Vector Databases and Embeddings Demystified

Jackie Gleason

Who am I

- Lead AI Developer NetJets
  - 20+ yrs experience
- iOT and robotics hacker
- THEJackieGleason.com
- 3D Printing enthusiast
- Short North Resident
- Traveler
- 2nd Codemash as a Speaker
  - Grails and AngularJS, bridging the gap (2015)
  - More like Spring Boot and Vite these days.

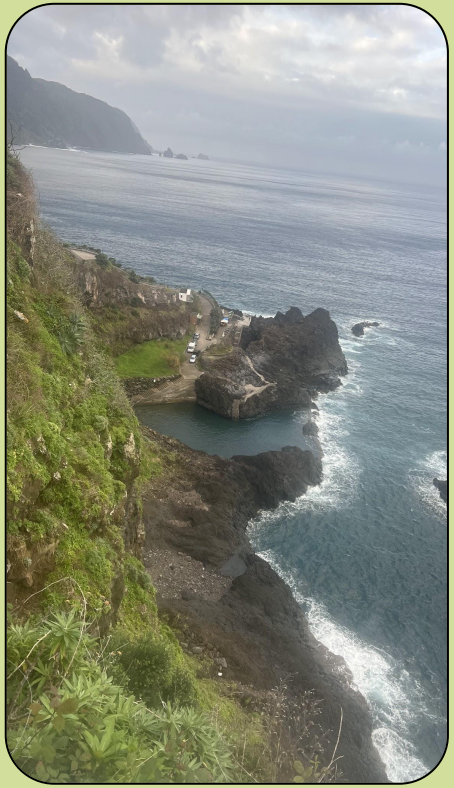
Shout out Claude

What we will learn

- What are Vectors and Embeddings
- How do we use Embeddings
- What Problems do Embeddings Solve
- What comes next?
- Summary

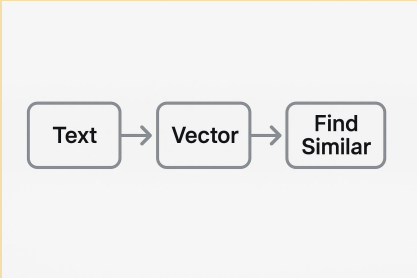
The Mystery

- How does Netflix know you'll love that obscure documentary?
- How does Google Photos find every picture of your dog?
- How does ChatGPT answer questions about your documents?



The Secret Sauce

Vector Embeddings + Semantic similarity



1. Transform data into numerical representations
2. Measure distance between vectors
3. Shift from 'exact match' to 'semantic understanding'

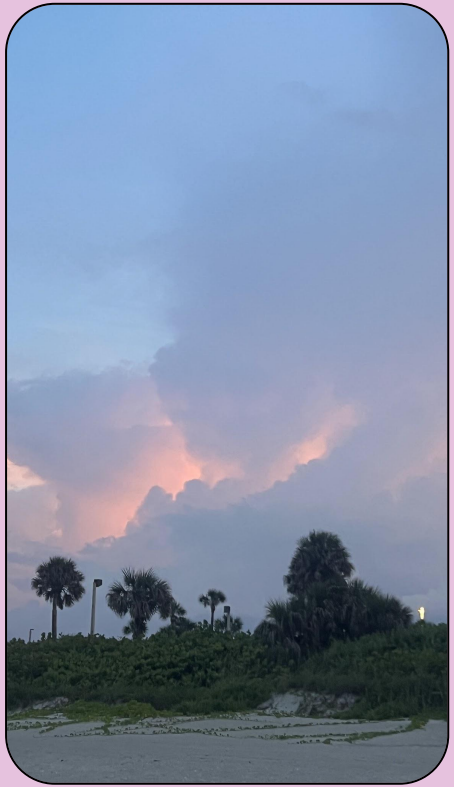
Semantic similarity

How similar things are

... is a metric defined over a set of documents or terms, where the idea of distance between items is based on the likeness of their meaning or semantic content[citation needed] as opposed to lexicographical similarity.

What is a Vector?

- A vector is just an ordered list (typically numbers) that represents something measurable.
- Physical world examples:
  - GPS coordinates → [latitude, longitude]
  - RGB color values → [red, green, blue]
- In AI:
  - We can represent concepts like words or images as numeric lists — e.g., 'dog' → [0.8, 0.3, 0.1]
- Key idea:
  - Turning things into numbers lets us use math to compare and find relationships — similar items have similar vectors.



What is an Embedding?

What Is an Embedding?

- A **vector** is just a list of numbers.
- An **embedding** is a vector trained to capture meaning or relationships between things.

Examples:

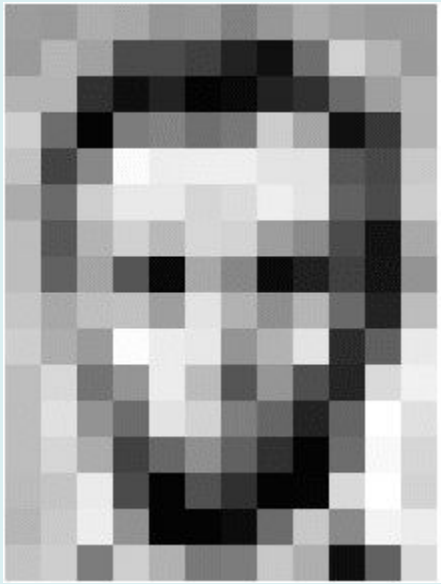
- A random number on a book is just a number (vector).
  - A Dewey Decimal number encodes what the book is about, this is a type of embedding
- A random coordinate is just a point on a map (vector).
  - A coordinate that maps to a real location with meaning (e.g., a city) is an embedding.

Embeddings give numbers meaning — they map things into a space where distance reflects similarity.

Example

Image

Limitations?



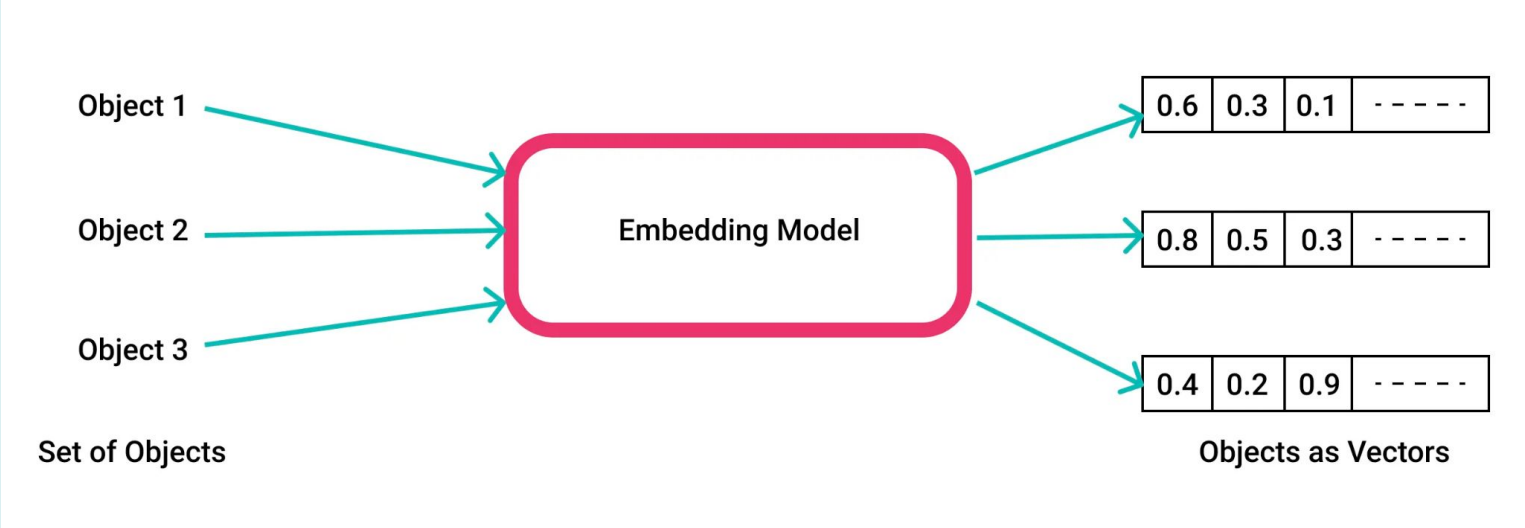
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

https://ai.stanford.edu/~syyeung/

Diagram

How Embeddings Work



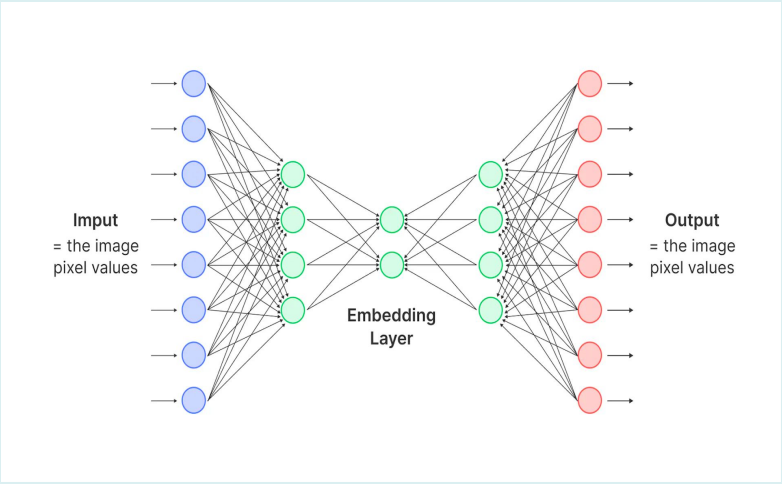
https://www.pinecone.io/learn/vector-embeddings/

Terms

Neural Network

What is it?

- Text is broken into chunks (**tokens**) and passed through layers of **transformer blocks**.
- Each layer adjusts how strongly tokens “pay attention” to others, guided by **learned weights**.
- As the text moves through layers, it gathers **context**—words influence each other based on meaning and position.
- By the final layer, the model has built a rich understanding of the input to predict the next word or generate a response.
- The network is **trained** by adjusting the weights to improve accuracy over time.



Types of Neural Networks

Type	Best For	How It Creates Embeddings	Examples
CNN	Images, spatial data	Learns local patterns (edges → shapes → objects)	MobileCLIP, ResNet
Transformer	Text, sequences	Attention across all tokens simultaneously	BERT, GPT, CLIP
Hybrid	Mobile/edge optimization	CNN efficiency + Transformer reasoning	MobileCLIP

Continued

- The text is sent to the LLM.
- It is split into **tokens**, and an **embedding** is created for each one.
- These embeddings pass through the **neural network**, where **weights** help the model learn relationships and context.
- The model uses that understanding to **predict the next word or generate a response**.

Dimensions vs Parameters

STUDENT 2

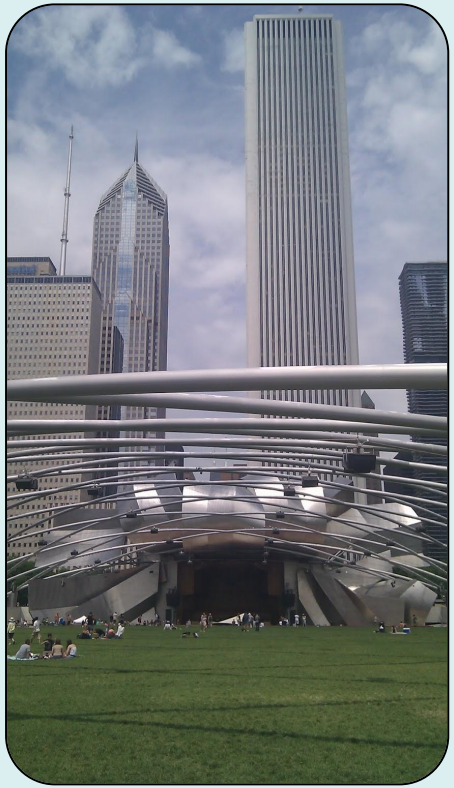
- **Dimensions**:
  - The size or shape of vectors and tensors
  - Determines how many numbers are in an embedding or layer output
- **Parameters**
  - The **trainable values** learned during training
    - Weights
    - Biases
    - Layer normalization scales
    - Attention matrices

✔ Dimensions:
- Input dimension = 1024
- Output dimension = 4096

Weight matrix dimensions = 1024×4096

✔ Parameters:
- Number of parameters = 1024×4096 weights + 4096 biases

Qwen3:0.6B Shows that there are 600 million parameters



Deep Dive

Topology V. Ontology

When you train embeddings:

- The **network weights** create a *topology* — a “shape” in high-dimensional space where meaning is organized.
- The **ontology** is the *human-level interpretation* of that structure — what those clusters *mean* (e.g., “these are all animals,” “these are all financial terms”).

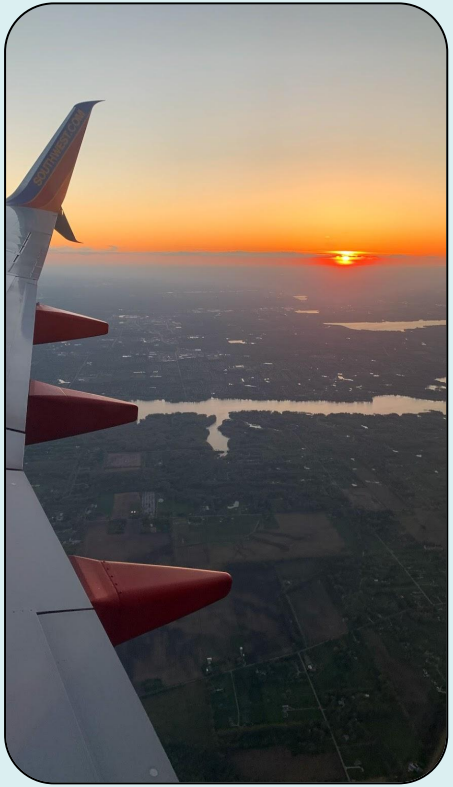
So you can think of it like this:

Topology = how embeddings are positioned.
Ontology = what those positions represent.

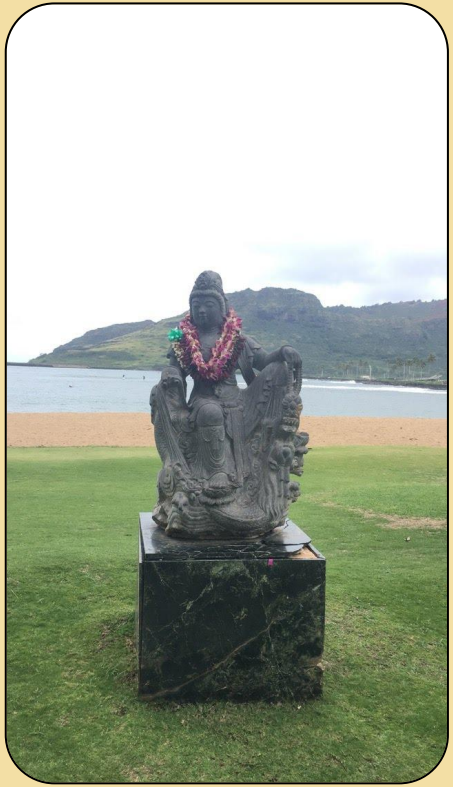
They’re related — but **topology is what the model learns**, while **ontology is what we, humans, infer** from it.

Encoding

- ***Definition:*** Converting raw input (text, audio, image, etc.) into a structured numerical representation (vector/tensor)
- ***Purpose:*** Make the input understandable to neural networks
- Encoded inputs from different modalities can be projected into a shared **embedding space**
- Common encoding types:
  - Tokenization + embedding (text)
  - Spectrograms or MFCCs (audio)
  - Pixel or patch embeddings (images)



Tokenization



- Tokenization breaks text into smaller units (“tokens”) that an LLM can process
- Tokens can be words, sub-words, characters, or symbols (e.g., “automation” → “auto”, “mation”)
- Each token is mapped to a numeric ID from the model’s vocabulary
- These token IDs are converted into vectors (embeddings) that capture semantic meaning
- Similar tokens and phrases produce similar embeddings, enabling semantic search and clustering

Tokenization impacts

Embedding quality (how meaning is preserved)

Context length (token limits)

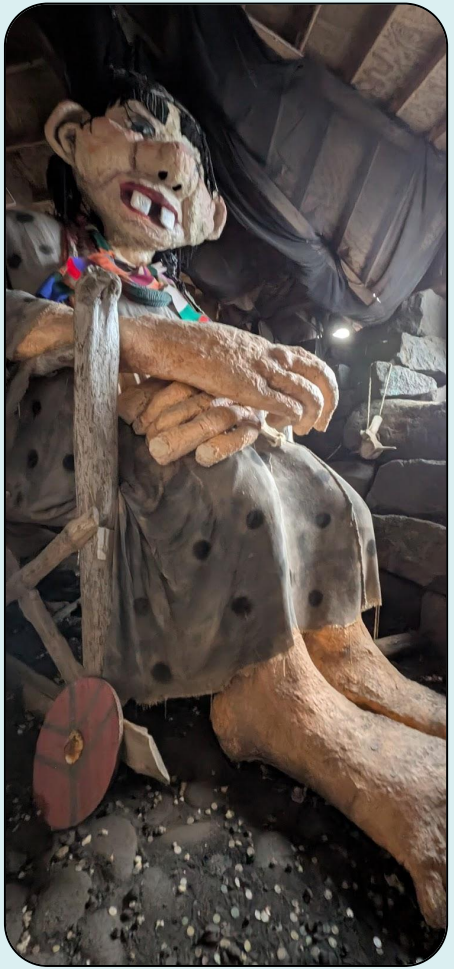
Cost and performance (more tokens = more compute)

Where things get mathy

Semantic Similarity

Deep Dive

- ***Semantic similarity*** measures how much two pieces of text (words, phrases, sentences) share meaning rather than just having the same words.
- **Metrics:** To quantify “closeness”, we use measures like
  - **Euclidean distance**: Distance
  - **Dot product:** Scaled alignment
  - **Cosine similarity:** Angle
- **LLM context**: The model’s weights adjust how embeddings are formed and transformed. Using similarity between embeddings, the system can choose which tokens, concepts or contexts are most relevant (i.e., “similar”) and thereby influence prediction or retrieval..



Deep Dive

Measuring Similarity

Three Main Similarity Metrics:

1. Cosine Similarity - *Measures angle between vectors*
- Range: -1 to 1 (1 = identical)
- Best for: Text embeddings (default choice)

2. Euclidean Distance - *Straight-line distance*
- Range: 0 to ∞ (0 = identical)
- Best for: When magnitude matters

3. Dot Product - *Angle + magnitude*
- Range: -∞ to ∞ (higher = more similar)
- Best for: Speed-critical, normalized vectors

Pro tip: OpenAI embeddings are pre-normalized

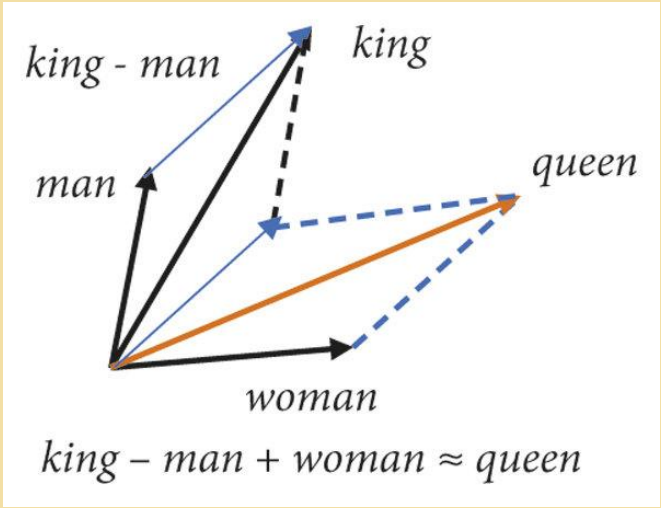
High-Dimensional Space

- 1D: One concept (e.g., sentiment: sad ←→ happy)
- 2D: Two concepts (e.g., animal type + sentiment)
- 3D: Three concepts (what we can visualize)
- 1,536D: Capturing rich nuance of language

Example: 'bank' – financial? river edge? verb? formal? historical?

Example: King - Man + Woman = Queen

How do we treat language like vector match?



More Info

Section 1

Cosine Similarity

Meaning = Direction, Not Distance

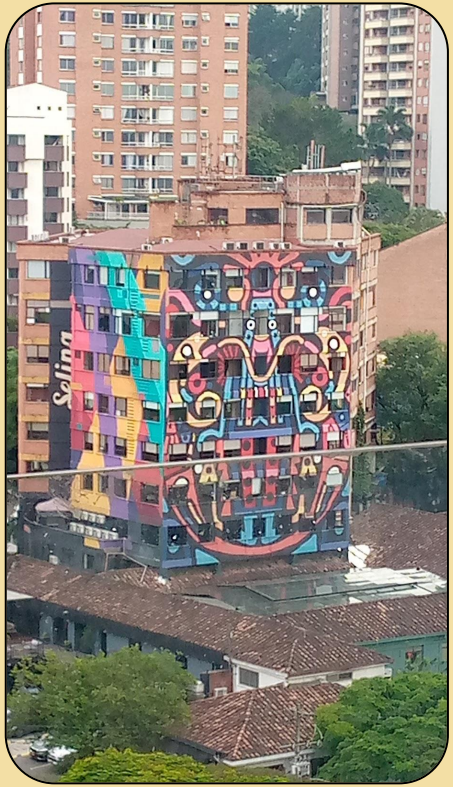
- To compare meanings, we measure the angle between two vectors, not their length.
- Cosine similarity = how aligned the directions are.

$$\text{cosine}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

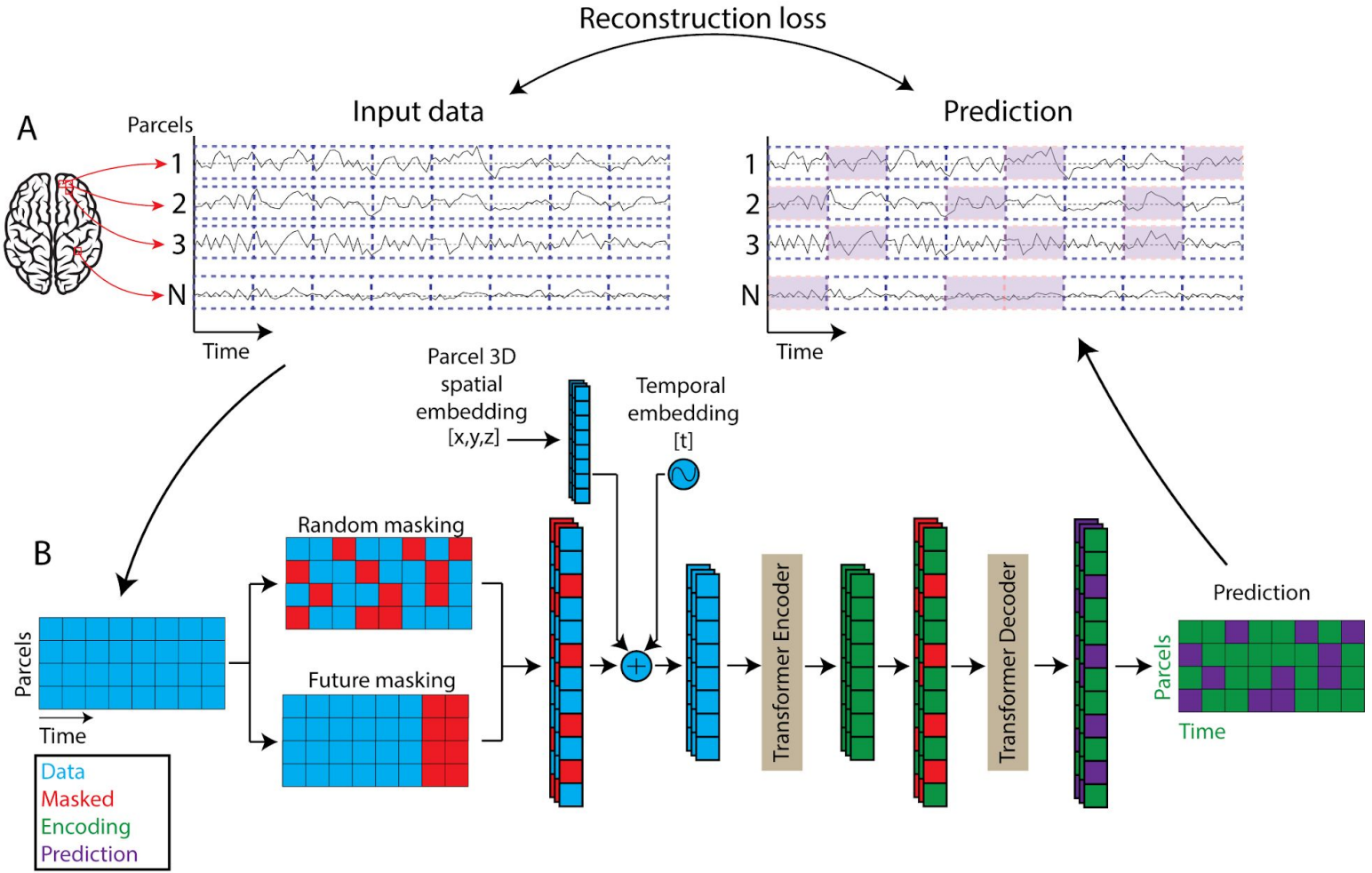
- Why angle matters:
  - If two vectors point in the same direction → meanings are similar
  - Magnitude (length of the vector) doesn't matter
  - Direction = the semantic pattern across dimensions
- Real example:
  - "King" and "Queen" vectors point in similar directions
  - "King – Man + Woman" produces a vector pointing toward "Queen"
  - This works because cosine similarity captures relational direction, not raw distance

Best Practice

MultiModal Embeddings



- Most common embeddings are using text to predict text.
- Non-text based embeddings are starting to take off
- Examples
  - Ultravox: Speech directly to LLM without transcription
  - Wav2Vec (Meta): Turns soundwaves into embeddings
  - ImageBind (Meta): Works on images, videos, thermal, etc
  - BrainLM: BrainWaves to embeddings


Reconstruction loss
A
Parcels
Input data
Prediction
1
2
3
N
Time
Parcel 3D spatial embedding [x,y,z]
Temporal embedding [t]
B
Random masking
Future masking
Parcels
Time
Data
Masked
Encoding
Prediction
Transformer Encoder
Transformer Decoder
Prediction
Parcels
Time

Another Great Example

MultiModal

Clip

- Open source model maintained by OpenAI
- Creates embeddings for both text and images
- Allows to search for images based on text and generate text based on images.
- Both vectors live in the same high dimensional space.
- "A golden retriever on a beach" will have high similarity to images of dogs on beaches

MobileClip

- Apple’s version great for low resource devices and apple silicon

OCR v. Clip?

Visualization

- Embeddings live in **hundreds or thousands of dimensions**, far beyond human perception.
- To visualize them, we **reduce dimensionality** down to 2D or 3D.
- **t-SNE** (t-Distributed Stochastic Neighbor Embedding):
- Captures **local structure** (nearby points stay nearby)
- Preserves **clusters of meaning**
- Repositions points so humans see the relationships clearly

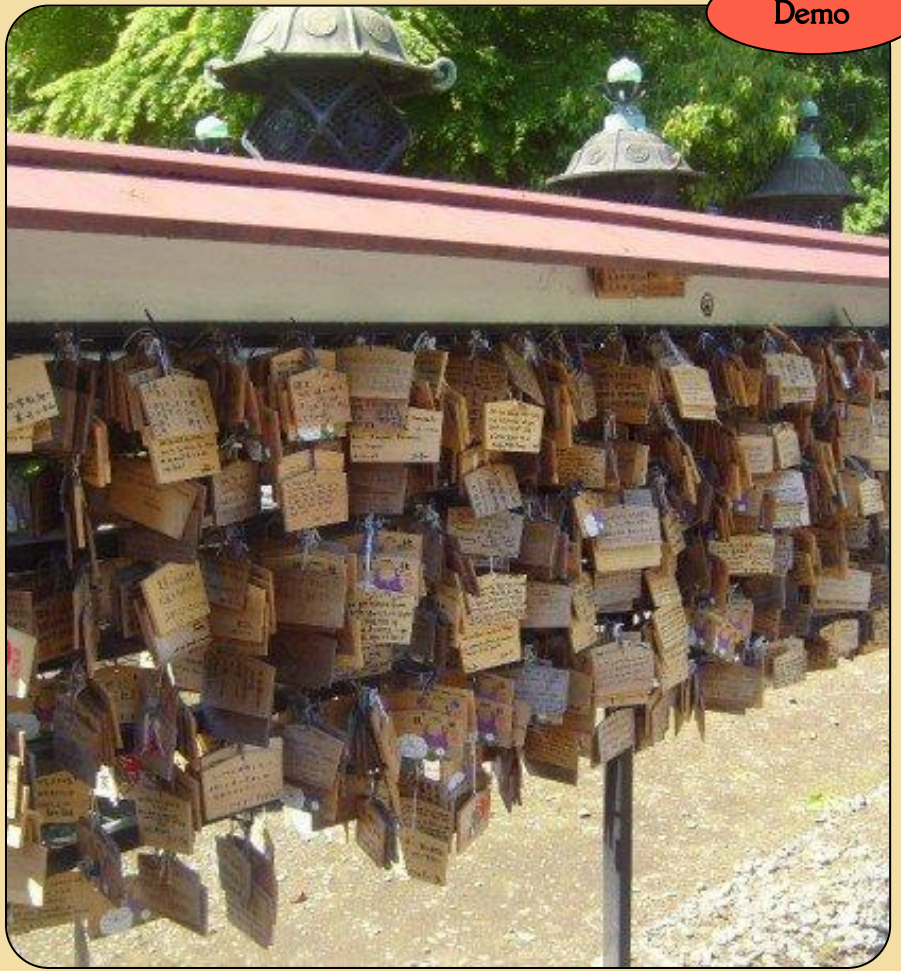
Process overview:

1. Reduce dimensions to 2
2. Plot each vector as a point on an **XY graph**
3. t-SNE (or UMAP) maintains **relative proximity**, so similar words cluster

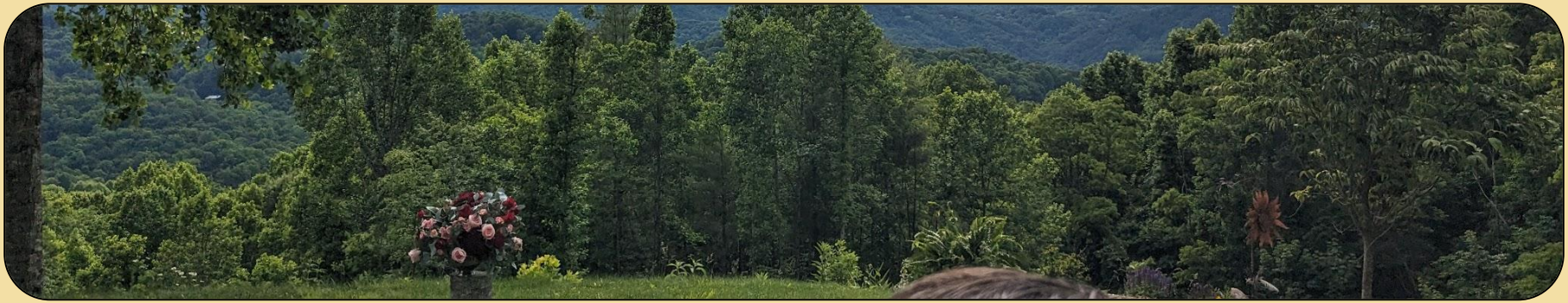
Angle still matters after projection:

- In high dimensions, similarity = small angle
- Dimensionality reduction preserves those local angles
- Clusters in 2D correspond to highly aligned vectors in the original space

Demo



DEMO



From Concepts to Models

- Embeddings are learned — not hand-coded.
- Different models specialize in creating embeddings for specific goals.
- Some focus on **words**, others on **sentences**, **documents**, or even **images**.
- The choice of model affects **accuracy**, **speed**, and **context understanding**.

What do we do?!

Popular Embedding Models

- Most people just default to using the **Open Ai Ada 002** Embedding model.
- Embedding models are typically rated using **MTEB** or **Massive Text Embedding Benchmark**
- Available on **HuggingFace**
- **Qwen3** (Ali Baba) has been making waves (Did you see my IoT presentation?)

Section 1

Review

Review Time!

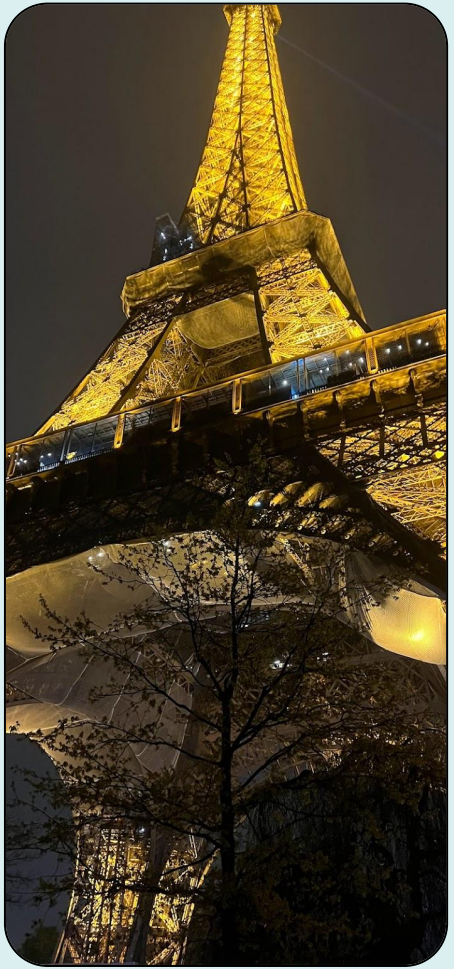
- What is an embedding
  - Dimensions
  - Shape
  - Similarity
- How do we use embedding
  - Tokenize
  - Semantic Similarity
  - Embedding models
- MultiModal
  - Embeddings to/from non-text data
  - Layering

How do we use embeddings?

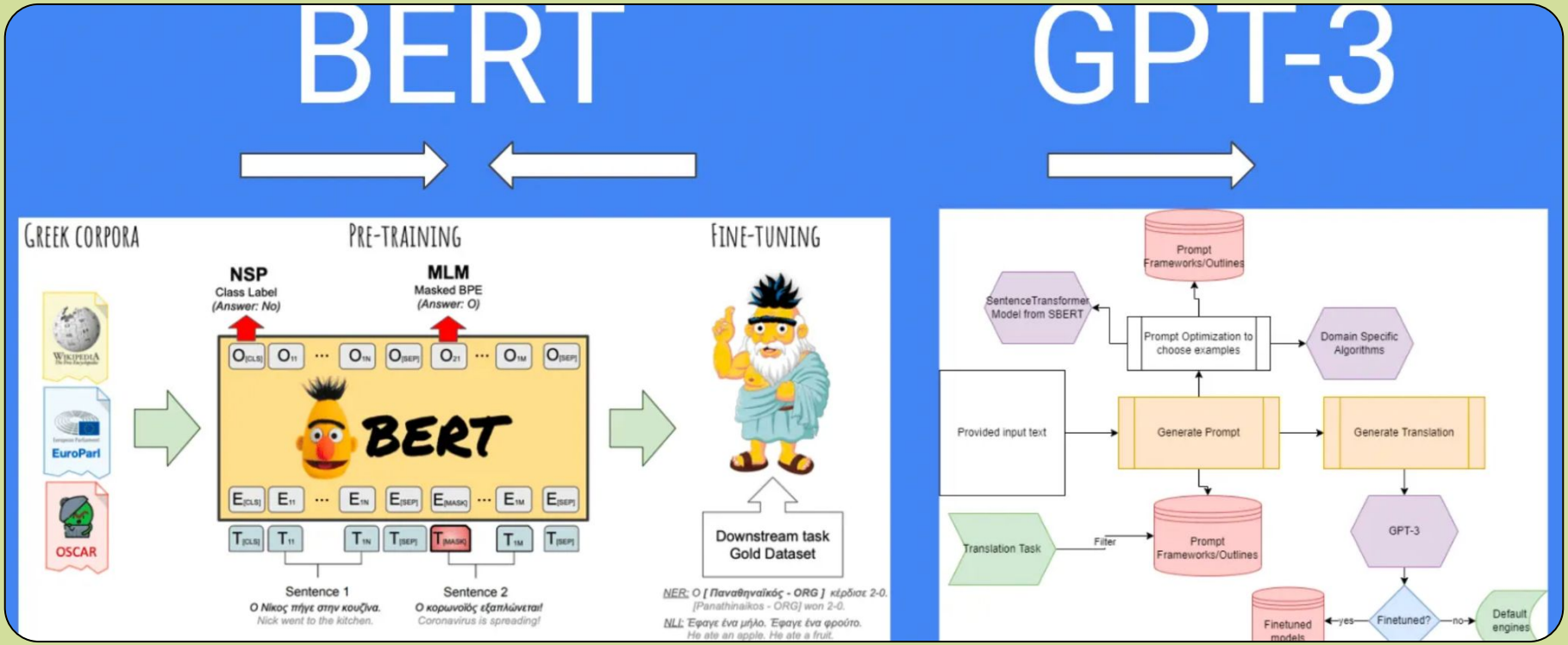
BERT

Cir. 2018

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- Developed by Google for NLP
- BERT embeddings encode contextual meaning, and fine-tuned BERT models can learn domain-specific knowledge.
    - Can be fine-tuned to learn domain-specific language patterns
- Uses bidirectional self-attention, allowing each token to consider both left and right context when forming its embedding.



VS GPT

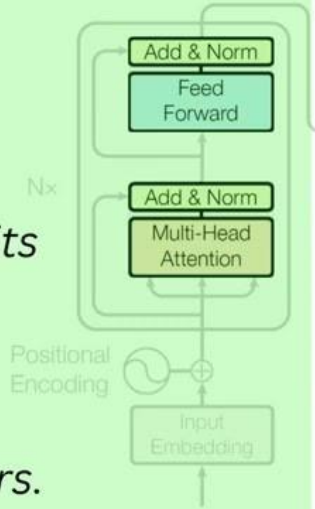

BERT
GPT-3
GREEK CORPORA
PRE-TRAINING
FINE-TUNING
NSP
Class Label
(Answer: No)
MLM
Masked BPE
(Answer: O)
BERT
Sentence 1
Ο Νίκος πήγε στην κουζίνα.
Nick went to the kitchen.
Sentence 2
Ο κορωνοϊός εξαπλώνεται!
Coronavirus is spreading!
Downstream task
Gold Dataset
NER: Ο [Παναθηναϊκός - ORG] κέρδισε 2-0.
[Panathinaikos - ORG] won 2-0.
NLI: Έφαγε ένα μήλο. Έφαγε ένα φρούτο.
He ate an apple. He ate a fruit.
Prompt Frameworks/Outlines
SentenceTransformer Model from SBERT
Prompt Optimization to choose examples
Domain Specific Algorithms
Provided input text
Generate Prompt
Generate Translation
Translation Task
Filter
Prompt Frameworks/Outlines
GPT-3
Finetuned models
yes
Finetuned?
no
Default engines

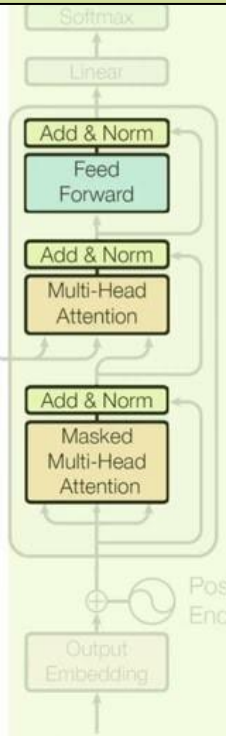
VS GPT

BERT

Google

use transfer learning to ***continue learning*** *from its existing data when adding user-specific tasks and layers.*





GPT

OpenAI

decodes from its massive pre-learned embeddings to present output that matches user prompts. It

Cir. 2020

RAG

- Retrieval-Augmented Generation
  - Stores the original material along-side the generated embedding
  - The embedding can then be used for similarity search.
  - The similarity search can then be used to retrieve the original material to be included in the request being sent to the LLM.
- Compliments the LLM
  - Domain specific knowledge
  - Up-to-date knowledge
  - Provide facts

Part 2

Vector Stores

Using Embeddings

- Stores and searches embeddings
- Deterministic
- Can increase performance with minimal accuracy hit using Approximate Nearest Neighbor (ANN)
- Vector space can be multi-modal (Text/Img/Video)
- Example
  - 1 billion vectors × 1536 dimensions
  - Naive approach: **25 minutes** per query

Approximate Nearest Neighbor

- ANN (Problem Category)
  - Graph-based algorithms
    - **HNSW**
    - NSW (Navigable Small World)
    - ONNG
  - Tree-based algorithms
    - KD-Tree
    - Ball Tree
  - Hash-based algorithms
    - LSH (Locality Sensitive Hashing)
    - **Product Quantization**
  - Clustering-based algorithms
    - **IVF (Inverted File Index)**

Tradeoffs

- Time (query latency)
- Memory (RAM to hold the index structure)
- Build time (how long to construct the index initially)
- Accuracy (how often you find the true nearest neighbors)

Constraint	Best Family	Why
Fastest query time	Graph-based (HNSW)	Efficient navigation through connected nodes; scales well to high dimensions
Lowest memory	Hash-based (LSH, Product Quantization)	PQ especially — compresses vectors themselves; no extra structure to store
Fastest build time	Tree-based or Hash-based	Simpler index construction; trees just recursively partition, hashing is straightforward
Highest accuracy	Graph-based (HNSW)	Multiple entry points and layered navigation find true neighbors more reliably

STUDENT 1

HSNW (Hierarchical Navigable Small Worlds)

- Graph-based algorithm for fast similarity search in high-dimensional vector spaces
- Builds multiple layers of “small-world” graphs — sparse on top, dense at the bottom
- High-recall, low-latency, in-memory vector search
- Supports real-time updates

Graph-Based RAG (Neo4j)

- Stores embeddings as node properties
- Maintains rich relationships between nodes
- Supports complex queries combining relationships + similarity
- Retrieval example: “Find documents similar to this chunk AND related to a given topic or user”
- Good for knowledge graphs, provenance, recommendations

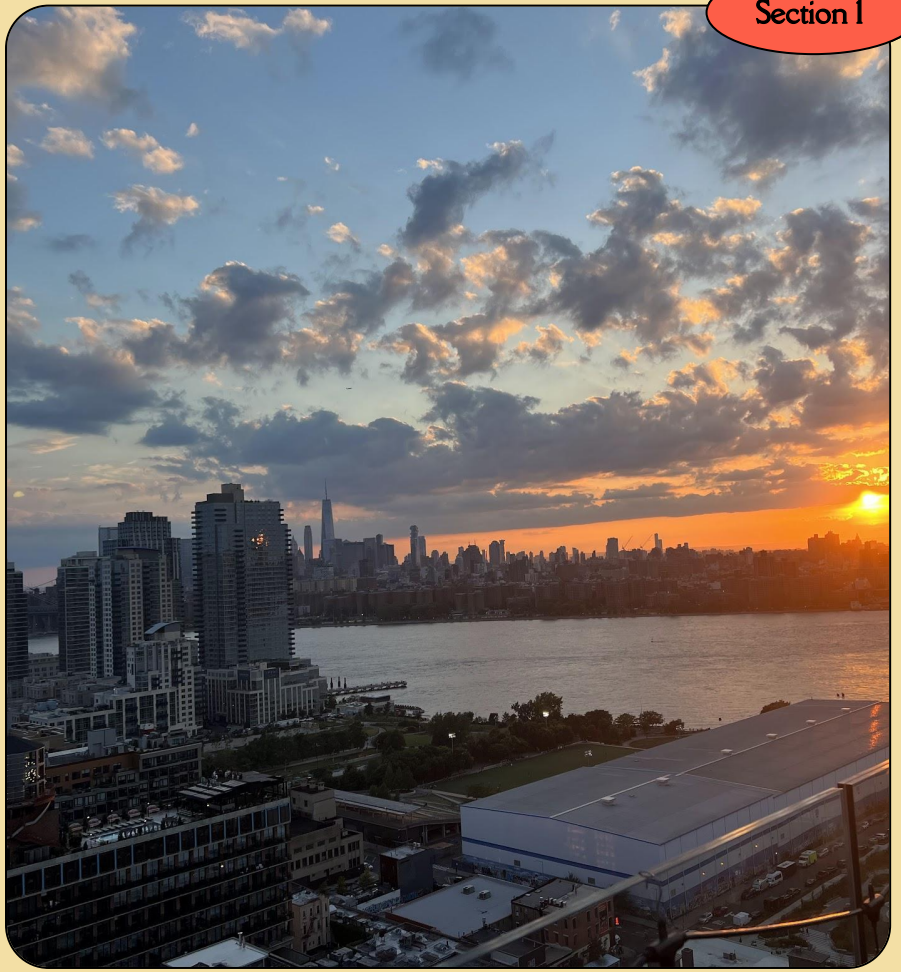
Vs SQL, Etc

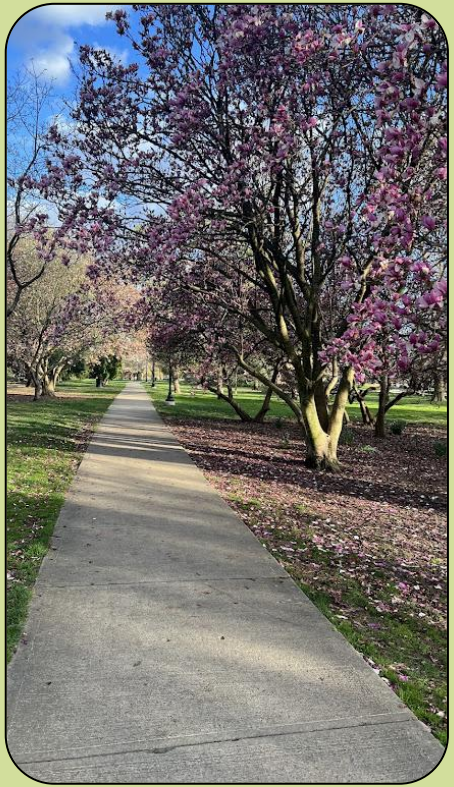
SQL: Use complex logic to get exactly what you want

NoSQL: Retrieve a record by ID

Graph: Retrieve information related to a record

Vector: Retrieve records similar to a query embedding





ANN vs Graph

	Vector Search (ANN/HNSW)	Graph Queries (Cypher)
Best for	"Find things *similar* to X"	"Find things *connected* to X"
How it works	Embedding distance (cosine similarity)	Traversing explicit relationships
Strengths	Semantic matching, unstructured text	Aggregation, multi-hop paths, filtering
Struggles with	"How many?" / "Which ones depend on?"	"What does this document mean?"
Example	"Find tasks about billing updates"	"Which services depend on Database?"

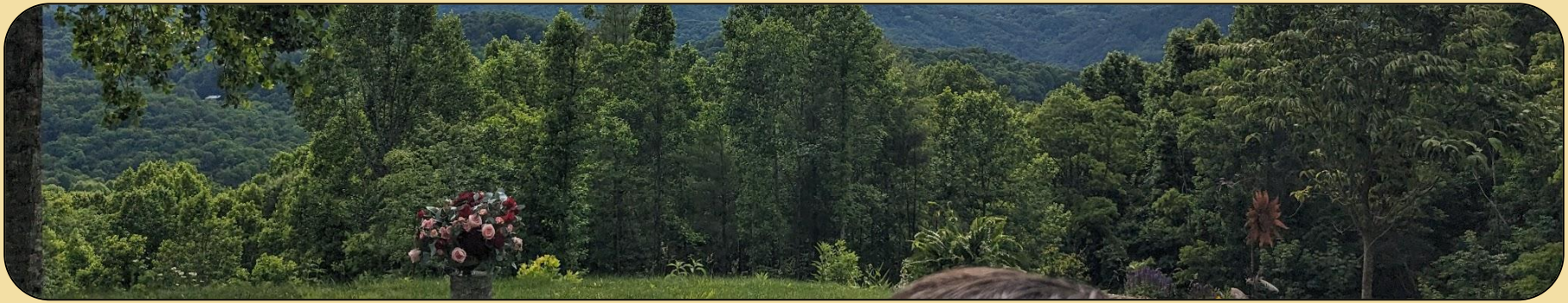
Vector Storage Options

- ***Vector-native databases:*** Pinecone, Milvus (fast ANN indexing)
- ***In-memory stores***: LangChainJS, Redis (low-latency, ephemeral)
- ***Document DBs:*** MongoDB Atlas Vector Search (partitioned, structured)
- ***Graph DBs:*** Neo4j (embedding stored on nodes for relational queries)

Vector Storage Best Practices

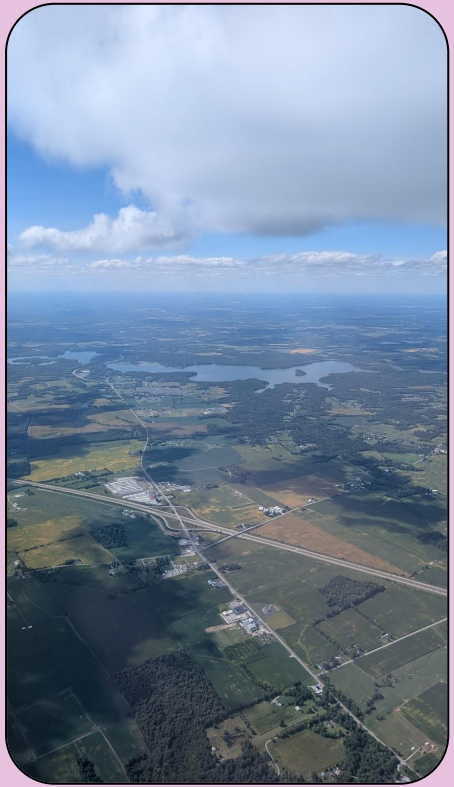
- ***Indexing pipeline:*** preprocess → embeddings → normalize → upsert to vector DB
- ***Metric:*** Cosine vs dot vs L2 — normalize embeddings for cosine similarity
- ***Hybrid search:*** Combine keyword + vector for high precision/recall
- ***Metadata & filtering:*** Store extra fields (timestamp, tenant, access control)
- ***Incremental updates:*** Add or refresh embeddings when corpus changes

DEMO



Fine Tuning

- Pre-trained models are generalists (know a little about everything)
- Fine-tuning = teaching a specialist
- Analogy: Medical school (pre-training) → Residency in cardiology (fine-tuning)
- We're adapting Qwen3 0.6B to become an IoT/MQTT expert



Why Not Just Use Prompts?

Approach	How it works	Trade-offs
Prompting	You are an IoT expert...	Temporary, uses context window, inconsistent
RAG	Retrieve relevant docs at runtime	Good for facts, adds latency, needs vector DB
Fine-tuning	Knowledge baked into weights	Permanent, no context cost, consistent behavior

Fine-tuning benefits:

- Consistent behavior without long system prompts
- Domain-specific vocabulary and patterns
- Smaller models can match larger ones on specific tasks
- Runs on edge devices (Raspberry Pi)

Thank You

Vector Databases and Embeddings Demystified

Vector Databases and Embeddings Demystified

Building AI-Powered Search in Python

References

- PineconeDB Vectors:
  - https://www.pinecone.io/learn/vector-embeddings/
  - https://www.pinecone.io/learn/series/faiss/hnsw/
- OpenAI
  - https://platform.openai.com/docs/guides/embeddings
- Wikipedia
  - https://en.wikipedia.org/wiki/Semantic_similarity
- Other
  - https://alexop.dev/posts/how-to-implement-a-cosine-similarity-function-in-typescript-for-vector-comparison/#step-by-step-example-calculation
  -

Section 1

FAISS

Chapter Title

What Is FAISS?

- **FAISS** = *Facebook AI Similarity Search*, an open-source library from Meta/Meta AI. Wikipedia+2GitHub+2
- Written in **C++** with Python wrappers; supports CPU and GPU (CUDA) implementations. GitHub+1
- Designed for **efficient similarity search** (nearest neighbor) and clustering of high-dimensional dense vectors.

Why FAISS Is Powerful

- **Scalable**: Handles datasets ranging from millions to **billions** of vectors. geeksforgeeks.org
- **High Performance**: GPU acceleration can massively speed up search. DataCamp
- **Flexible**: Choose tradeoffs — exact vs approximate, memory vs speed. GitHub+1
- **Widely used**: Often used inside vector databases; serves as benchmark in ANN research. Wikipedia

When (and Why) to Use FAISS

- When you have **very large embedding datasets** (e.g., millions+ vectors) and need fast, accurate similarity search. DataCamp+1
- When **GPU resources are available**, and you want to leverage them for faster search. GitHub
- When you need to **optimize memory** by compressing vectors (e.g., via PQ). DataCamp
- For **batch processing** of queries — FAISS is optimized for handling many queries at once. GitHub+1
- When you want **local control** over indexing and search (i.e., building search into your own system, not relying on a managed vector DB). FAISS is a library, *not* a full database. Designveloper+1

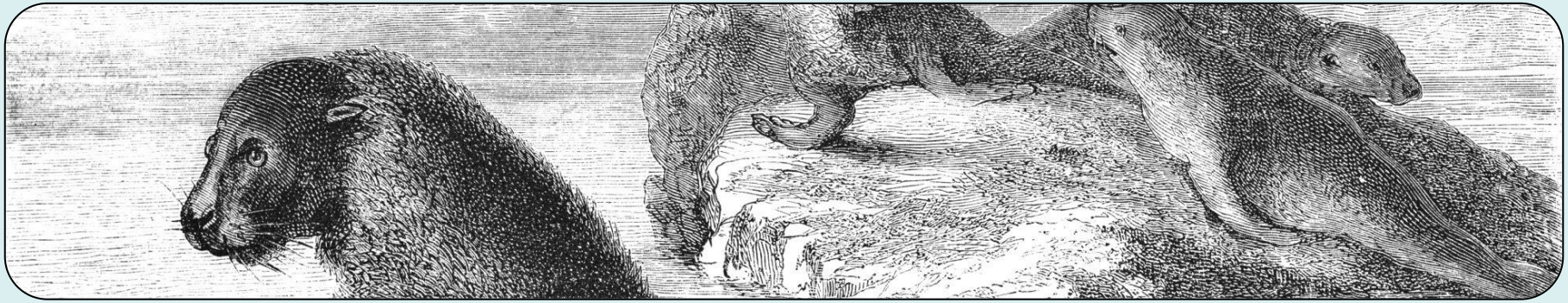
Analytics

MTEB

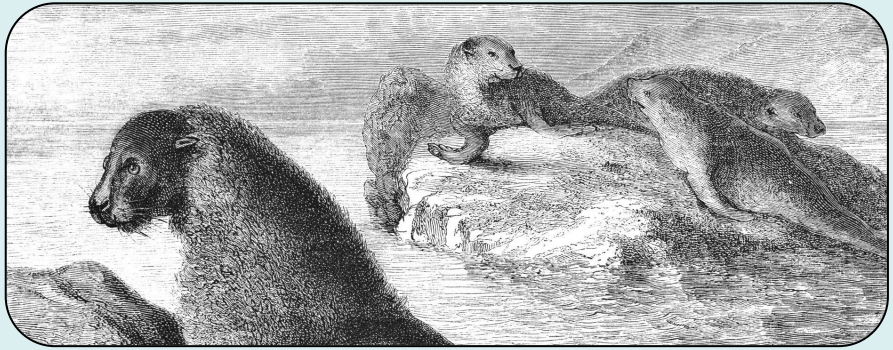
https://huggingface.co/spaces/mteb/leaderboard

TODO: Clustering Scale

-



THANK YOU



Jackie Gleason

Vector Databases & Architecture

- The Challenge: 1 billion vectors × 1536 dimensions
- Naive approach: 25 minutes per query (unacceptable!)
- Vector databases solve this with specialized algorithms
- Goal: Millisecond queries on billions of vectors
- Trade-off: 99% accuracy for 1000× speed

HNSW - Hierarchical Navigable Small Worlds

Highway System for Vectors

Multi-layer graph structure:

Layer 3 (Top): Highways - long jumps, fast navigation
Layer 2: State roads - medium distance
Layer 1: Local streets - fine-grained
Layer 0 (Bottom): All vectors - precise results

Performance:

- Search Time: O(log n) - only ~30-40 nodes visited
- Memory: High (2-5× vector size)
- Accuracy: 95-99% recall
- Used by: Chroma, Weaviate, Qdrant, Elasticsearch

✅ Best for: High accuracy, read-heavy, sufficient RAM
❌ Avoid: Tight memory, frequent updates

IVF & Product Quantization

Two Complementary Techniques

IVF (Inverted File Index) - Clustering

- Cluster vectors into groups (like library sections)
- Search only relevant clusters
- 1B vectors, 10K clusters = search ~100K (99.99% reduction)
- Accuracy: 90-95% recall

Product Quantization (PQ) - Compression

- Split vector into chunks, quantize each
- 1536 floats × 4 bytes = 6,144 bytes
- PQ: 8 bytes (768× compression!)
- Accuracy: 85-95% recall

IVF + PQ Combined = Production Standard

- Used by: Faiss, Milvus, Pinecone
- Billions of vectors, millisecond queries, reasonable memory

Choosing the Right Algorithm

Algorithm Comparison

Algorithm	Speed	Memory	Accuracy	Best Scale
HNSW	O(log n)	High	95-99%	<100M
IVF	O(n/c)	Medium	90-95%	10M-500M
PQ	Fast	Low	85-90%	Any
IVF + PQ	Fast	Low	90-95%	100M+

Decision Framework:

- < 1M vectors → HNSW (best accuracy)
- 1M-100M vectors → IVF or HNSW
- 100M+ vectors → IVF + PQ
- Tight memory → PQ or IVF + PQ

Key Takeaway: No single "best" - depends on constraints

GROUP/CLASS PROJECT #1

A Digital Storybook:
'Book Name'

Name 1 | Name 2 | Name 3 | Name 4

INTRODUCTION

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud

TABLE OF CONTENTS

STUDENTS

Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Name

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

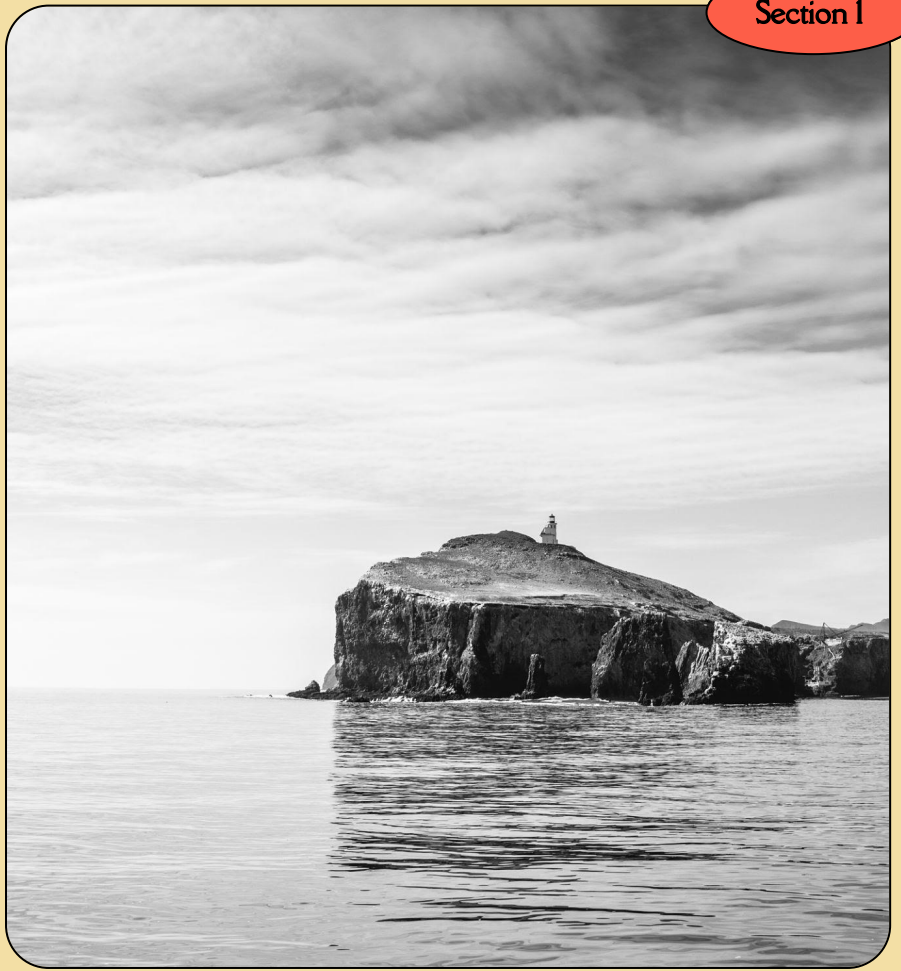
STORYBOOK

Section 1

Chapter Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla.



STORYBOOK

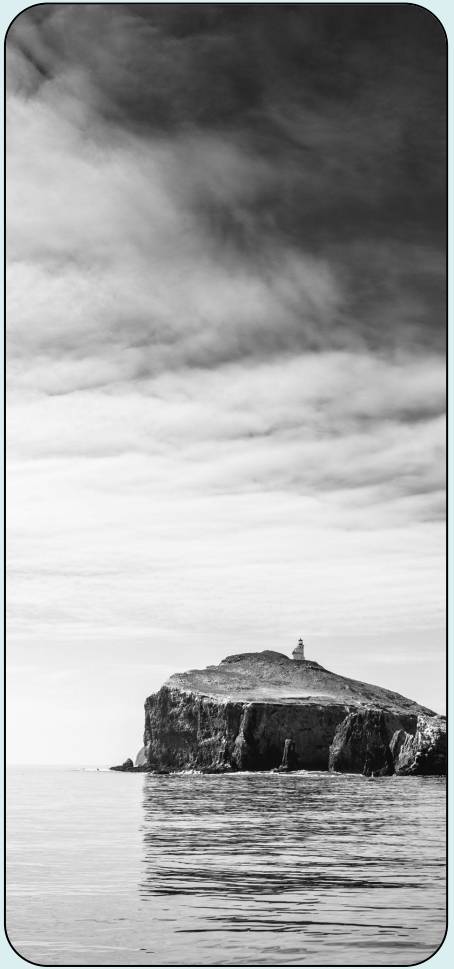
Section 1, Chapter Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla pariatur.

Excepteur occaecat cupidatat non proident. sunt in culpa qui officia deserunt mollit anim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna.



Section 1

STORYBOOK

Chapter Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla pariatur. Excepteur occaecat cupidatat non proident. sunt in culpa qui officia deserunt mollit anim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation.

Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla pariatur. Excepteur occaecat cupidatat non proident. sunt in culpa qui officia.

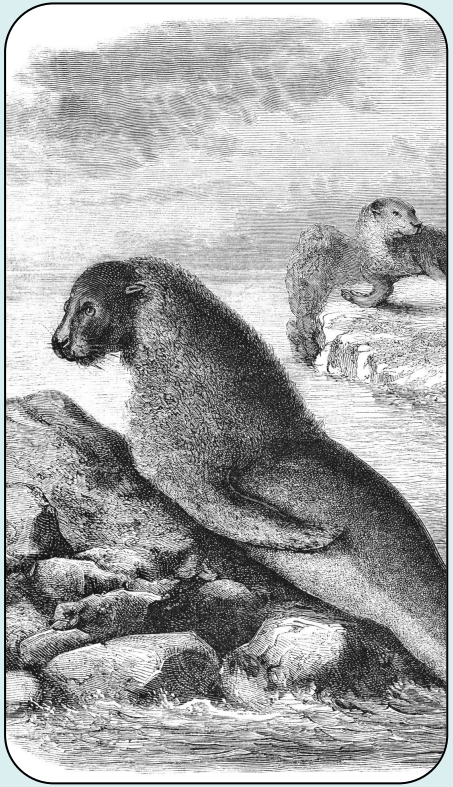
TITLE

STUDENT 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

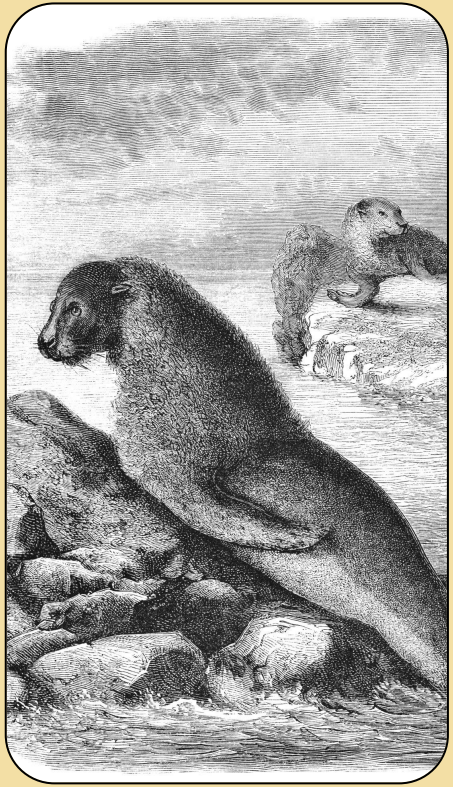
Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla pariatur. Excepteur occaecat cupidatat non proident. sunt in culpa qui officia deserunt mollit anim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore.



STUDENT 1

TITLE



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

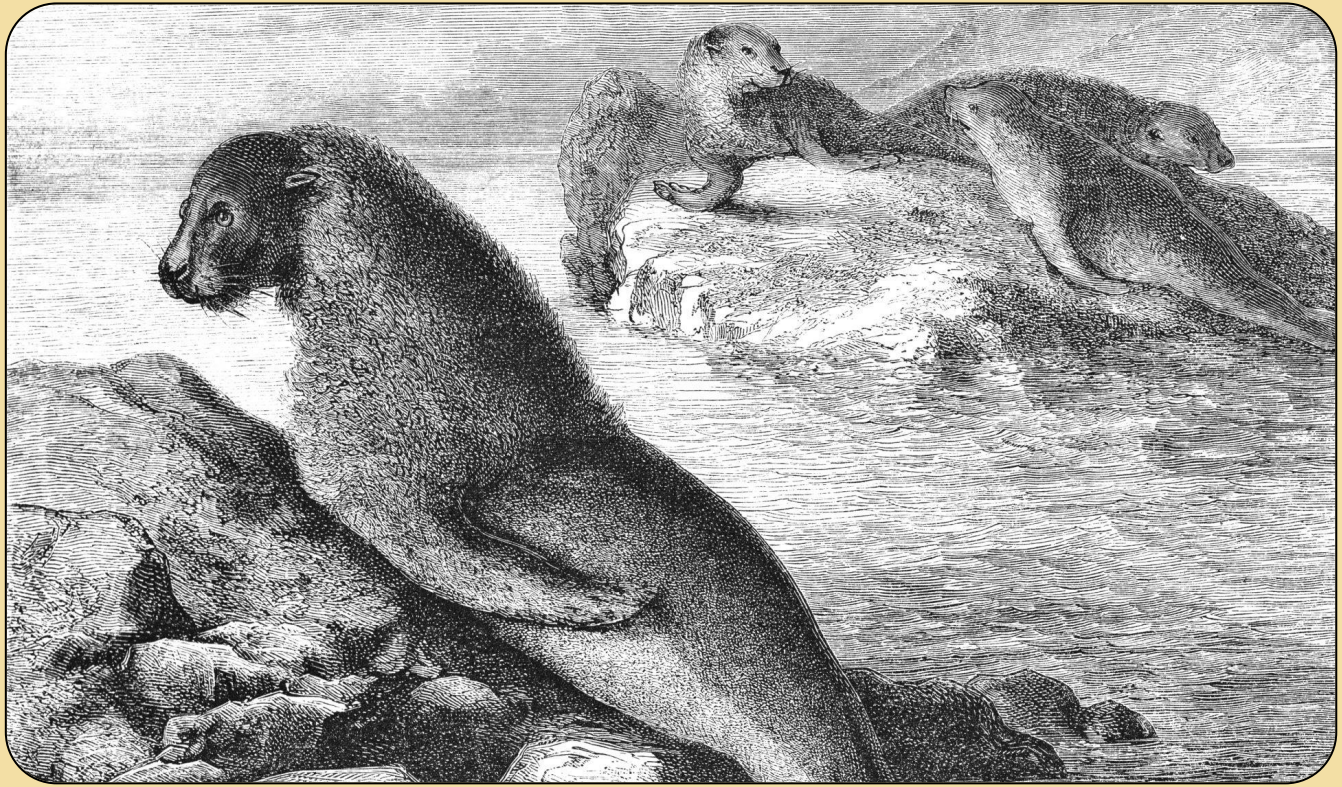
Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla pariatur. Excepteur occaecat cupidatat non proident. sunt in culpa qui officia deserunt mollit anim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore.

STUDENT 1

TITLE

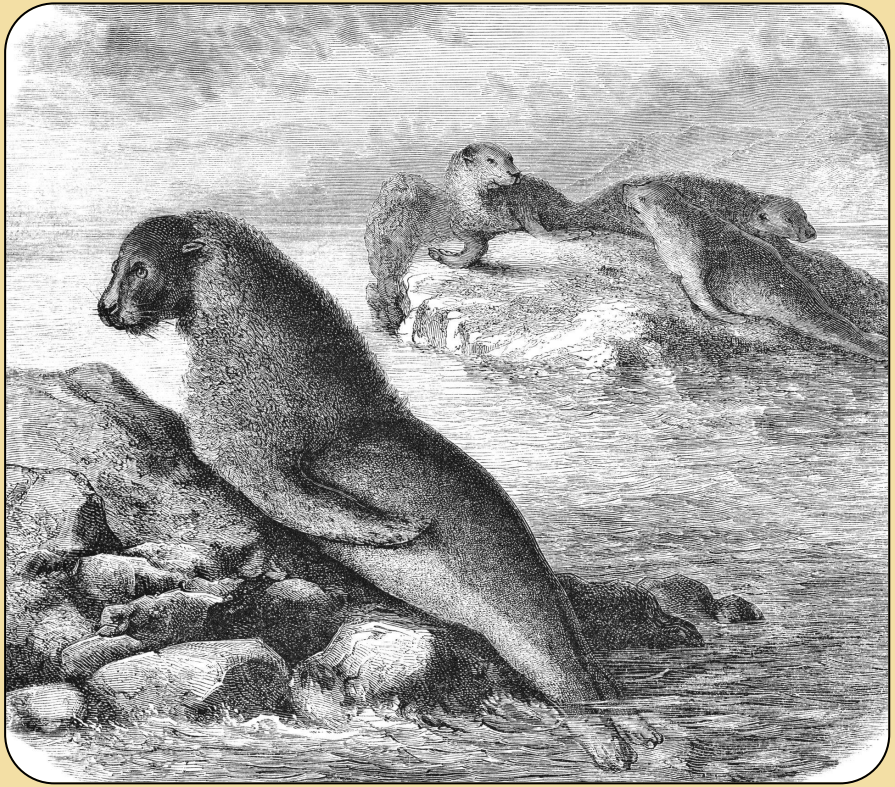
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis.



STUDENT 1

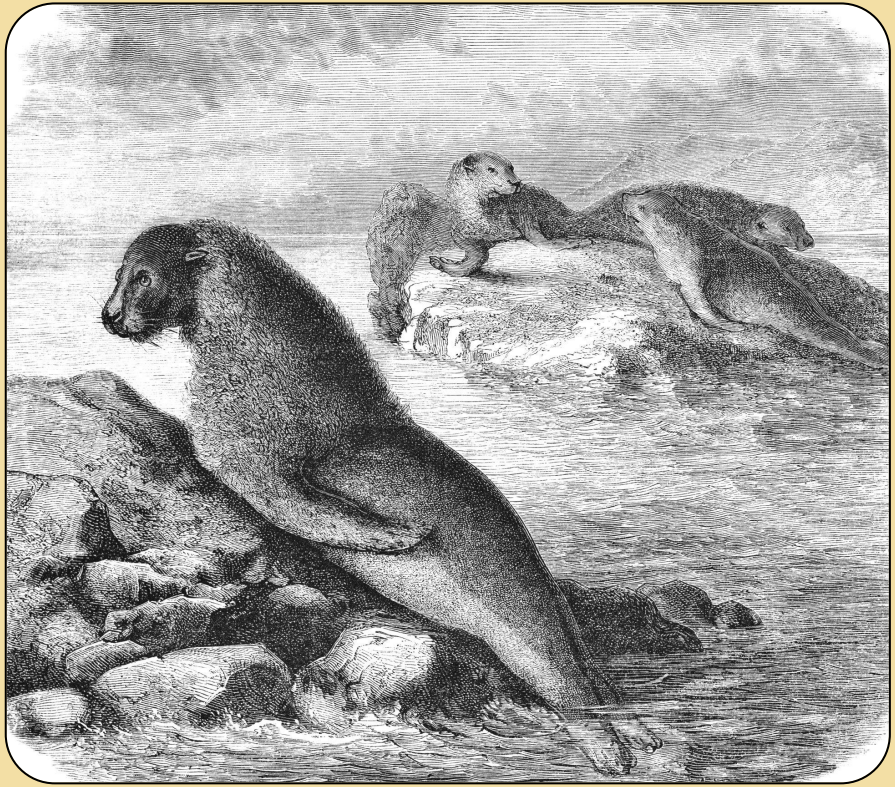
TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dignissim id est vitae posuere. Duis eu quam convallis, feugiat turpis in, tempor elit. Integer tristique sit amet tellus in eleifend. Nulla bibendum lacus ut ex gravida, sit amet porttitor lacus pretium. Sed auctor neque quis augue scelerisque, ut auctor lacus gravida. Vivamus mattis ut sapien nec lacinia. Quisque id fringilla nulla.



STUDENT 1

TITLE

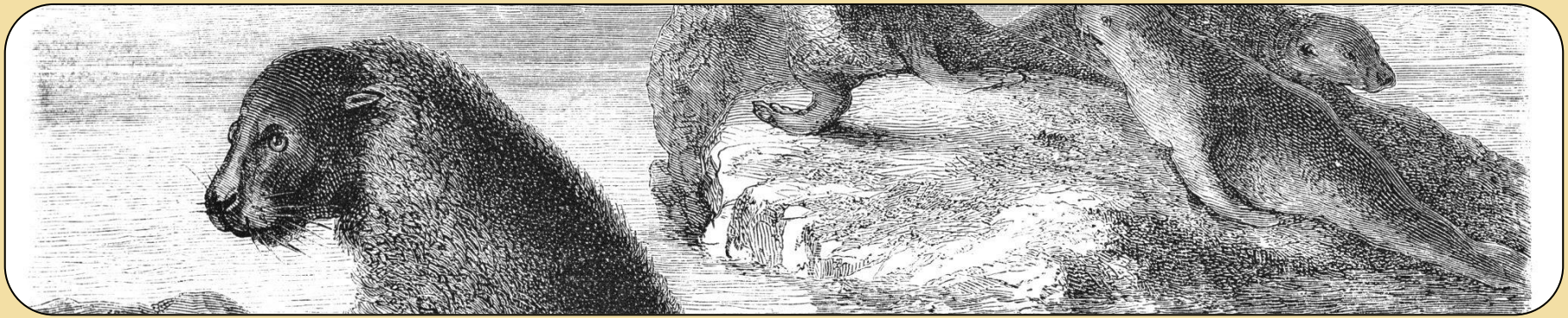


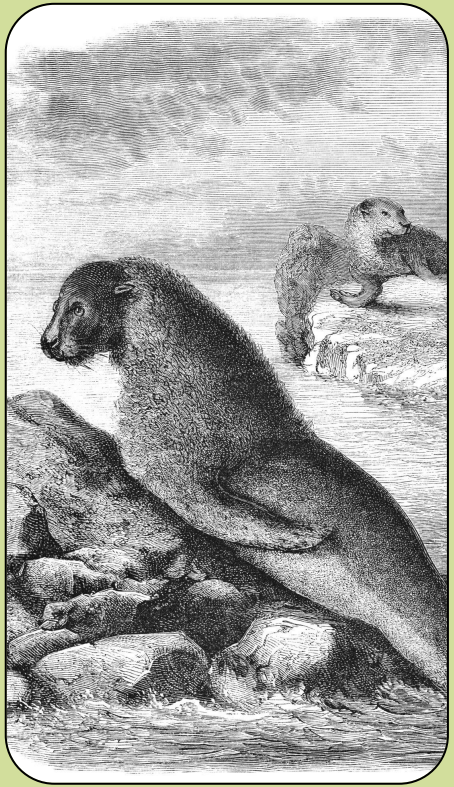
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dignissim id est vitae posuere. Duis eu quam convallis, feugiat turpis in, tempor elit. Integer tristique sit amet tellus in eleifend. Nulla bibendum lacus ut ex gravida, sit amet porttitor lacus pretium. Sed auctor neque quis augue scelerisque, ut auctor lacus gravida. Vivamus mattis ut sapien nec lacinia. Quisque id fringilla nulla.

STUDENT 1

TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dignissim id est vitae posuere. Duis eu quam convallis, feugiat turpis in, tempor elit. Integer tristique sit amet tellus in eleifend. Nulla bibendum lacus ut ex gravida, sit amet porttitor lacus pretium. Sed auctor neque quis augue scelerisque, ut auctor lacus gravida. Vivamus mattis ut sapien nec lacinia. Quisque id fringilla nulla.





STUDENT 3

TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla pariatur. Excepteur occaecat cupidatat non proident. sunt in culpa qui officia deserunt mollit anim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore.

PAGE TITLE HERE

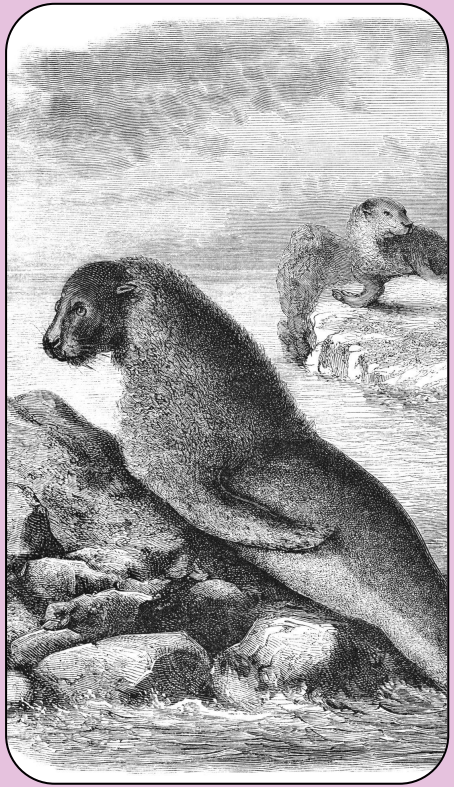
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla pariatur. Excepteur occaecat cupidatat non proident. sunt in culpa qui officia deserunt mollit anim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore.

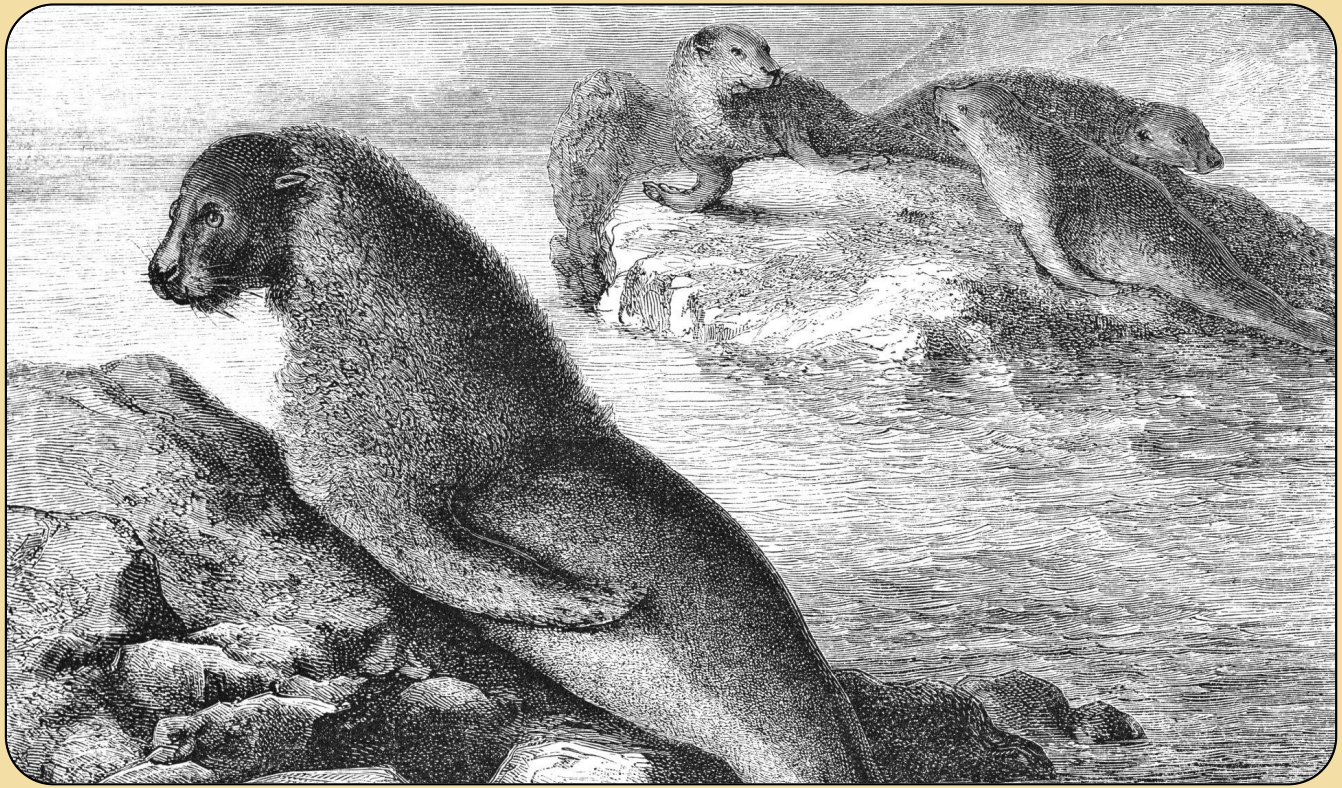


STUDENT 4

STUDENT 1

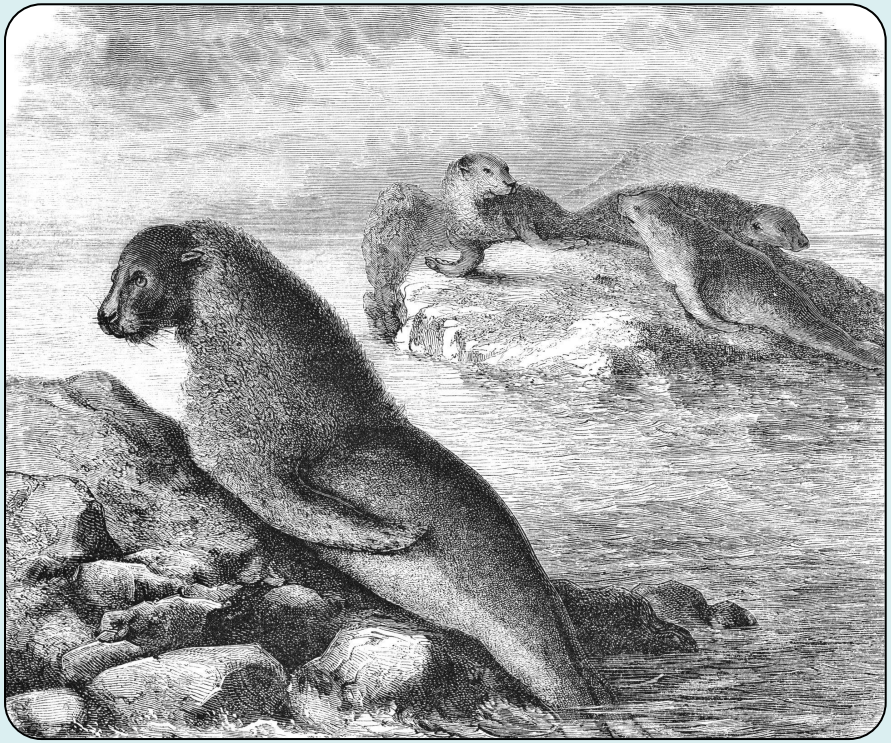
TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Student Name 2

PAGE TITLE HERE



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

SUMMARY

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad, quis nostrud exercitation ullamco laboris ut aliquip ex ea commodo consequat.

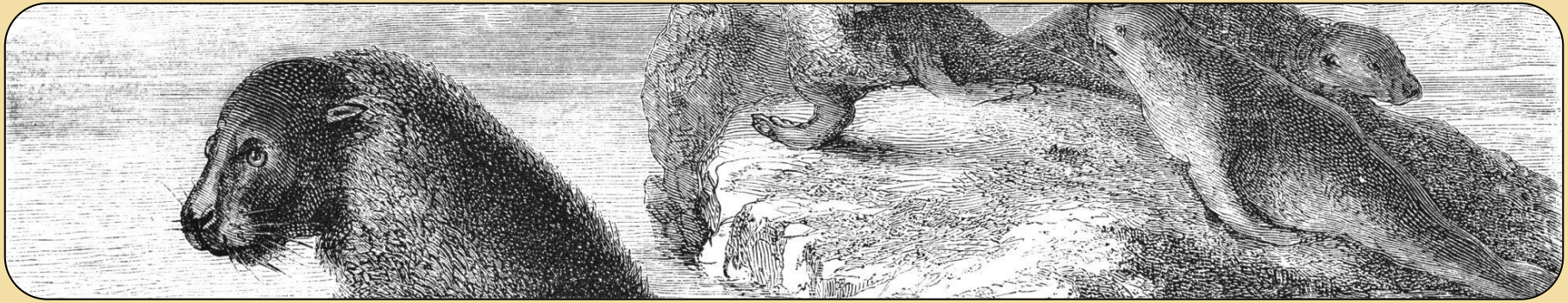
Duis aute irure dolor in reprehenderit in voluptate cillum dolore eu fugiat nulla pariatur. Excepteur occaecat cupidatat non proident. sunt in culpa qui officia deserunt mollit anim.

SUMMARY

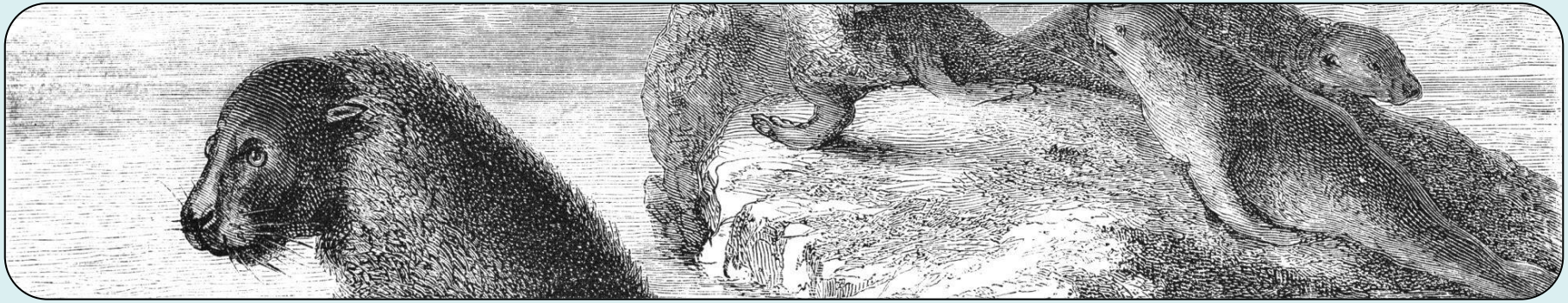
1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

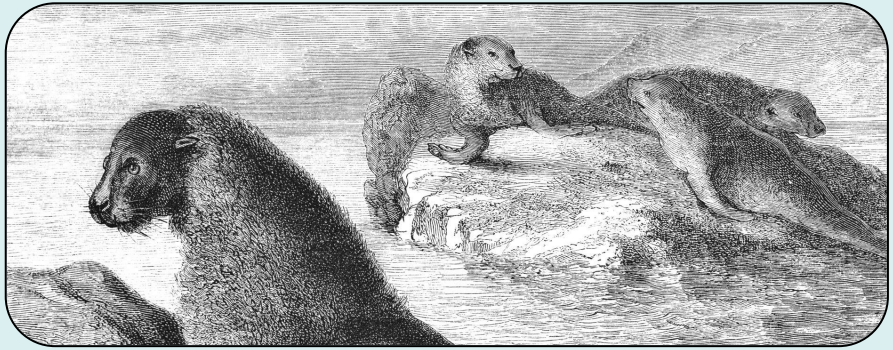
3. Lorem ipsum dolor sit amet.



THANK YOU



THANK YOU



Student Name 1
Student Name 2
Student Name 3
Student Name 4

king - man + woman = queen

Word Arithmetic That Actually Works!

Results:
queen: 0.8932 ← Closest!
monarch: 0.8654
princess: 0.8543
royal: 0.8234

What's happening:
king = [royalty] + [male] + [leadership]
- man = [male] + [adult]
+ woman = [female] + [adult]
= [royalty] + [leadership] + [female] = queen

This isn't magic - it's geometry!
The embedding model learned relationships from context.

Other examples: paris - france + germany ≈ berlin