Word Embedding Methods in NLP

From distributional semantics to retrieval-augmented generation

Max Pellert

[Slides as PDF]

The core question

How do you represent **meaning** in a computationally tractable way?

Simplest approach could be a numerical one-hot encoding: a vocabulary of 50,000 words, each word a vector of 49,999 zeros and one 1.

```
"cat"  → [0, 0, 1, 0, 0, ..., 0]
"dog"  → [0, 1, 0, 0, 0, ..., 0]
```

Problems:

- Vectors are super sparse and “cat” and “dog” are exactly as similar as “cat” and “democracy”
- No useful semantic information is encoded
- Dimensionality grows with vocabulary

The distributional hypothesis

> "You shall know a word by the company it keeps." — J.R. Firth, 1957

Words that appear in **similar contexts** tend to carry **similar meanings**.

"I adopted a **cat** / **dog** / **rabbit**." "The **cat** / **dog** / **rabbit** sat on the mat."

Context statistics *are* semantic content. This one insight underlies every method we will cover today.

The thread running through this entire class:

Embeddings operationalise semantics and make *similarity between texts* directly and cheaply measurable. This powerful capability is the engine behind retrieval, attention, and behavioural measurement of AI.

Word2Vec: learning from context

Skip-gram objective: given a word, predict its neighbours.

Train a shallow neural network on this task across billions of word windows. The hidden layer weights *become* the embedding – a dense, low-dimensional vector.

```
"cat"  → [ 0.21, -0.54,  0.87, ...,  0.03]   # 300 dimensions
"dog"  → [ 0.19, -0.51,  0.84, ...,  0.07]   # similar!
"bank" → [-0.12,  0.33, -0.41, ...,  0.61]   # far away
```

The model was never told what “similar” means. It learned it from co-occurrence statistics alone.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space (arXiv:1301.3781). arXiv. http://arxiv.org/abs/1301.3781

The geometry of meaning

Semantic similarity becomes **geometric proximity**.

And directions carry meaning:

$$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} \approx \overrightarrow{\text{queen}}$$

This works because the training objective compresses contextual information into a space where **semantic relationships become linear transformations**.

Animate/inanimate, gender, tense, geography, all these concepts emerge as interpretable axes. The model was never explicitely told any of this.

Similarity is the core operation

Everything downstream reduces to one question: **how close are these two vectors?**

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|}$$

This single number – easy to compute, cheap at scale – is what makes for example the following possible:

- **Retrieval:** find the passage most similar to a query
- **Clustering:** group texts by mutual similarity
- **Bias auditing:** measure which concepts cluster near which groups
- **Semantic shift:** track how the meaning of words changes over time by training embeddings on historical corpora from different periods

And it is the same operation inside the transformer's famous **attention mechanism**.

From words to sentences

Word2Vec gives one vector **per word type**, regardless of context.

“I went to the **bank** to deposit money.” “We sat on the **bank** of the river.”

Same word. Different meaning. **Same vector.**

Transformer models (BERT, 2018) produce **context-sensitive** token representations. Each token’s embedding depends on all other tokens in the sequence.

Sentence transformers (Reimers & Gurevych, 2019) extend this to full sentences – fine-tuned so that semantically similar sentences are close in embedding space.

How do we know which embeddings are good?

MTEB: Massive Text Embedding Benchmark (Muennighoff et al., 2022) – 56 datasets across 8 task categories including retrieval, semantic similarity, clustering, classification, and reranking

huggingface.co/spaces/mteb/leaderboard

Large models (7B+) dominate overall – but task type matters more than size.

- **Retrieval:** use models fine-tuned on retrieval (*bge-large*, *e5-mistral*)
- **Semantic similarity:** general sentence transformers often suffice
- **Speed at scale:** *all-MiniLM-L6-v2* (22M params) within a few points of SOTA

> **Don't just pick any embedding model.**
>
> Check MTEB for your specific task type. In my own work on SQuID, MTEB ranking turned out to be a good guide even for psychometric tasks.

What else can you do with sentence embeddings?

Beyond retrieval and classification – **Question:** if meaning lives in embedding space, do **psychological constructs** also have geometric structure there?

We embedded the text of items from the **Schwartz Portrait Values Questionnaire (PVQ-RR)** using a sentence transformer.

Result: the Schwartz value structure (universalism, power, achievement, benevolence…) re-emerges from the geometry of the embedding space alone.

Pellert, M., Lechner, C. M., Sen, I., & Strohmaier, M. (2025). Neural network embeddings recover value dimensions from psychometric survey items on par with human data. In Findings of the Association for Computational Linguistics: EACL 2026, Rabat, Morocco. Association for Computational Linguistics. https://doi.org/10.48550/arXiv.2509.24906

SQuID: Survey and Questionnaire Item Embeddings Differentials

The problem: raw embedding similarities are always positive – shared abstract linguistic features inflate pairwise similarity even between *opposing* items. Opposing values should be *negatively* correlated.

The fix: subtract the mean embedding over all questionnaire items

$$\mathbf{y}_i - \bar{\mathbf{y}}, \quad \text{where } \bar{\mathbf{y}} = \frac{1}{57}\sum_{i=1}^{57} \mathbf{y}_i$$

This removes shared linguistic features and recovers negative correlations.

Result: the circular Schwartz value structure emerges from embeddings on par with human data from 49 countries. No finetuning, no labels, any model.

> Embeddings can be an **instrument for representational and behavioural measurement** of AI systems.

Now: a practical application

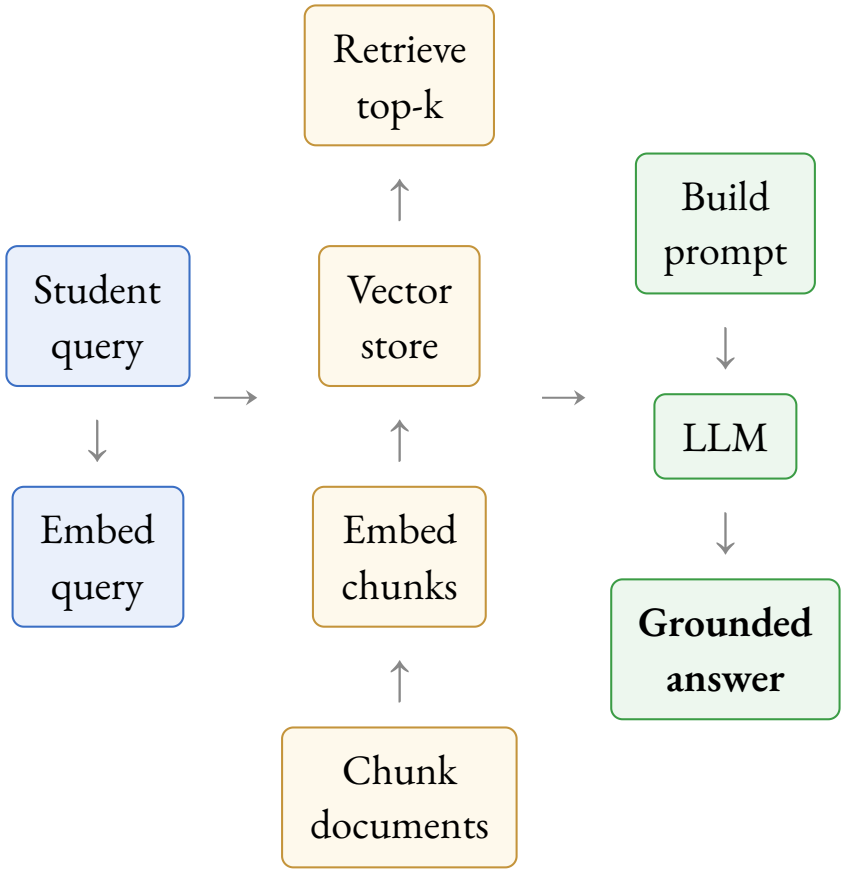
Let's build something using an embedding approach.

Scenario: CEU DNDS has course descriptions, faculty pages, and research group texts.

A new master's student asks: *"Which courses or research groups work on network resilience?"*

- Keyword search: fails if the document says "robustness" or "cascading failures"
- LLM without context: will hallucinate a confident, plausible, wrong answer
- **Retrieval-augmented generation (RAG) with sentence embeddings:** retrieves semantically relevant passages, generates a grounded answer

RAG architecture



Four steps: **chunk, embed, retrieve, generate**.

Step 1 · Chunk the documents

```
# DNDS course descriptions and research group pages
documents = {
    "network_science_course": "This course introduces ...",
    "css_course": "Students learn to apply ...",
    "dnds_research_groups": "The department hosts ...",
}

chunk_size, overlap = 400, 50

all_chunks, all_labels = [], []
for name, text in documents.items():
    chunks = [
        text[i : i + chunk_size]
        for i in range(0, len(text), chunk_size - overlap)
    ]
    all_chunks.extend(chunks)
    all_labels.extend([name] * len(chunks))
```

Step 2 · Embed the chunks

```python
from sentence_transformers import SentenceTransformer

# MTEB retrieval-task pick for this scale
model = SentenceTransformer("all-MiniLM-L6-v2")

# Shape: (n_chunks, 384)
chunk_embeddings = model.encode(all_chunks, show_progress_bar=True)
```

Step 3 · Retrieve by semantic similarity

```python
import numpy as np

def retrieve(query: str, k: int = 3) -> list[str]:

    # Embed the query in the same space as the chunks
    query_embedding = model.encode([query])

    # Cosine similarity -- the same geometry as the Schwartz analysis
    scores = np.dot(chunk_embeddings, query_embedding.T).squeeze()
    top_k_idx = np.argsort(scores)[::-1][:k]

    return [all_chunks[i] for i in top_k_idx]
```

“Which courses cover network resilience?” will surface passages mentioning “robustness”, “cascading failures”, “structural vulnerability” – because they live nearby in embedding space.

Step 4 · Generate a grounded answer

```python
import anthropic
client = anthropic.Anthropic()

def answer(query: str) -> str:
    context = "\n\n---\n\n".join(retrieve(query, k=3))

    prompt = f"""You are a helpful assistant for CEU's Department of
Network and Data Science. Answer using only the context provided.
If the answer is not in the context, say so clearly.

Context:
{context}

Question: {query}"""

    r = client.messages.create(
        model="claude-opus-4-5",
        max_tokens=512,
        messages=[{"role": "user", "content": prompt}]
    )
    return r.content[0].text

print(answer("Which courses cover network resilience?"))
```

Step 4 · Output

```
print(answer("Which courses cover network resilience?"))
```

OUTPUT

Based on the provided context, network resilience is most directly covered in the **Network Science and Applications** course, which includes dedicated sessions on robustness of complex networks, cascading failures, and targeted vs. random attack tolerance. The course uses both analytical tools and empirical network datasets to study how structural properties relate to a network's ability to withstand perturbations.

The **DNDS research groups page** additionally mentions ongoing work on systemic risk in financial and infrastructure networks within the Computational Social Systems group. No direct match was found for the exact phrase "network resilience" -- the above is based on semantically related content.

The unified picture

Everything today is the same operation: **Text → vector → geometry**

Use	What you embed	What the geometry gives you
Word2Vec	Words	Lexical similarity, analogies
Sentence transformers	Sentences	Semantic similarity
MTEB	Many tasks	Model quality, task fit
SQuID	Survey items	Psychological structure
RAG	Documents + query	Relevance, grounded generation

The representation shift the embeddings bring is the enabling substrate. The interesting question is always: **what structure lives in this space?**

Recap and open questions

Distributional hypothesis · Firth 1957

→ **Word2Vec** · static, word-level

→ **Sentence Transformers** · contextual, sentence-level

→ **MTEB** · evaluation across tasks

→ **SQuID** · psychometric measurement of LLMs

→ **RAG** · grounded generation from documents

Key takeaway: embeddings are a general-purpose representation of meaning. Every application is a different answer to: *what is close to what, and why does that matter?*

Open question for discussion: when does semantic similarity *fail* as a proxy for relevance – and what would you build with this for your own research?

Further reading: Mikolov et al. 2013 · Reimers & Gurevych 2019 · Muennighoff et al. 2022 · Lewis et al. 2020 · Pellert et al. 2026

Attention is learned similarity

In a transformer, each token asks: *which other tokens are most relevant to me right now?*

The answer is a similarity computation over learned query and key vectors:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

$\mathbf{Q}\mathbf{K}^\top$ is a **dot product similarity matrix** over token embeddings. The model learns *which* similarities matter for *which* tasks.

"Bank" in "river bank" attends to "river"; in "savings bank" it attends to "savings".

Same word, different similarity profile, different contextual embedding. The distributional hypothesis – now end-to-end differentiable.