From Words to Worlds: Embeddings in the Age of LLMs

How Computers Went from Reading Words to Understanding Language

Prof. Dr. Jörg Osterrieder

Methods and Algorithms — MSc Data Science

Spring 2026

How Did We Get Here?

How did we go from computers that can't read... to ChatGPT?

Today we trace the journey: from simple word vectors to large language models.

XKCD #1838 by Randall Munroe (CC BY-NC 2.5).

Learning Objectives

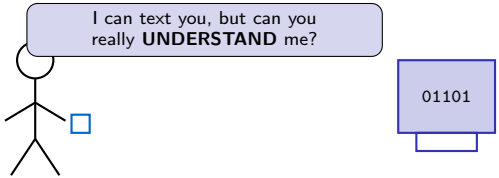
By the end of this lecture you will be able to:

1. **Explain** what word embeddings do and why they matter
2. **Identify** the limit of static embeddings (polysemy)
3. **Describe** how LLMs use contextual embeddings to understand language

---

Everything is explained with pictures and analogies.

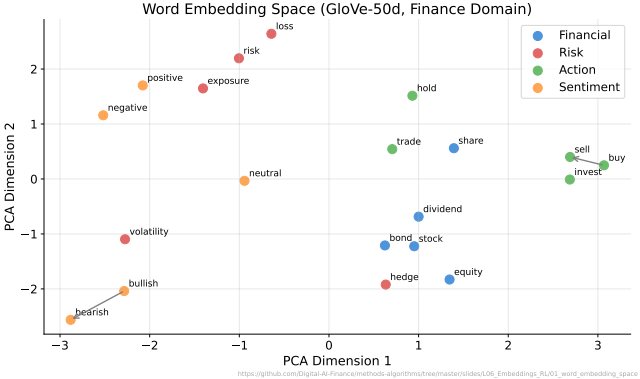
The Problem: Can Computers Really Understand Us?



Computers see characters. We need them to understand **meaning**.

From Siri to ChatGPT — it all starts with turning words into numbers.

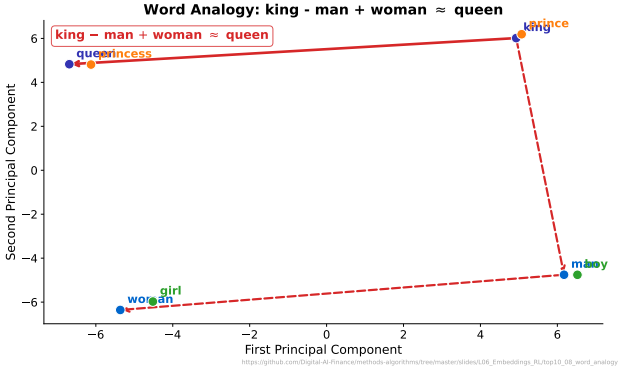
Words as Points in Space



- Word2Vec solved this — words become **points in space**
- Similar words cluster together
- The computer learned these positions by reading text

Word2Vec (2013) was the breakthrough: learn word meaning from context.

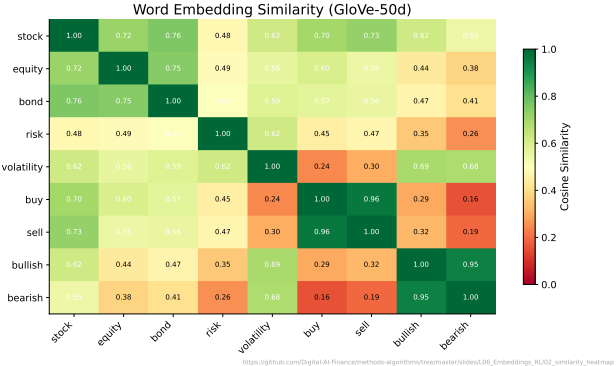
The “Aha Moment”: Vector Arithmetic on Words



- King − Man + Woman = Queen
- Paris − France + Germany = Berlin
- Embeddings capture relationships as **vector arithmetic**

This is the “aha moment”: meaning has direction in embedding space.

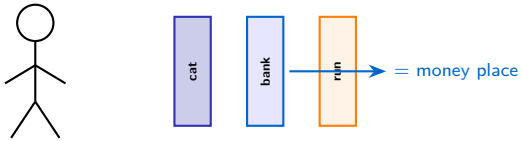
Measuring Word Similarity



- Dark squares = similar words
- Light squares = unrelated words
- The computer learned this from text alone

Cosine similarity measures the angle between word vectors. Small angle = similar meaning.

Static Embeddings = A Dictionary

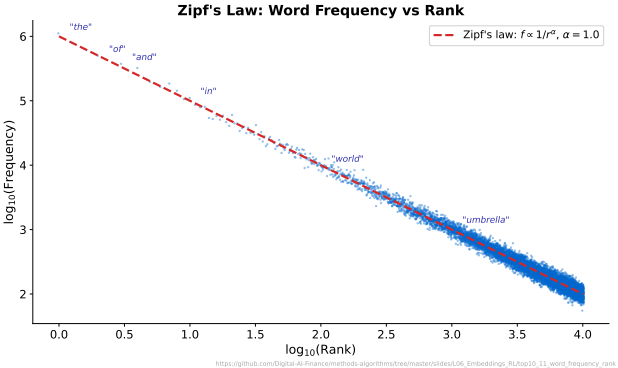


One word, one definition, forever.

Word2Vec gives each word **ONE fixed vector**. Great for analogies.
But is one definition enough?

Static embeddings: fast, efficient, but frozen. Every use of "bank" maps to the same point.

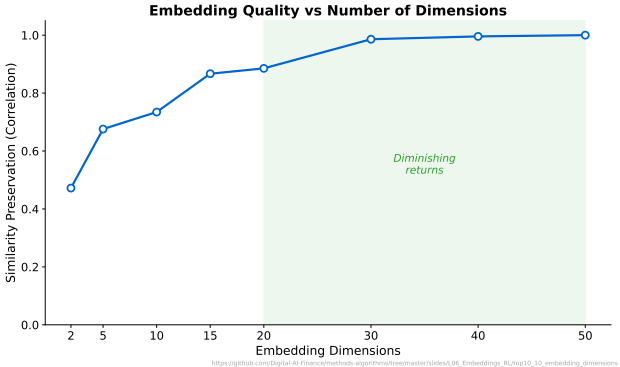
How Models Learn: Word Frequency Patterns



- Words follow Zipf's law (few common, many rare)
- Embeddings learn from word co-occurrence patterns
- LLMs read trillions of words — they see every pattern

The more text a model reads, the better its embeddings. LLMs read the entire internet.

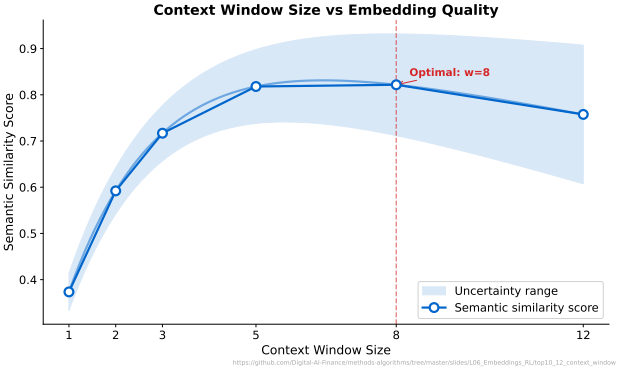
More Dimensions = More Nuance



- More dimensions = more nuance in meaning
- This chart shows up to 50 dimensions
- BERT uses 768 dimensions. GPT-4 uses over 12,000

Dimension count is a key difference: Word2Vec uses 100–300, modern LLMs use thousands.

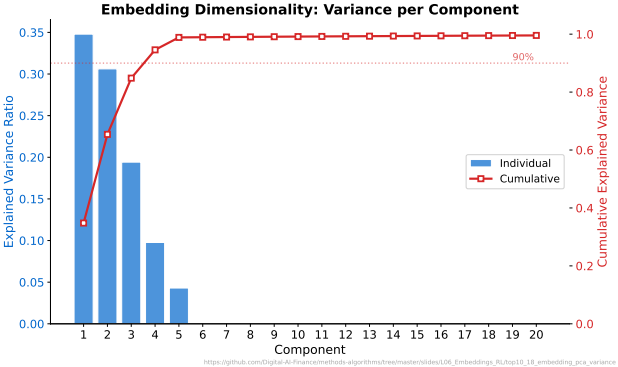
The Context Window: How Far Can a Model See?



- Word2Vec looks at a small context window (5–10 words)
- Bigger windows = better embeddings
- What if you could look at **everything** at once?

Word2Vec: 5–10 word window. BERT: 512 tokens. GPT-4: 128,000 tokens.

Embeddings Have Internal Structure



- Embeddings have internal structure
- A few principal components capture most meaning
- LLMs build richer, deeper structure with more layers

PCA reveals that embeddings are not random — they encode linguistic structure.

But Wait — One Word, Many Meanings!

"I deposited money at the **bank**"

"I sat on the river **bank**"

Word2Vec gives **both** sentences the exact same vector for "bank".
That's clearly wrong! The word means completely different things.

The Polysemy Problem

This is called **polysemy** — and static embeddings can't handle it.

Polysemy (many meanings) is everywhere: "run" has 645 definitions in the dictionary.

What If Embeddings Could Change?

What if “bank” near “river” got a **DIFFERENT** vector

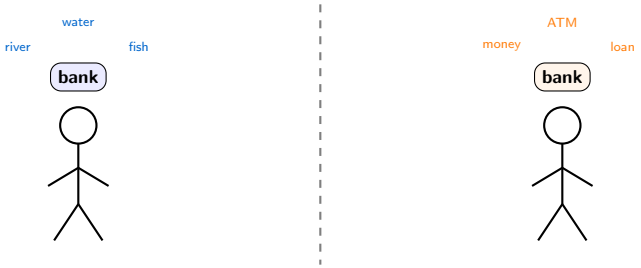
than “bank” near “money”?

What if the embedding could read the surrounding words and **adapt**?

This is exactly what contextual embeddings do.

The key insight of 2018: embeddings should depend on context, not just the word itself.

Context Changes Everything

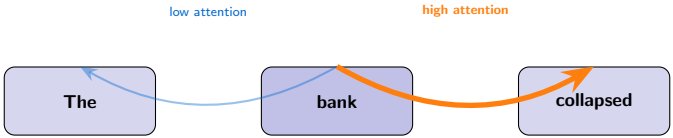


Same word, different meaning — because the context is different!

Contextual embeddings compute a **new vector every time**, based on surrounding words.

BERT (2018) and GPT (2018) were the first widely successful contextual embedding models.

How It Works: Words Talk to Each Other

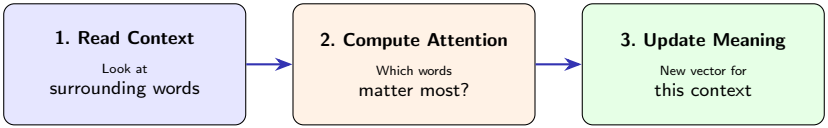


Each word asks: which other words help me understand **my** *meaning?*

This is called **self-attention**: every word looks at every other word to decide its meaning.

Attention is the core mechanism of transformers — the architecture behind BERT and GPT.

How Contextual Embeddings Work



That's it. Read context, weigh importance, update the embedding. Repeat for every word.

BERT reads all words at once. GPT reads left-to-right. Both use attention to weigh which words matter most.

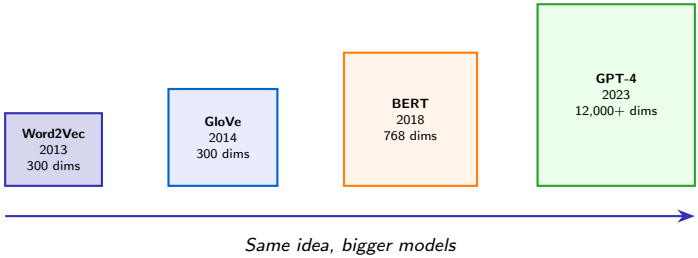
Static vs Contextual: What Changed?

Aspect	Static (Word2Vec)	Contextual (BERT/GPT)
Vector	One per word, fixed	New one each time
Model size	Small ($\sim$100 MB)	Large ($\sim$1–100 GB)
Year	2013	2018+
"bank"	Always same vector	Different per context
Example	Word2Vec, GloVe	BERT, GPT-4

Same fundamental idea (words as vectors), but now they're **dynamic**.

The jump from static to contextual was the single biggest leap in NLP history.

The Scaling Story: Same Idea, Bigger Models



GPT-3 has 175 billion parameters. GPT-4 is estimated at over 1 trillion.

What Can LLMs Do?

- **Generate text** — write emails, reports, code
- **Answer questions** — from documents, databases, knowledge
- **Translate** — between languages, styles, formats
- **Reason** — multi-step logic, math, planning

One Foundation

All built on the same foundation: **embeddings + attention**.

Every capability of ChatGPT traces back to contextual embeddings and self-attention.

LLMs in Finance

Understanding

- Sentiment analysis of earnings calls
- ESG report classification
- Contract similarity search

Acting

- Automated report generation
- Document Q&A for compliance
- Market news summarization

Key Insight

Every LLM application starts with **embeddings**.

Bloomberg, JPMorgan, and Goldman Sachs all deploy LLMs for financial text analysis.

Try It Yourself

Classic (Word2Vec)

```
from gensim.models import Word2Vec
model = Word2Vec(sentences,
                 vector_size=100)
model.wv.most_similar("bank")
```

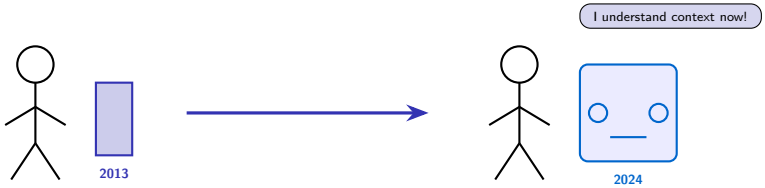
Modern (OpenAI API)

```
from openai import OpenAI
client = OpenAI()
response = client.embeddings.create(
  input="bank loan application",
  model="text-embedding-3-small")
```

2013: train your own. 2024: call an API.

Classic: pip install gensim. Modern: pip install openai (requires API key).

The Journey: From Dictionary to Conversation Partner



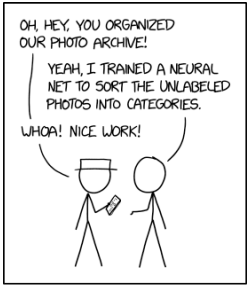
The same core idea — words as vectors — but now with context, scale, and understanding.

Five Things to Remember

1. Embeddings turn words into **meaningful numbers**
2. Static embeddings: one vector per word — fast but miss context
3. Contextual embeddings (BERT/GPT): new vectors based on surrounding words
4. LLMs = contextual embeddings scaled to **billions of parameters**
5. Every AI language tool — from search to ChatGPT — is built on embeddings

From 300 dimensions to 12,000+. From megabytes to terabytes. Same core idea.

Go Teach a Machine Something



XKCD #2173 by Randall Munroe (CC BY-NC 2.5). Next: try the Jupyter notebook to build your own embeddings!