Class 4: The Amazing World of Embeddings

Giving Words Superpowers to Help Computers Understand!

Natanael Waraney Gerald Massie, MCIT

May 14, 2025

What We'll Explore Today!

Welcome, Word Explorers!

- Ever wonder how computers can seem so smart with words?
- Today, we pull back the curtain on a key piece of that magic!
- Our adventure: Understanding something called "Embeddings."
- Don't worry, we'll keep it fun and simple!

Humans are Word Pros vs. The Computer's Challenge

- We humans understand language nuances:
  - "Happy" is the opposite of "sad."
  - A "kitten" is a baby "cat."
- Computers, at their core, understand numbers (0s and 1s).
- To a computer, "apple" and "orange" are just different strings of letters. It doesn't inherently know they're both fruits.
- How can we bridge this gap? By turning words into numbers!

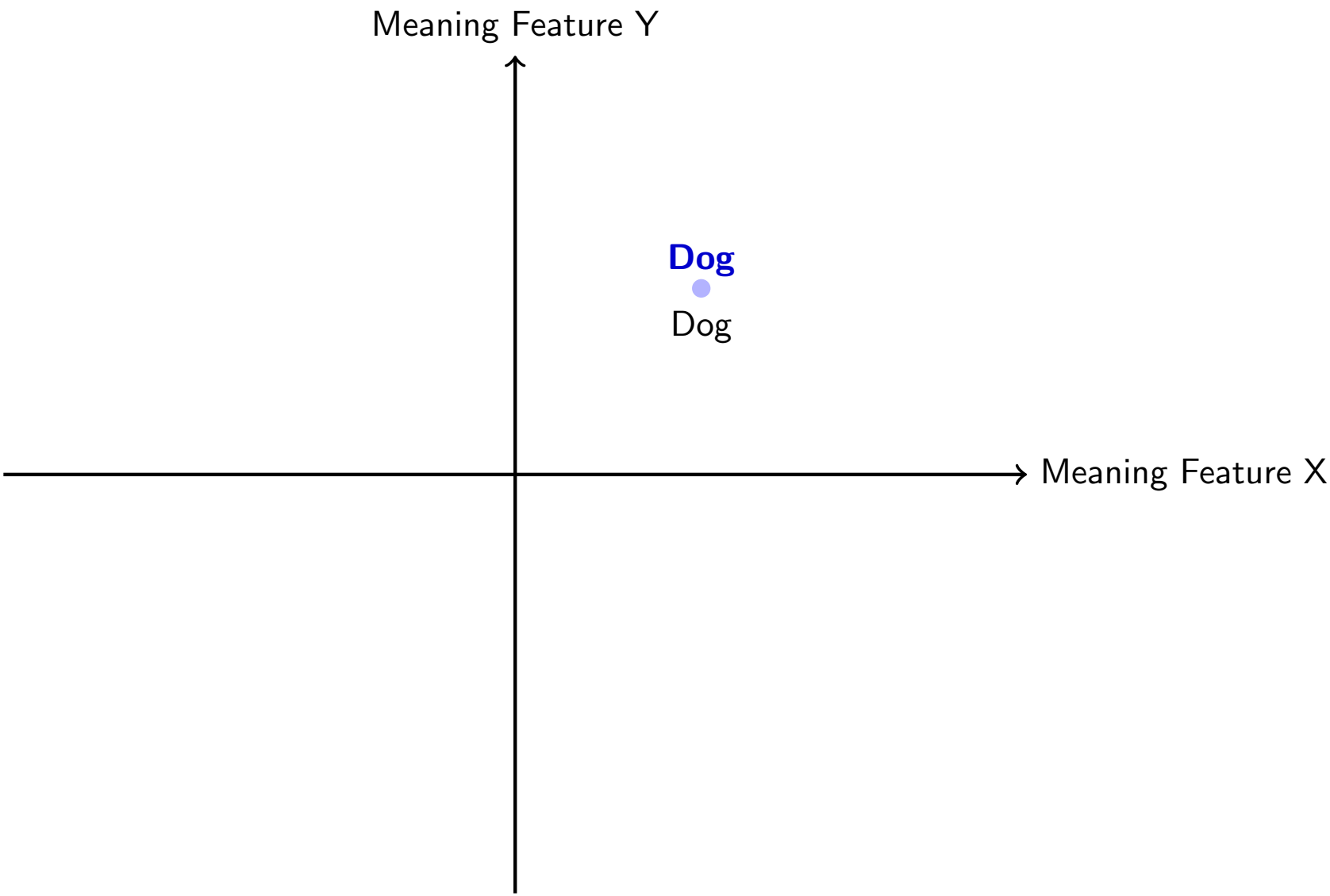
Embeddings: The Big Idea

- Embeddings are **numerical representations** of text (words, phrases, sentences).
- Think of them as a special "number code" or "fingerprint" for each piece of text.
- This code is a list of floating-point numbers, also called a **vector**.
- The goal? To capture relationships and meaning.

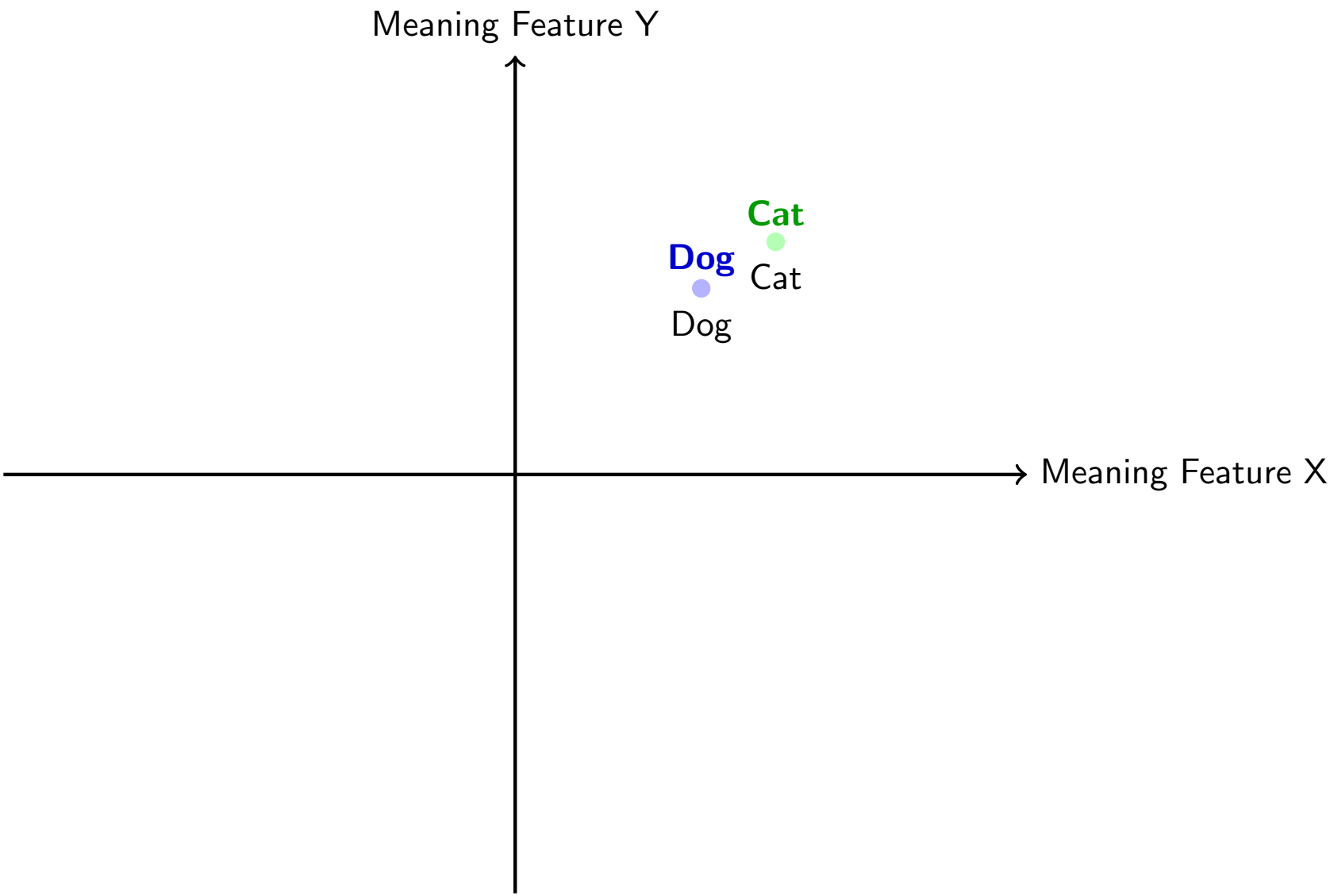
Analogies to Understand Embeddings

- **Secret Codes:**
  - "King": `[0.5, 1.2, -0.3, ...]`
  - "Queen": `[0.6, 1.1, -0.2, ...]` (Similar code to King!)
  - "Apple": `[-2.1, 0.1, 1.5, ...]` (Different code!)
- **GPS Coordinates for Words:** Words with similar meanings are "located" close together in a conceptual "meaning map." (Let's see this now!)
- **Word's "Personality Profile":** Numbers representing different abstract features (e.g., how "royal," how "edible").

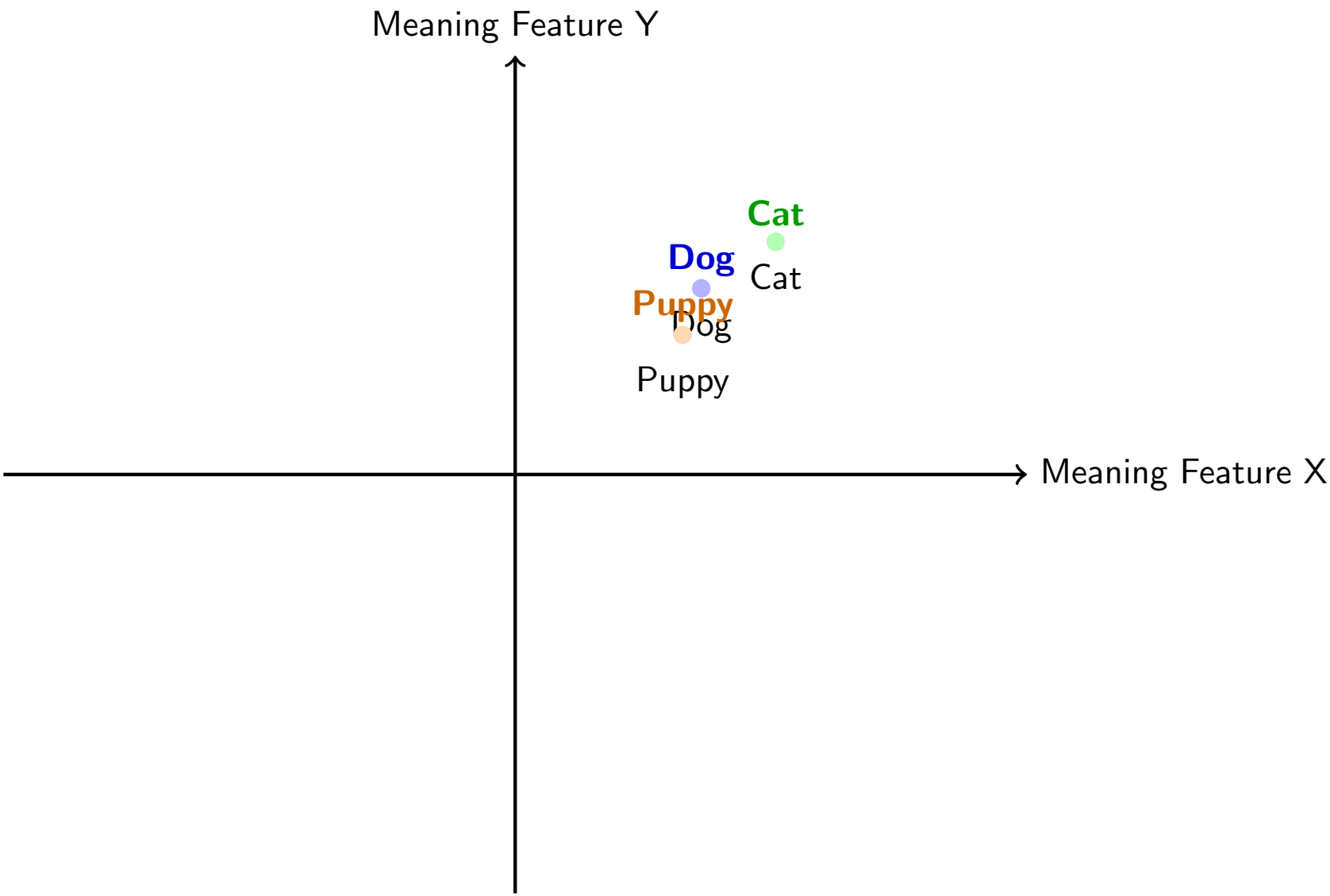
Visualizing the "Word Map" - Building Clusters



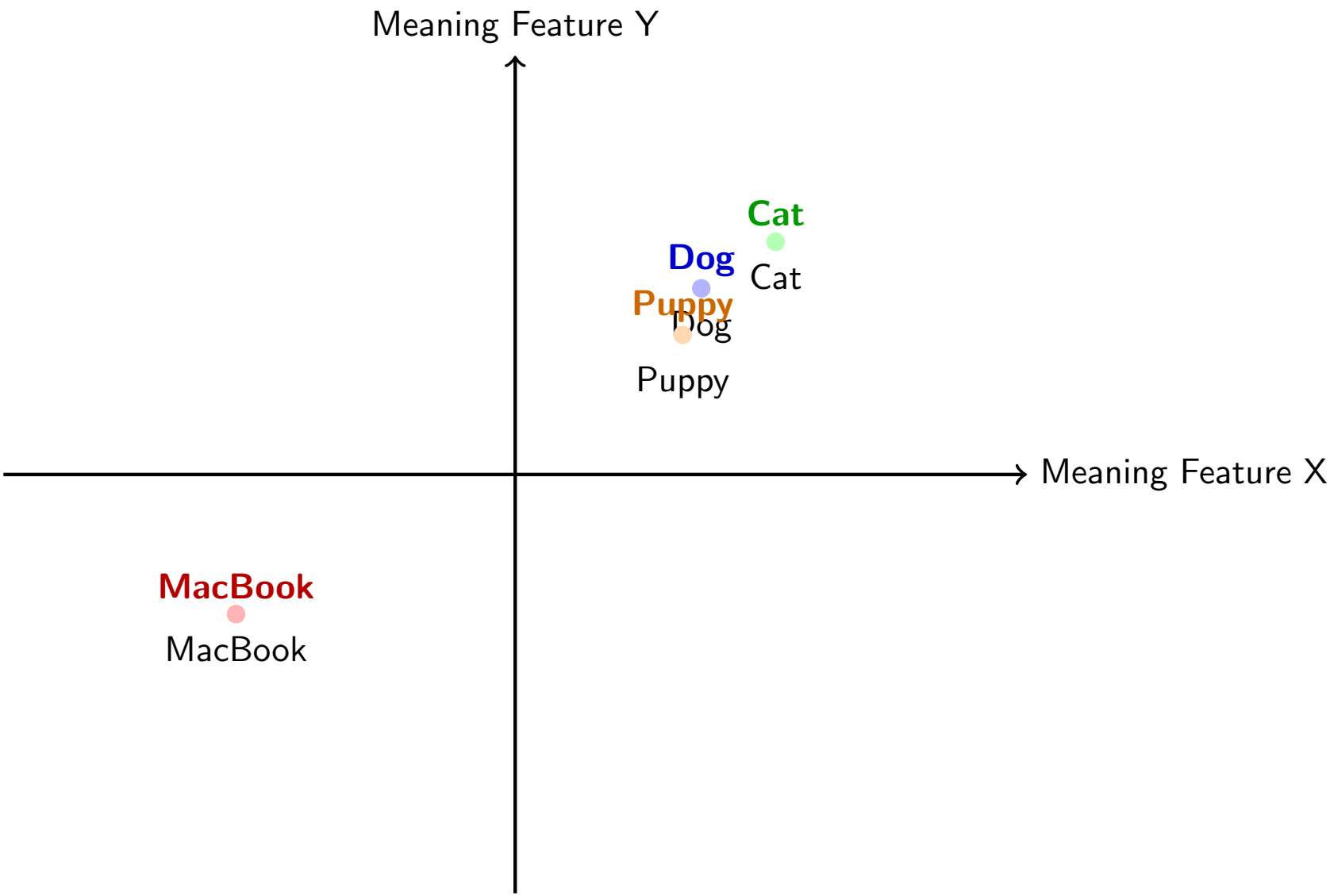
Visualizing the "Word Map" - Building Clusters



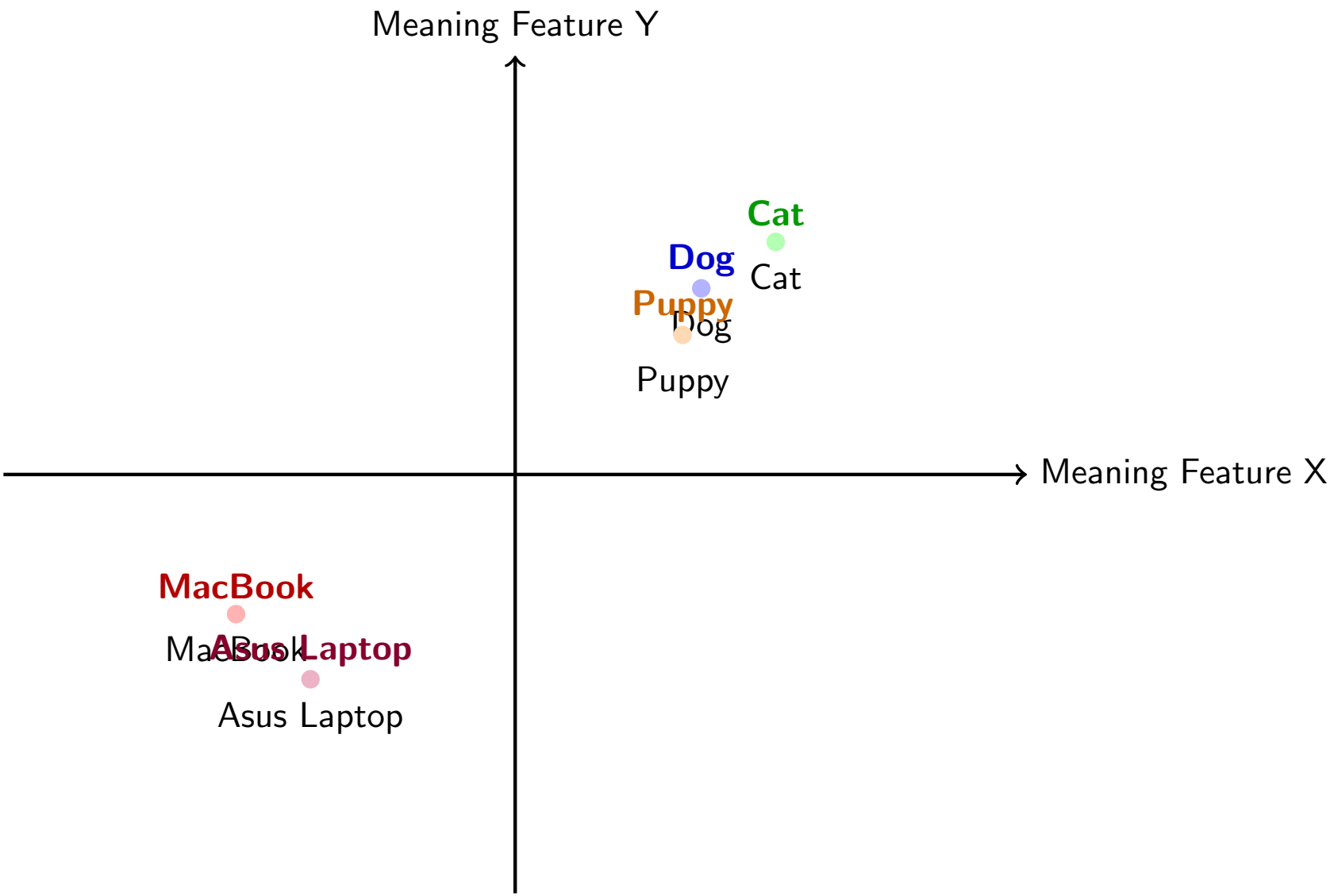
Visualizing the "Word Map" - Building Clusters



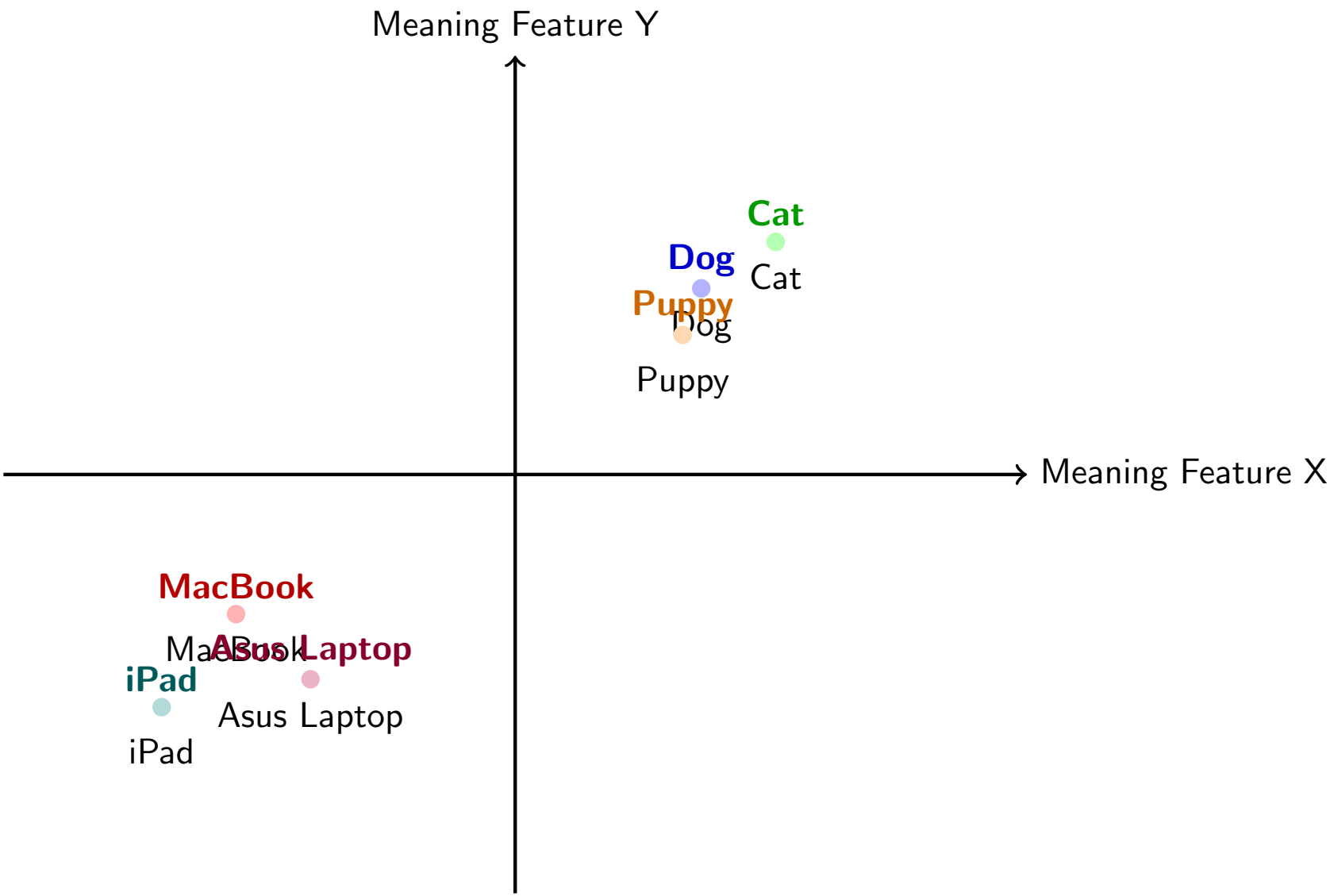
Visualizing the "Word Map" - Building Clusters



Visualizing the "Word Map" - Building Clusters



Visualizing the "Word Map" - Building Clusters



Dimensions Meaningful Closeness (Recap)

- The list of numbers (vector) has a length, its **dimensionality** (e.g., hundreds of numbers).
- Each number represents some learned aspect of meaning/context.
- **Key Idea:** Text with similar meanings gets "closer" embedding codes.
  - "I took my dog to the vet" vs. "I took my cat to the vet" = close.
  - "cat" vs. "dog" = closer than "cat" vs. "car."
- "Closeness" is measured mathematically (e.g., **cosine similarity**).

Bridging the Gap: Words to Numbers

- Computers excel with numbers.
- Embeddings translate words/text into this computer-friendly numerical format.
- This allows computers to:
  - Compare different texts and understand how they relate.
  - Perform calculations to find patterns and relationships.
  - Go beyond simple keyword matching to grasp conceptual similarity.
- Essentially, embeddings make computers "smarter" with language.

Learning from Examples: The Power of Text

- AI models create embeddings by "reading" and processing vast amounts of text (books, articles, websites).
- The core idea: **Words that occur in similar contexts tend to have similar meanings.** This is the distributional hypothesis.
- The model learns by looking at words and their "neighbors" (the context words around them).
- It tries to predict which words are likely to appear near a given word. As it gets better at this, it generates the embeddings.

Who Makes Embedding Models?

- Various tech companies and research institutions develop these AI models.
- Examples:
  - Google's Gemini API offers several embedding models (e.g., 'gemini-embedding-exp-03-07', 'text-embedding-004').
  - OpenAI provides models like 'text-embedding-3-small' and 'text-embedding-3-large'.
- These are often pre-trained, meaning they've already learned from a massive dataset and are ready to use.

The Evolution of "Finding Stuff" in Text

- **Level 1: Ctrl+F (Literal Match)**
  - You type "apple". It finds exactly "apple". Simple and fast!
  - But it won't find "apples" or "Apple" (if case-sensitive) or "fruit".
- **Level 2: Regex (Super Patterns - Week 2!)**
  - You write a pattern like
    '[A-Za-z0-9.*Itfindsallemailaddresses*!*Muchmorepowerfulforstructuredtext*.
- Still, you need to know the exact pattern you're looking for.

Level 3: Embeddings - The "Meaning Search" (This Week!)

- What if you want to find an **idea**, a **concept**, or something described **in context**?
- You don't know the exact words. You don't know a specific pattern.
- **Embeddings give us a "Super-Duper-Crazy-Ctrl+F"!**
  - You search with a phrase, a sentence, or even a whole paragraph describing what you want.
  - It finds things that are **semantically similar** – close in meaning – even if the words are totally different!
  - This is because it's comparing the "meaning codes" (embeddings).

Examples of "Meaning Search" with Embeddings

- **Your Query:** "Fun activities for a sunny weekend outdoors with family"
  - **Embedding Search Might Find:**
    - "Plan a picnic at the park with the kids."
    - "Ideas for a family bike ride when the weather's good."
    - "Beach day games everyone will enjoy."
- **Your Query:** "customer complaints about slow software performance"
  - **Embedding Search Might Find:**
    - "Users frustrated with application lagging."
    - "The program takes forever to load."
    - "Feedback on UI unresponsiveness."
- Notice how the words are different, but the context and meaning are captured!

Use Case Overview (Recap)

- This "meaning search" is core to many applications:
  - Semantic Search / Information Retrieval
  - Clustering
  - Recommendations
  - Classification
  - Anomaly Detection
  - Question Answering / RAG
  - And more like Diversity Measurement, Code Search!

Use Case: Super-Smart Search (Information Retrieval)

- Challenge: Find relevant information even if the exact keywords aren't used.
- Embeddings Solution:
  - Convert your search query into an embedding.
  - Convert documents/items to be searched into embeddings.
  - Find documents whose embeddings are "closest" (most similar) to the query's embedding.
- Example: Search "healthy snacks for children," find articles on "nutritious fruit for kids."
- This is about finding **semantically similar** text.

Use Case: Grouping Similar Items (Clustering)

- Challenge: Discover underlying groups or themes in a large text collection (e.g., customer reviews, news articles).
- Embeddings Solution:
  - Convert each text item into an embedding.
  - Items with similar meanings will have embeddings that are close together in the "meaning map."
  - Algorithms can then automatically group these "close" embeddings into clusters.
- Example: Grouping customer feedback into "positive experiences," "shipping issues," "product feature requests."

Use Case: Automatic Labeling (Classification)

- Challenge: Automatically assign predefined categories to text (e.g., spam detection, topic labeling).
- Embeddings Solution:
  - Train a model using examples of text already labeled with the correct categories, using their embeddings as input.
  - The model learns what embedding patterns correspond to each category.
  - For new, unlabeled text, generate its embedding and the model predicts the most likely category.
- Example: Classifying news articles as "Sports," "Business," or "Technology."

Use Case: Personalized Recommendations

- Challenge: Suggest items (movies, products, songs) a user might like.
- Embeddings Solution:
  - Represent items and user preferences/history using embeddings.
  - If a user liked items with certain embedding characteristics...
  - ...recommend other items whose embeddings are similar to those liked items, or to the user's overall preference embedding.
- Example: "Because you watched 'Space Adventure X', you might like 'Galaxy Quest Y'."

Use Case: Finding the Odd One Out (Anomaly Detection)

- Challenge: Identify unusual or unexpected data points in a set of text.
- Embeddings Solution:
  - Convert all text items into embeddings.
  - Anomalous or outlier text will have embeddings that are "far away" from the main cluster of embeddings.
- Example: Identifying a fraudulent review that's very different in tone/content from genuine reviews.

Use Case: Question Answering (RAG)

- Challenge: Answer questions based on a given set of documents.
- Embeddings Solution (Retrieval Augmented Generation - RAG):
  - Embed the user's question.
  - Embed chunks of text from the available documents.
- Retrieve the document chunks whose embeddings are most similar to the question embedding. (Gemini mentions $\text{`RETRIEVAL}_{Q}\textit{UERY`and`RETRIEVAL}_{D}\textit{OCUMENT`tasktypes}).\textit{Aseparatelangu}$
- Note: Sometimes questions and answers aren't semantically similar directly. Task-specific embeddings help here.

Use Case: Code Search Vector Databases

- **Code Search**:
  - Embed natural language queries (e.g., "function to sort an array").
  - Embed code snippets/functions.
- Match the query embedding to the closest code embeddings. (Gemini has 'CODE$_R$*ETRIEVAL*$_Q$*UERY*').
- **Vector Databases**:
  - When working with many embeddings, you need efficient storage and search.
  - Vector databases are specialized to store embedding vectors and perform fast similarity searches.

Not All Embeddings Are Created Equal

- Different models (e.g., from Google Gemini, OpenAI) produce different embeddings.
- **Model Choice**: Some models are newer, larger, or designed for better multilingual performance.
- **Dimensionality**:
  - Models can produce embeddings of different lengths (dimensions), e.g., 256, 1536, 3072.
  - Larger dimensions can capture more detail but cost more and use more resources.
  - Some APIs allow you to choose/reduce dimensionality.
- **Task Types (Gemini API)**: You can specify the task (e.g., 'SEMANTIC$_S$*IMILARITY*', '*CLASSIFICATION*', '*RETRIEVAL*$_D$*OCUMENT*')

Embeddings Analogies: "Word Math"

- A fascinating property: Embeddings can capture relational similarities.
- This allows for "vector arithmetic" with word meanings.
- Famous example: **vector("King") - vector("Man") + vector("Woman") ≈ vector("Queen")**
- How it works (conceptually):
  - The subtraction "King - Man" might isolate the concept of "royalty" without maleness.
  - Adding "Woman" introduces femaleness to that "royalty" concept.
- This shows embeddings learn complex relationships from data! Other examples: Paris - France + Italy ≈ Rome.

(Conceptual image: Arrows showing vector math, like the parallelogram model)

Caution 1: Bias in Embeddings

- Embeddings learn from the text data they are trained on.
- If this data contains societal biases or stereotypes (e.g., gender roles, racial biases), the embeddings can unfortunately learn and even amplify these biases.
- Example: If old texts associate "doctor" with men and "nurse" with women, embeddings might reflect this, leading to unfair outcomes in applications like job searches.
- This is a significant ethical challenge ("allocational harm" or "representational harm"). Researchers are actively working on "debiasing" techniques, but it's an ongoing problem.

Caution 2: Knowledge Cutoffs Limitations

- **Knowledge Cutoff**: Embedding models are trained on data up to a certain point in time. They won't know about very recent events or newly coined terms. (e.g., OpenAI models mentioned a Sept 2021 cutoff).
- **Not True Understanding**: While powerful, embeddings don't "understand" or "reason" like humans. They are sophisticated pattern matchers based on statistical relationships in data.
- **Context Window Size**: The amount of surrounding text used during training (context window) can affect what kind of similarities the embeddings capture (e.g., short windows might capture more syntactic similarity, long windows more topical relatedness).

The Grand Recap: Embeddings Made Easy!

- **What?** Number codes (vectors) for text, capturing meaning.
- **Why?** Help computers "understand" word relationships numerically.
- **How?** AI models learn from context in massive text datasets.
- **Use Cases?** Smart search, recommendations, classification, clustering, QA, and much more!
- **Cool Trick?** "Word Math" like King - Man + Woman $\approx$ Queen.
- **Cautions?** Be mindful of potential biases and limitations.

Embeddings: A Key to Modern AI

- Embeddings are a fundamental building block in many modern Natural Language Processing (NLP) and AI systems.
- They are working behind the scenes in many applications you use daily.
- Understanding their basic principles helps demystify how AI works with language!

Further Exploration Questions

- **To learn more, you could explore:**
  - Specific embedding models like Word2Vec, GloVe, fastText.
  - Advanced models like BERT and Transformers that build upon these ideas.
  - The documentation for Gemini API and OpenAI API for embeddings.
- Relevant links:
  - Gemini Document Search Tutorial: `https://github.com/google/generative-ai-docs/blob/main/site/en/gemini-api/tutorials/document_search.ipynb`
  - OpenAI Embeddings information (refer to their main API docs).

Questions?