://ragaboutit.com/the-definitive-guide-to-document-chunking-for-ai-applications/

Semantic/Vector Search

1. Find text embedding vectors of all words in the query
2. Average the vectors
3. Search for the closest (nearest neighbor) vectors in the database and return their chunks

https://www.couchbase.com/blog/what-is-vector-search/

Search Setup Example

Assume you wish to index this document using semantic/vector search:

> *The king is a tall man. The queen is a beautiful woman. They sit together in the throne room of the castle.*

If you choose a chunk size of sentences, you will store and index:

1. The king is a tall man. (average 6 vectors)
2. The queen is a beautiful woman. (average 6 vectors)
3. They sit together in the throne room of the castle. (average 10 vectors)

Search Example

A search of:

Who is the king?

will find the text embedding vectors of the query words, average them, and find the closest (nearest neighbor) stored vector which is this chunk:

The king is a tall man.

Example: Install pgvector

```
CREATE EXTENSION IF NOT EXISTS vector;
```

Pgvector can be found at https://github.com/pgvector/pgvector. All queries in this presentation can be downloaded from https://momjian.us/main/writings/pgsql/trenches.sql.

Create Table to Store Documents and Vectors

```
CREATE TABLE document (
    id SERIAL PRIMARY KEY,
    content TEXT NOT NULL
);

CREATE TABLE document_embedding (
    id INTEGER PRIMARY KEY,
    -- ChatGPT4's text embedding vectors have 1536 dimensions
    embedding vector(1536) NOT NULL
);
```

Example based on the URL at the bottom right.

https://www.enterprisedb.com/blog/what-is-pgvector

Populate Document Table

```
INSERT INTO document (content) VALUES
        ('The king is a tall man.'),
        ('The queen is a beautiful woman.'),
        ('They sit together in the throne room of the castle.');

-- It is performant to create HNSW indexes on empty tables, unlike IVFFlat.
-- https://www.cybertec-postgresql.com/en/indexing-vectors-in-postgresql/
CREATE INDEX document_embedding_embedding_idx ON document_embedding
    USING hnsw (embedding vector_l2_ops);
```

Populate Text Embedding

```
#! /usr/bin/env python

""" Add document embeddings to the vector search table """

import os
import sys
from openai import OpenAI
import psycopg

if len(sys.argv) != 1:
    print("Usage: " + os.path.basename(__file__), file=sys.stderr)
    sys.exit(1)

# Get OpenAI API key
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# Specify the embedding model
MODEL_ID = "text-embedding-ada-002"
```

Populate Text Embedding

```
# Connect to the database
conn = psycopg.connect("host=localhost port=5432 dbname=ai user=postgres")

# Fetch documents
cur = conn.cursor()
cur.execute("SELECT id, content FROM document")

# Create embeddings for each document and store in the database
for doc_id, doc_content in cur.fetchall():
    embedding = (
        client.embeddings.create(input=doc_content, model=MODEL_ID).data[0].embedding
    )
    cur.execute(
        """
INSERT INTO document_embedding (id, embedding)
VALUES (%s, %s);""",
        (doc_id, embedding),
    )

# Commit and close the database connection
conn.commit()
conn.close()
```

Embeddings Stored

```
-- show embeddings for documents
SELECT content,
       substring(embedding::text, 1, 30) AS embedding
FROM document JOIN document_embedding USING (id);
                        content                        |           embedding
-------------------------------------------------------+--------------------------------
 The king is a tall man.                               | [0.004228712,-0.0132633485,0.0
 The queen is a beautiful woman.                       | [-0.006298352,0.008259722,-0.0
 They sit together in the throne room of the castle.   | [0.01191413,-0.03098976,-0.010

-- each dimension is four bytes (float4)
SELECT pg_column_size(embedding),
       pg_column_size(embedding) / 1536 AS bytes_per_dim
FROM document_embedding;
 pg_column_size | bytes_per_dim
----------------+---------------
           6148 |             4
           6148 |             4
           6148 |             4
```

Query Text Embedding

```
#! /usr/bin/env python

""" Perform vector search of documents """

import os
import sys
from openai import OpenAI
import psycopg

if len(sys.argv) != 2:
    print("Usage: " + os.path.basename(__file__) + " search_string", file=sys.stderr)
    sys.exit(1)

# Get the user query
search = sys.argv[1]

# Get OpenAI API key
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# Specify the embedding model
MODEL_ID = "text-embedding-ada-002"
```

Query Text Embedding

```
# Get embedding for the user query
embedding = client.embeddings.create(input=search, model=MODEL_ID).data[0].embedding

# Connect to the database
conn = psycopg.connect("host=localhost port=5432 dbname=ai user=postgres")

# Fetch documents in order of their vector search similarity to the user query
cur = conn.cursor()
cur.execute(
    """
SELECT content, embedding <-> %s::vector
FROM document JOIN document_embedding USING (id)
ORDER BY 2;""",
    (embedding,),
)

# Output documents ordered by similarity
for doc_content, distance in cur.fetchall():
    print(f"{doc_content:<52} {distance}")

# Commit and close the database connection
conn.commit()
conn.close()
```

Query Examples

```
-- This is matching a word, so it could have been done with full text search.
Who is the king?
The king is a tall man.                                   0.6235435682429836
The queen is a beautiful woman.                           0.6943989871300065
They sit together in the throne room of the castle.      0.7003168615080256

-- same
Who is tall?
The king is a tall man.                                   0.6273594902980766
They sit together in the throne room of the castle.      0.6957198032305553
The queen is a beautiful woman.                           0.7227548068086962
```

Query Examples

```
-- The vector for "short" is near "tall".
Who is short?
The king is a tall man.                          0.6878928016051950
The queen is a beautiful woman.                  0.7552441994053801
They sit together in the throne room of the castle.  0.7593614930258540

-- The vector for "pretty" is near "beautiful".
Who is pretty?
The queen is a beautiful woman.                  0.6579895352151753
They sit together in the throne room of the castle.  0.7473554877225373
The king is a tall man.                          0.7480856783292117

-- The vector for "palace" is near "castle".
Who is in the palace?
They sit together in the throne room of the castle.  0.5872522481044645
The king is a tall man.                          0.6646927703591086
The queen is a beautiful woman.                  0.6908769898680150

-- The vector for "chair" is near "sit".
Where is the chair?
They sit together in the throne room of the castle.  0.6865519576029434
The king is a tall man.                          0.7417508497217221
The queen is a beautiful woman.                  0.7578119887517404
```

5. Transformers

Transformers allow the embedding vectors just shown and attention blocks to generate output. While images and videos can also be created, this presentation will focus on text generation. How is this done?

1. Load the first attention block with the text embedding vectors of the words used in the user query
2. Adjust vectors to be closer to previous words/vectors
3. Repeat step 2 several times
4. Find the word closest to the last vector, and use it as the first output word
5. Repeat step 2 and later to get successive words

https://medium.com/@RobinVetsch/nlp-from-word-embedding-to-transformers-76ae124e6281
https://medium.com/@b.terryjack/deep-learning-the-transformer-9ae5e9c5a190

Attention Block Details

- Shown are self-attention blocks:
  - only previous vectors affect current vectors, not later ones
  - used to generate new words
  - translation uses cross-attention blocks where later vectors can also affect earlier ones
- While the following slides show attention block 1 in seven steps, attention blocks do vector calculations in parallel as matrix multiplication
  - GPUs are very efficient at matrix multiplication
  - internally uses a query/key/value process

https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html

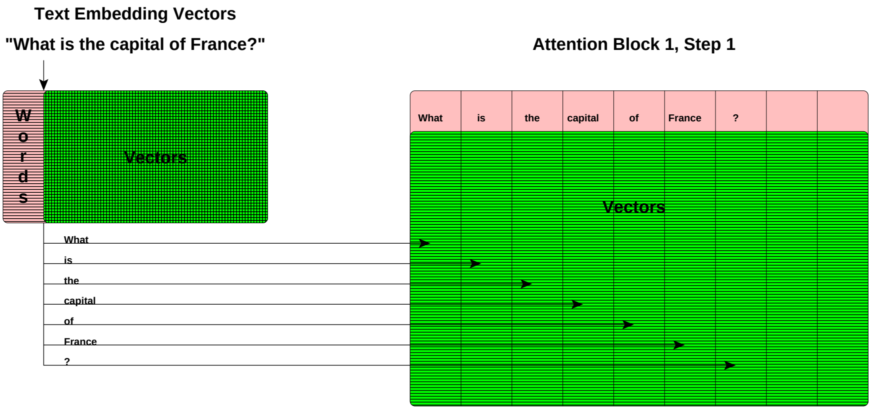
https://www.youtube.com/watch?v=4naXLhVfeho
https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms

Attention Block Details

- While text embedding vectors can be thousands of dimensions, attention blocks usually use one-tenth the number of dimensions:
  - GPT-3 uses 96 attention blocks with 128-dimension vectors
- Shown is zero-shot learning
  - some models use single or multi-shot learning, e.g., “The capital of Spain is Madrid.” is prepended to the user query
- Not shown
  - word positions are numbered in the attention vectors
  - sentence endings are also encoded; this is shown as “?”
  - used for sentence construction

https://aigrowthguys.com/zero-shot-vs-multi-shot-prompting-prompt-engineering/
https://datascience.stackexchange.com/questions/118273/specifics-about-chatgpts-architecture

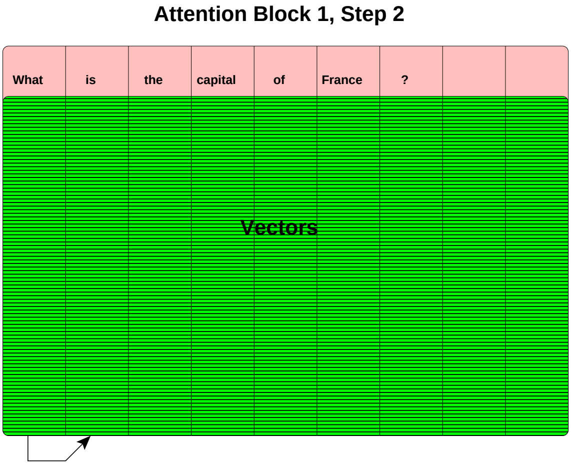
Populate Initial Attention Block



Current Attention Block Vectors

1. “What”
2. “is”
3. “the”
4. “capital”
5. “of”
6. “France”
7. “?”

Adjust the Second Vector



Adjust second vector to be closer to the first vector

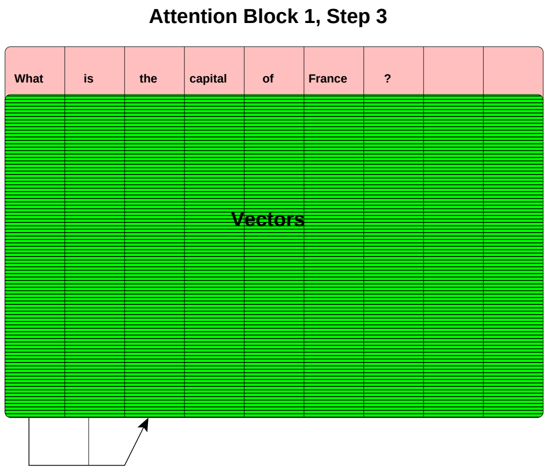
Current Attention Block Vectors

1. “What”
2. “What” “is”
3. “the”
4. “capital”
5. “of”
6. “France”
7. “?”

Matrix Mask Prevents Adjustments from Later Words/Vectors

Words	What	is	the	capital	of	France	?
What	1	0	0	0	0	0	0
is	1	1	0	0	0	0	0
the	1	1	1	0	0	0	0
capital	1	1	1	1	0	0	0
of	1	1	1	1	1	0	0
France	1	1	1	1	1	1	0
?	1	1	1	1	1	1	1

Adjust Vector 3

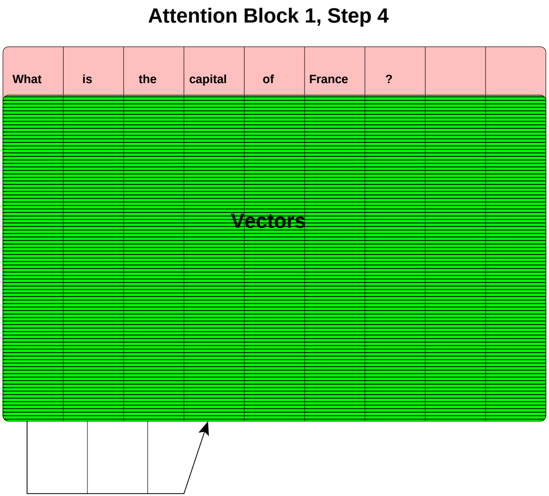


Adjust third vector to be closer the previous two.

Current Attention Block Vectors

1. "What"
2. "What" "is"
3. "What" "is" "the"
4. "capital"
5. "of"
6. "France"
7. "?"

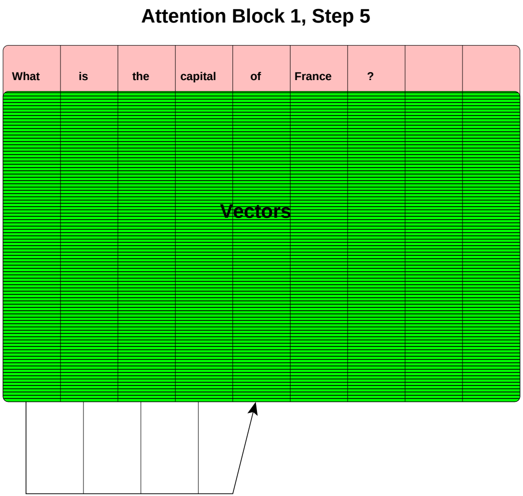
Adjust Vector 4



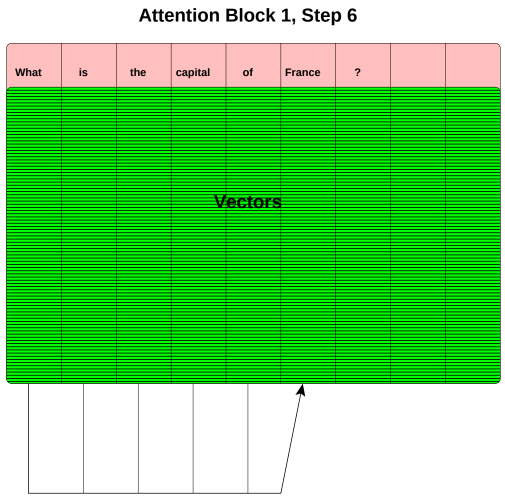
Current Attention Block Vectors

1. “What”
2. “What is”
3. “What” “is” “the”
4. “What” “is” “the” “capital”
5. “of ”
6. “France”
7. “?”

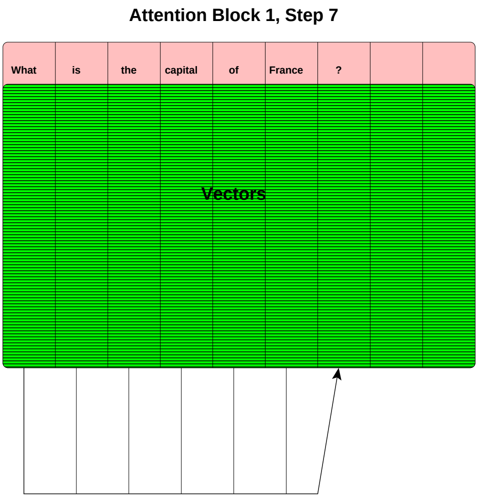
Adjust Vector 5



Adjust Vector 6



Adjust Vector 7



End of Attention Block 1 Vectors

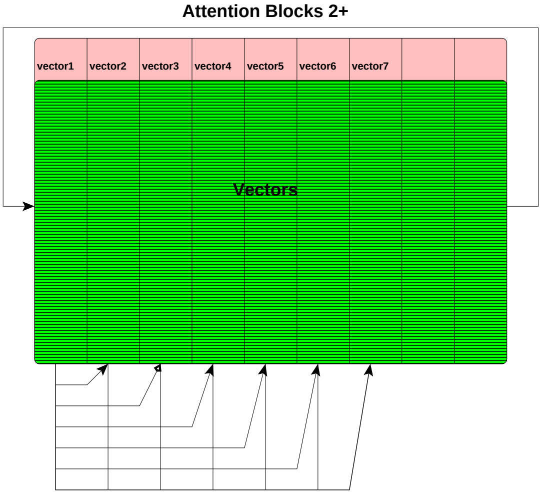
1. "What"
2. "What" "is"
3. "What" "is" "the"
4. "What" "is" "the" "capital"
5. "What" "is" "the" "capital" "of"
6. "What" "is" "the" "capital" "of" "France"
7. "What" "is" "the" "capital" "of" "France" "?"

Attention Block Details

- Each vector of an attention block potentially can be moved closer to other vectors, except the first vector:
  - this is the "attention" aspect of attention blocks
- Vectors are normalized to maintain the same magnitude
- Vector movement is not uniform
  - movement favors dimensionally-close vectors and previous movement (backpropagation)
  - distances are computed via vector dot products

https://www.datacamp.com/blog/attention-mechanism-in-llms-intuition

Continue Running Attention Blocks



The next attention block uses vectors that were adjusted by the previous attention block. The headings are now labeled as “vectors” because they no longer point to the original words of the user query.

End of Attention Block 2 Vectors

1. (“What”)
2. (“What”)×(“What” “is”)
3. (“What”)×(“What” “is”)×(“What” “is” “the”)
4. (“What”)×(“What” “is”)×(“What” “is” “the”)×(“What” “is” “the” “capital”)
5. (“What”)×(“What” “is”)×(“What” “is” “the”)×(“What” “is” “the” “capital”)×(“What” “is” “the” “capital” “of”)
6. (“What”)×(“What” “is”)×(“What” “is” “the”)×(“What” “is” “the” “capital”)×(“What” “is” “the” “capital” “of”)×(“What” “is” “the” “capital” “of” “France”)
7. (“What”)×(“What” “is”)×(“What” “is” “the”)×(“What” “is” “the” “capital”)×(“What” “is” “the” “capital” “of”)×(“What” “is” “the” “capital” “of” “France”)×(“What” “is” “the” “capital” “of” “France” “?”)

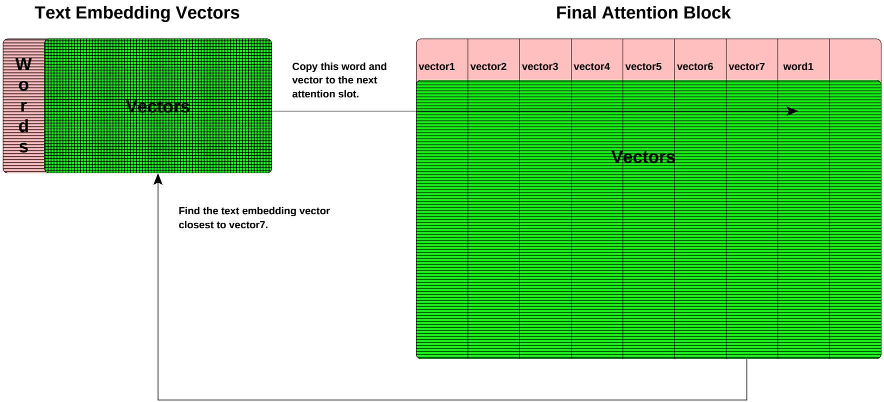
End of Attention Block 3 Vectors

1. (“What”)
2. (“What”) × ((“What”) × (“What” “is”))
3. ((“What”) × (“What” “is”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”))
4. ((“What”) × (“What” “is”) × (“What” “is” “the”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”))
5. ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”))
6. ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”) × (“What” “is” “the” “capital” “of” “France”))
7. ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”) × (“What” “is” “the” “capital” “of” “France”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”) × (“What” “is” “the” “capital” “of” “France”) × (“What” “is” “the” “capital” “of” “France” “?”))

End of Attention Block 4 Vectors

1. (“What”)
2. (“What”) × (“What”) × ((“What”) × (“What” “is”))
3. (“What”) × ((“What”) × (“What” “is”)) × ((“What”) × (“What” “is”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”))
4. ((“What”) × (“What” “is”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”))
5. ((“What”) × (“What” “is”) × (“What” “is” “the”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”))
6. ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”) × (“What” “is” “the” “capital” “of” “France”))
7. ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”) × (“What” “is” “the” “capital” “of” “France”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”) × (“What” “is” “the” “capital” “of” “France”)) × ((“What”) × (“What” “is”) × (“What” “is” “the”) × (“What” “is” “the” “capital”) × (“What” “is” “the” “capital” “of”) × (“What” “is” “the” “capital” “of” “France”) × (“What” “is” “the” “capital” “of” “France” “?”))

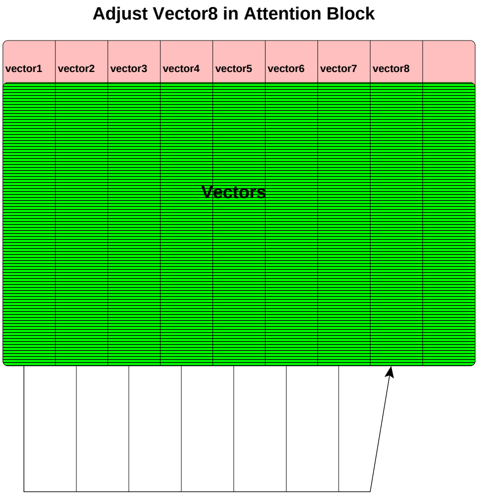
After Running All Attention Blocks: Store and Output First Word



The last vector of the final attention block has dimensional values which represent the entire user query; its position among other word vectors is used to generate answer text. AI chat apps usually display results one word at a time because they generate results one word at a time.

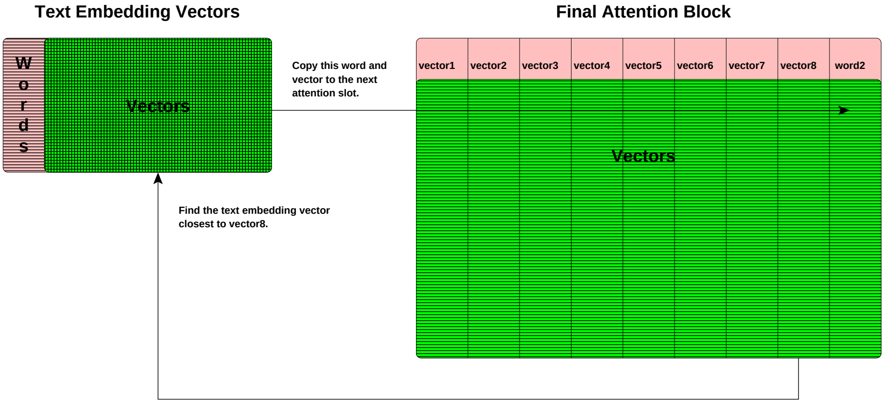
https://jalammar.github.io/illustrated-transformer/

Adjust the First Word

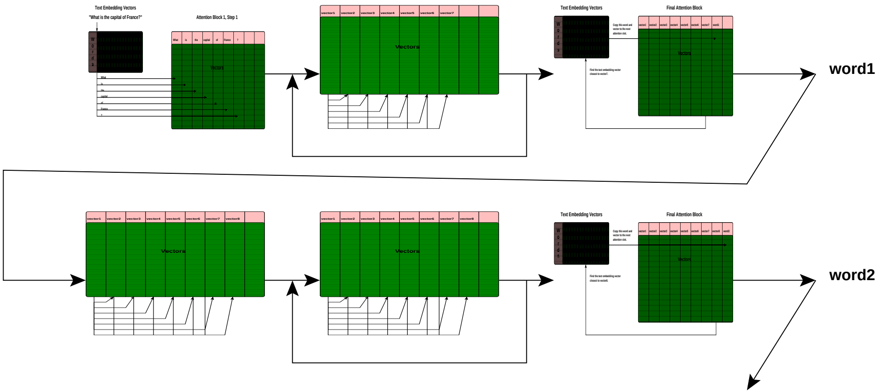


Vectors only adjust to previous vectors, so only vector8 needs adjustment.

After Running All Attention Blocks Again: Store and Output Second Word



Attention Blocks



https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html

https://www.elastic.co/search-labs/blog/generative-ai-transformers-explained
https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html

6. Generative AI

Generated using large language model (LLM) ChatGPT4 o1-mini:

What is the capital of France?
The capital of France is Paris.

The capital of France is?
The capital of France is Paris.

Where is Paris?
Paris is the capital city of France, located in the north-central part of the country along the Seine River. It is known for its rich history, art, culture, and landmarks such as the Eiffel Tower, the Louvre Museum, and Notre-Dame Cathedral.

https://dugas.ch/artificial_curiosity/GPT_architecture.html

Columbus is in Ohio. Where is Paris?
Paris is in France. There is also a city named Paris in the United States, located in Texas. If you were referring to a different Paris, please specify!

I live in Texas. Where is Paris?
In Texas, there is a city named Paris, which is located in the northeastern part of the state, in Lamar County. It is approximately 100 miles northeast of Dallas. There is also a well-known city named Paris in France, which is the capital and largest city of the country. If you're referring to the Texas city, it's known for its local attractions, including the Eiffel Tower replica with a cowboy hat. If you meant the one in France, it's famous for its art, culture, and history.

7. Retrieval-Augmented Generation (RAG)

Retrieval-augmented generation (RAG) allows for data (below in red) to be passed to the generative transformer to affect the user query (transformers can accept thousands of words/tokens).

Where is paris?
Paris is the capital city of France, located in the northern part of the country along the Seine River. It is known for its rich history, culture, art, and landmarks such as the Eiffel Tower, the Louvre Museum, and Notre-Dame Cathedral.

reply briefly
Where is Paris?
Paris is the capital city of France, located in the north-central part of the country along the Seine River.

https://www.youtube.com/watch?v=Y08Nn23o_mY
https://qdrant.tech/articles/what-is-rag-in-ai/

RAG Internals

```
$ rag -n 'reply briefly' 'Where is Paris?'
…
Payload:
--------------
{
  "model": "gpt-4o-mini",
  "stream": true,
  "temperature": 0.5,
  "max_tokens": 8192,
  "messages": [
    {
      "role": "system"
      "content": "reply briefly",
    },
    {
      "role": "user",
      "content": "Where is Paris?"
    }
  ]
}
```

Columbus is in Ohio.
Where is Paris?
Paris is the capital city of France, located in the north-central part of the country along the Seine River. It is known for its art, fashion, gastronomy, and culture, as well as landmarks such as the Eiffel Tower, the Louvre Museum, and Notre-Dame Cathedral.

I live in Texas.
Where is Paris?
Paris is the capital city of France, located in the north-central part of the country along the Seine River. It is known for its rich history, culture, art, and landmarks such as the Eiffel Tower, the Louvre Museum, and Notre-Dame Cathedral. **If you meant a different Paris, such as Paris, Texas, it is a city located in the northeastern part of the state of Texas.**

Data Can Be Passed via RAG

Using the following RAG data for each query:

{“order_id”: 735, “product”: "gears", “amount”: 21, “inv_date”: 2024-04-12}
{“order_id”: 4423, “product”: "pencils", “amount”: 13, “inv_date”: 2024-02-05}
{“order_id”: 1334, “product”: "glue", “amount”: 5, “inv_date”: 2024-03-25}

A more sophisticated example would be to first create a list of questions or keywords and their relevant SQL queries. A semantic/vector search could then be performed on each user query to choose the relevant SQL query, run it, and supply its output as RAG data.

Data Can Be Passed via RAG

When were the orders made?
The orders were made on the following dates:

1. Order ID 735: April 12, 2024
2. Order ID 4423: February 5, 2024
3. Order ID 1334: March 25, 2024

What is the earliest invoice?
The earliest invoice is for the product "pencils" with order_id 4423, dated February 5, 2024.

Data Can Be Passed via RAG

What is the total of all invoices?
To calculate the total of all invoices, we need to sum the amounts for each order:

1. Order 735: Amount = 21
2. Order 4423: Amount = 13
3. Order 1334: Amount = 5

Now, adding them together:

Total = 21 + 13 + 5 = 39

The total of all invoices is 39.

Data Can Be Passed via RAG

What is the average invoice amount?
To calculate the average invoice amount, we first need to sum the amounts from the provided orders and then divide by the number of orders.

Here are the amounts from the orders:
- Order 735: 21
- Order 4423: 13
- Order 1334: 5

Now, let's calculate the total amount:
Total amount = 21 + 13 + 5 = 39
Next, we count the number of orders, which is 3.
Now, we can calculate the average:
Average invoice amount = Total amount / Number of orders = 39 / 3 = 13
So, the average invoice amount is 13.

Complex Calculations Are Possible

Today is April 15, 2024. With 30-day payment expected, which invoices are overdue?
To determine which invoices are overdue as of April 15, 2024, we need to check the invoice dates and see if they are more than 30 days old from the current date.

1. **Invoice for gears**:
- Inv Date: April 12, 2024
- Payment Due Date: May 12, 2024 (April 12 + 30 days)
- Status: Not overdue

2. **Invoice for pencils**:
- Inv Date: February 5, 2024
- Payment Due Date: March 6, 2024 (February 5 + 30 days)
- Status: Overdue

Complex Calculations Are Possible

3. **Invoice for glue**:
- Inv Date: March 25, 2024
- Payment Due Date: April 24, 2024 (March 25 + 30 days)
- Status: Not overdue

Based on this analysis, the only overdue invoice is for **pencils** (order_id 4423).

Complex Calculations Are Possible

What is the total of invoices that are not pencils?
To calculate the total of invoices that are not for pencils, we will look at the provided orders and sum the amounts for each product except for pencils.

Here are the relevant orders:

1. Gears: 21
2. Glue: 5

Now, we add these amounts together:
21 (gears) + 5 (glue) = 26
Therefore, the total of invoices that are not for pencils is **26**.

Complex Calculations Are Possible

Today is April 15, 2024. What is the next expected order date?
Since today's date is April 15, 2024, and the most recent order date in the provided data is April 12, 2024, we can estimate the next expected order date by looking at the pattern of previous orders. However, there is not enough information in the provided data to determine a specific frequency or pattern of orders. If we assume that orders are placed regularly, we can consider the time intervals between the previous orders:

1. Order on February 5, 2024
2. Order on March 25, 2024 (48 days after the previous order)
3. Order on April 12, 2024 (18 days after the previous order)

Given the decreasing intervals, it's challenging to predict a specific date. However, if we were to assume a similar timeframe, we could expect the next order to occur sometime in the following weeks. A rough estimate might suggest the next order could be expected around late April to early May 2024, but without a clear pattern, it remains uncertain.

SQL Queries Can Be Generated

What SQL query will generate the latest invoice date?
To generate the latest invoice date from a table (let's assume the table is named 'invoices'), you can use the following SQL query:

```sql
SELECT MAX(inv_date) AS latest_invoice_date
FROM invoices;
```

This query uses the 'MAX()' function to find the most recent date in the 'inv_date' column, returning it as 'latest_invoice_date'. Make sure to replace 'invoices' with the actual name of your table if it's different.

Data Can Be Taken from a Database

```
CREATE TABLE invoice (
        order_id INTEGER PRIMARY KEY,
        product TEXT NOT NULL,
        amount NUMERIC(10,2),
        inv_date DATE
);

INSERT INTO invoice VALUES
        (735, 'gears', 21, '2024-04-12'),
        (4423, 'pencils', 13, '2024-02-05'),
        (1334, 'glue', 5, '2024-03-25');

$ # This invoice list can be customized by customer.
$ rag "$(psql --tuples-only -c '\
         SELECT to_json(invoice.*) \
         FROM invoice \
         WHERE cust_id = 12;' ai)" \
> 'How many invoices are there?'
There are three invoices.
```

8. Deployment Options

As you have seen, there are several options for using generative AI:

- Cloud service model, e.g., ChatGPT
- Self-managed model
  - only publicly trained, e.g., Meta's Llama
  - publicly trained with private fine-tuning training, e.g., domain-specific chat assistant
  - locally trained public and private data sets, e.g., structured-wikipedia
  - locally trained with only private data
  - private training requires an open data model
    - Hugging Face offers many pretrained models and public data sets
    - most OpenAI models are not open*
- The above options can be augmented with local data, i.e., RAG; data derived from
  - personal preferences
  - relational data and supplied as JSON, e.g., PostgreSQL
  - text retrieved either via full text/phrase search or semantic/vector search

https://zapier.com/blog/hugging-face/

https://www.infoworld.com/article/2338922/5-easy-ways-to-run-an-llm-locally.html

https://www.nature.com/articles/d41586-024-02998-y

* https://analyticsindiamag.com/ai-mysteries/6-open-source-models-from-openai/

Relational Databases for Semantic/Vector Search & Generative AI

AI Feature	Details	DB Appropriateness
semantic/vector search	search existing database contents	good
text embedding vector training	batched changes, mostly static	poor
transformers for user queries	billions of comparisons	poor
retrieval-augmented generation (RAG)	add details to user queries	good
language queries on retrieved data	supplied as JSON	good
data analysis	regression and time series	good
generate SQL queries	natural language to SQL	unknown*

Relational databases continue to be appropriate for discriminative AI. AI tools can also help with database migrations, e.g., Oracle to Postgres.

https://momjian.us/main/blogs/pgblog/2018.html#November_28_2018
* https://www.reddit.com/r/SQL/comments/127zawr/who_here_is_using_chatgpt_to_help_with_sql_code/

Postgres AI Solutions

- pgvector, already covered
- Tembo's pg_vectorize
- EDB's AI extension *aidb* (Pipelines)
- Timescale's AI extension *pgai*
- PostgresML's AI toolkit
- AI toolkits from cloud vendors
- Lantern Cloud
- Workik, PopSQL, and SQL AI SQL query generators
- Postgres.AI's PostgreSQL chatbot
- DBtune for server parameter tuning

https://github.com/ftisiot/postgresql-ai-projects

9. Conclusion

- Pre-computer philosophy
- 1950's Turing test
- 1980's Expert systems
- 1970's - 1990's AI winter
- 2010's Robotics
- 2013 word2vec by Google
- 2017 Attention blocks by Google
- 2022 ChatGPT for generating text, DALL-E for generating images

https://www.youtube.com/watch?v=uocYQH0cWTs

How Did Google Miss the Boat?

- Groundbreaking research by Google in the 2010's to support language translation
- Products focused on revenue-generating activities like web search and advertising
- Worked on AI also to support device control
- Did not focus on aggregating knowledge across web pages like ChatGPT, or did not wish to risk existing revenue streams

https://www.reddit.com/r/MLQuestions/comments/18wc52b/why_isnt_google_ahead_of_the_competition_when_it/

The Future: Revolutionary Vision

A year ago, if you had said to me in our lifetime will we have capabilities like we have today now with ChatGPT4 … if you explain the kinds of things that ChatGPT4 can do I probably would have said to you a year ago I don't know if we will have those capabilities in our lifetime — and now we have it today — so the speed at which this is moving is staggering. — Jon Krohn, June 12, 2023

https://www.youtube.com/watch?v=Ku9PM26Cc2c&t=2h12m57s
https://neurosciencenews.com/ai-existential-threat-27543/

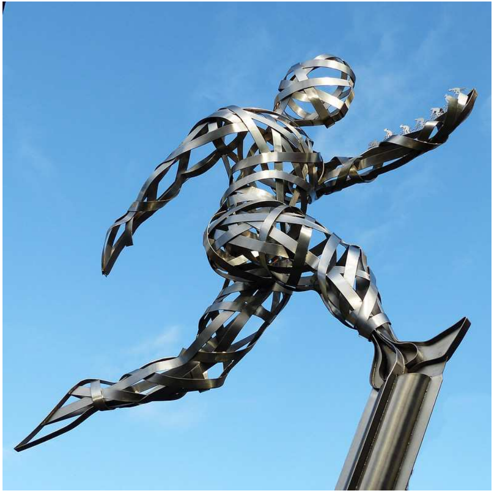
The Future: Incremental Vision

We are used to the idea that people or entities that can express themselves, or manipulate language, are smart — but that's not true. You can manipulate language and not be smart, and that's basically what LLMs (large language models) are demonstrating. — Yann LeCun, *October 11, 2024*

https://www.linkedin.com/posts/yann-lecun_an-article-in-the-wall-street-journal-in-activity-7250915579228827648-WWA2/

https://www.msn.com/en-in/money/news/why-were-unlikely-to-get-artificial-general-intelligence-anytime-soon/ar-AA1EWy4y

Conclusion



https://www.flickr.com/photos/davep-uk/

https://momjian.us/presentations