Text Embeddings: Mathematical Foundations and Implementation

AI Tutor

Abstract

This document provides a comprehensive mathematical treatment of text embeddings, covering the fundamental concepts, implementation details, and practical applications. We explore how textual data is transformed into numerical vectors while preserving semantic relationships through geometric structures in high-dimensional spaces.

Contents

1 Constructing Example Embeddings

1.1 Methodology for Creating Interpretable Examples

Example: Designing a 2D Semantic Space

When creating pedagogical examples, we carefully choose dimensions that are human-interpretable. For the royalty-gender example:

Step 1: Define Semantic Axes

- **X-axis**: Royalty (0.0 = common, 1.0 = royal)
- **Y-axis**: Gender (0.0 = feminine, 1.0 = masculine)

Step 2: Assign Values Based on Semantic Properties

$$\begin{aligned} \phi(\text{"king"}) &= [0.95, 0.90] \quad \text{(very royal, very masculine)} \\ \phi(\text{"queen"}) &= [0.95, 0.10] \quad \text{(very royal, very feminine)} \\ \phi(\text{"man"}) &= [0.10, 0.85] \quad \text{(common, masculine)} \\ \phi(\text{"woman"}) &= [0.10, 0.15] \quad \text{(common, feminine)} \end{aligned}$$

Step 3: Verify Semantic Relationships

$$\begin{aligned} \phi(\text{"king"}) - \phi(\text{"man"}) &= [0.85, 0.05] \quad \text{(royal}\rightarrow\text{common)} \\ \phi(\text{"queen"}) - \phi(\text{"woman"}) &= [0.85, -0.05] \quad \text{(similar direction!)} \end{aligned}$$

1.2 Systematic Approach for N-dimensional Examples

Example: Creating a 4D Semantic Space

For more complex examples, we can define multiple interpretable dimensions:

Dimensions:

1. Royalty (0-1)
2. Gender (0-1)
3. Age (0=young, 1=adult)
4. Animacy (0=object, 1=animate)

Sample Embeddings:

$$\begin{aligned}\phi(\text{"king"}) &= [0.95, 0.90, 0.95, 0.99]\\ \phi(\text{"queen"}) &= [0.95, 0.10, 0.95, 0.99]\\ \phi(\text{"prince"}) &= [0.85, 0.80, 0.30, 0.99]\\ \phi(\text{"car"}) &= [0.10, 0.50, 0.80, 0.01]\\ \phi(\text{"baby"}) &= [0.05, 0.50, 0.10, 0.99]\end{aligned}$$

Verification:

$$\begin{aligned}\cos(\phi(\text{"king"}), \phi(\text{"queen"})) &\approx 0.94 \quad \text{(high - same category)}\\ \cos(\phi(\text{"king"}), \phi(\text{"car"})) &\approx 0.35 \quad \text{(low - different categories)}\end{aligned}$$

1.3 From Examples to Real Embeddings

Example: How Real Models Create Embeddings

Real embedding models like BERT don't use manually designed dimensions. Instead:

Training Process:

1. Model reads billions of sentences
2. Learns that words in similar contexts should have similar vectors
3. Automatically discovers semantic dimensions through neural networks

Real vs. Example Embeddings:

- **Example embeddings**: 2-4 dimensions, human-designed, interpretable
- **Real embeddings**: 384-768+ dimensions, machine-learned, hard to interpret
- **Both preserve**: Semantic relationships through vector geometry

Example of Real Embedding:

$$\phi_{\text{real}}(\text{"king"}) = [0.134, -0.542, 0.218, \ldots, 0.076] \quad \text{(768 dimensions)}$$
$$\phi_{\text{real}}(\text{"queen"}) = [0.128, -0.538, 0.195, \ldots, 0.081]$$
$$\cos(\phi_{\text{real}}(\text{"king"}), \phi_{\text{real}}(\text{"queen"})) \approx 0.89 \quad \text{(still high!)}$$

1.4 Mathematical Foundation for Example Construction

Example: Ensuring Mathematical Consistency

To create valid examples, we ensure they satisfy key properties:
Property 1: Similar words have high cosine similarity

$$\begin{aligned}\cos(\phi(\text{"man"}), \phi(\text{"boy"})) &= \frac{[0.10, 0.85] \cdot [0.08, 0.75]}{\|[0.10, 0.85]\|\|[0.08, 0.75]\|} \\ &= \frac{0.008 + 0.6375}{\sqrt{0.7325} \cdot \sqrt{0.5689}} \approx 0.98\end{aligned}$$

Property 2: Analogies work vectorially

$$\begin{aligned}\phi(\text{"king"}) - \phi(\text{"man"}) + \phi(\text{"woman"}) &= [0.95, 0.90] - [0.10, 0.85] + [0.10, 0.15] \\ &= [0.95, 0.20] \approx \phi(\text{"queen"})\end{aligned}$$

Property 3: Semantic clusters emerge

- Royal cluster: king, queen, prince, princess (high X-values)
- Common male cluster: man, boy (low X, high Y)
- Common female cluster: woman, girl (low X, low Y)

1.5 Creating Your Own Examples

Example: Building a Custom Semantic Space

You can create your own examples for any domain:

Domain: Food

- Dimensions: [Sweetness, Temperature, Healthiness]

$$\begin{aligned}
\phi(\text{"ice cream"}) &= [0.9, 0.1, 0.2] \quad \text{(sweet, cold, unhealthy)} \\
\phi(\text{"salad"}) &= [0.1, 0.3, 0.9] \quad \text{(savory, cool, healthy)} \\
\phi(\text{"soup"}) &= [0.2, 0.9, 0.7] \quad \text{(savory, hot, healthy)} \\
\phi(\text{"cake"}) &= [0.95, 0.5, 0.1] \quad \text{(sweet, warm, unhealthy)}
\end{aligned}$$

Verification:

$$\begin{aligned}
\cos(\phi(\text{"ice cream"}), \phi(\text{"cake"})) &\approx 0.85 \quad \text{(both sweet treats)} \\
\cos(\phi(\text{"ice cream"}), \phi(\text{"salad"})) &\approx 0.25 \quad \text{(very different)}
\end{aligned}$$

2 Introduction to Text Embeddings

2.1 The Fundamental Problem

Example: The Language-Number Gap

Consider trying to teach a computer about animals:

- Human: "cat", "dog", "lion", "elephant"
- Computer needs: Numerical representations
- Solution: $\phi(\text{"cat"}) = [0.8, 0.2, 0.1]$ (pet, small, domestic)
- $\phi(\text{"lion"}) = [0.1, 0.9, 0.0]$ (wild, large, dangerous)

Now the computer can compute similarities mathematically.

Computers operate exclusively on numerical data, while human communication primarily uses natural language. The challenge is to bridge this gap by creating a mapping:

$$f : \mathcal{T} \to \mathbb{R}^d \tag{1}$$

where $\mathcal{T}$ is the set of all possible texts and d is the embedding dimensionality.

2.2 Historical Context

Example: Evolution of Embeddings

One-hot encoding:

- Vocabulary: ["cat", "dog", "bird"]
- $\phi_{\text{one-hot}}(\text{"cat"}) = [1, 0, 0]$
- $\phi_{\text{one-hot}}(\text{"dog"}) = [0, 1, 0]$
- Problem: All words equally distant, no semantics

Modern embeddings:

- $\phi(\text{"cat"}) = [0.8, 0.2, 0.1, \dots]$
- $\phi(\text{"dog"}) = [0.7, 0.3, 0.2, \dots]$
- $\phi(\text{"bird"}) = [0.3, 0.9, 0.0, \dots]$
- Semantic relationships preserved!

Early approaches included:

- **One-hot encoding**: $v_{\text{word}} \in \{0, 1\}^{|V|}$ where V is vocabulary
- **TF-IDF**: Term frequency-inverse document frequency
- **Word2Vec**: Neural network-based embeddings
- **Transformers**: Modern contextual embeddings

3 Why High-Dimensional Embeddings?

3.1 The Need for Multiple Semantic Dimensions

Example: Limitations of Low-Dimensional Spaces

2D Space (Royalty vs Gender):

$$\phi(\text{"king"}) = [0.9, 0.8]$$
$$\phi(\text{"queen"}) = [0.9, 0.2]$$
$$\phi(\text{"car"}) = [0.1, 0.5]$$

Problem: Where to place "computer"? It's not royal, but gender doesn't apply!

Real language requires capturing hundreds of nuanced semantic aspects simultaneously.

3.2 Semantic Dimensions in Real Embeddings

Example: What 768 Dimensions Represent

Each dimension captures a different semantic aspect:

- Dimension 1: Royalty vs commonness
- Dimension 2: Masculinity vs femininity
- Dimension 3: Age (young vs old)
- Dimension 4: Formality level
- Dimension 5: Positive vs negative sentiment
- Dimension 6: Concrete vs abstract
- Dimension 7: Human vs object
- Dimension 8: Size (small vs large)
- ... Dimensions 9-768: Thousands more subtle features

3.3 Mathematical Representation

Example: Real Word Embedding Structure

A 768-dimensional embedding for "king":

$$\begin{aligned}\phi(\text{"king"}) = [&0.134, -0.542, 0.218, 0.076, -0.289, 0.431, 0.152, -0.087,\\ &0.324, 0.198, -0.453, 0.267, 0.089, -0.176, 0.512, \ldots\\ &\ldots, 0.076, -0.234, 0.187, 0.423, -0.159, 0.298]\end{aligned}$$

Each number represents the word's position along that particular semantic dimension.

3.4 Why 768 Specifically?

Example: Trade-offs in Dimension Choice

Too few dimensions (e.g., 50):

- Cannot capture all semantic nuances
- Words collapse into same vectors
- Poor performance on complex tasks

Too many dimensions (e.g., 2048):

- Overfitting to training data
- Computational inefficiency
- Diminishing returns

768 dimensions:

- Enough capacity for complex semantics
- Computationally efficient
- Standard in models like BERT-base

3.5 The Curse of Dimensionality

Example: Distance Behavior in High Dimensions

In high dimensions, distance metrics behave differently:
For random vectors in 768D:

$$\mathbb{E}[\|\mathbf{u} - \mathbf{v}\|_2] \approx \sqrt{2d} \approx 39.2$$
$$\mathbb{E}[\cos(\mathbf{u}, \mathbf{v})] \approx 0$$

But for related words:

$$\cos(\phi(\text{"king"}), \phi(\text{"queen"})) \approx 0.7 - 0.9$$
$$\cos(\phi(\text{"king"}), \phi(\text{"car"})) \approx 0.1 - 0.3$$

Semantic relationships create structure in the high-dimensional space.

3.6 How Dimensions are Learned

Example: Neural Network Weight Matrix

The embedding matrix $W_E \in \mathbb{R}^{V \times 768}$ where:

- V = vocabulary size (e.g., 30,000)
- Each row is a 768D word embedding
- Learned through contextual prediction tasks

Training objective:

$$\max \sum_{i=1}^{N} \log P(w_i | w_{i-1}, w_{i-2}, \ldots, w_{i-k}) \quad (2)$$

The network discovers useful dimensions that help predict word contexts.

3.7 Interpretability Challenge

Example: Dimension Interpretation

Individual dimensions are hard to interpret:

$$\phi(\text{"king"})_1 = 0.134 \quad \text{(what does this mean?)}$$
$$\phi(\text{"queen"})_1 = 0.128$$
$$\phi(\text{"car"})_1 = -0.456$$

But directions matter:

$$\phi(\text{"king"}) - \phi(\text{"queen"}) \approx \text{"gender direction"}$$
$$\phi(\text{"king"}) - \phi(\text{"man"}) \approx \text{"royalty direction"}$$

Semantic meaning emerges from combinations of dimensions.

3.8 Empirical Justification

Example: Performance vs Dimension Size

Experimental results show:

Dimensions	**Semantic Accuracy**	**Speed (sentences/sec)**
128	68%	1200
256	78%	800
512	85%	400
768	88%	250
1024	89%	150

768 provides the best trade-off between accuracy and efficiency.

4 Mathematical Foundations

4.1 Vector Space Model

Example: Simple 3D Word Space

Let's create a 3-dimensional semantic space:

- Dimension 1: Animal (1.0) vs Object (0.0)
- Dimension 2: Size (1.0 = large, 0.0 = small)
- Dimension 3: Domestic (1.0) vs Wild (0.0)

$$\phi(\text{"cat"}) = [0.9, 0.2, 0.8]$$
$$\phi(\text{"dog"}) = [0.9, 0.3, 0.9]$$
$$\phi(\text{"elephant"}) = [0.9, 1.0, 0.3]$$
$$\phi(\text{"car"}) = [0.1, 0.7, 0.6]$$

Now "cat" and "dog" are close, while "car" is distant from all animals.

Given a vocabulary $V = \{w_1, w_2, \ldots, w_n\}$, we seek to find an embedding function:

$$\phi : V \to \mathbb{R}^d \tag{3}$$

such that semantic relationships are preserved:

$$\text{sim}(w_i, w_j) \approx \cos(\phi(w_i), \phi(w_j)) \tag{4}$$

4.2 Similarity Metrics

4.2.1 Cosine Similarity

Example: Calculating Cosine Similarity

Let's compare two word vectors:

$$\begin{aligned}\mathbf{u} &= \phi(\text{"king"}) = [0.9, 0.8]\\ \mathbf{v} &= \phi(\text{"queen"}) = [0.9, 0.2]\end{aligned}$$

Calculate cosine similarity:

$$\begin{aligned}\mathbf{u}\cdot\mathbf{v} &= 0.9\times 0.9 + 0.8\times 0.2 = 0.81 + 0.16 = 0.97\\ \|\mathbf{u}\| &= \sqrt{0.9^2 + 0.8^2} = \sqrt{0.81 + 0.64} = \sqrt{1.45} \approx 1.204\\ \|\mathbf{v}\| &= \sqrt{0.9^2 + 0.2^2} = \sqrt{0.81 + 0.04} = \sqrt{0.85} \approx 0.922\\ \cos(\mathbf{u},\mathbf{v}) &= \frac{0.97}{1.204\times 0.922} \approx \frac{0.97}{1.110} \approx 0.874\end{aligned}$$

High similarity (0.874) confirms semantic relationship.

$$\cos(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \frac{\sum_{i=1}^{d} u_i v_i}{\sqrt{\sum_{i=1}^{d} u_i^2}\sqrt{\sum_{i=1}^{d} v_i^2}} \tag{5}$$

4.2.2 Euclidean Distance

Example: Euclidean Distance Calculation

Using the same vectors:

$$\mathbf{u} = [0.9, 0.8]$$
$$\mathbf{v} = [0.9, 0.2]$$
$$d(\mathbf{u}, \mathbf{v}) = \sqrt{(0.9 - 0.9)^2 + (0.8 - 0.2)^2} = \sqrt{0 + 0.36} = 0.6$$

Compare with dissimilar words:

$$\mathbf{w} = \phi(\text{"car"}) = [0.1, 0.2]$$
$$d(\mathbf{u}, \mathbf{w}) = \sqrt{(0.9 - 0.1)^2 + (0.8 - 0.2)^2} = \sqrt{0.64 + 0.36} = 1.0$$

"king" and "queen" are closer than "king" and "car".

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2} \tag{6}$$

4.3 The Famous Word Analogy

Example: Complete Word Analogy Calculation

Let's verify the famous analogy with 4D vectors:

$$\begin{aligned} \phi(\text{"king"}) &= [0.9, 0.8, 0.1, 0.9] \\ \phi(\text{"queen"}) &= [0.9, 0.2, 0.9, 0.9] \\ \phi(\text{"man"}) &= [0.1, 0.7, 0.1, 0.8] \\ \phi(\text{"woman"}) &= [0.1, 0.3, 0.9, 0.8] \end{aligned}$$

Calculate the analogy:

$$\begin{aligned} \phi(\text{"king"}) - \phi(\text{"man"}) &= [0.8, 0.1, 0.0, 0.1] \\ \phi(\text{"king"}) - \phi(\text{"man"}) + \phi(\text{"woman"}) &= [0.9, 0.4, 0.9, 0.9] \\ \cos([0.9, 0.4, 0.9, 0.9], \phi(\text{"queen"})) &\approx 0.94 \end{aligned}$$

Very high similarity confirms the analogy holds!

The classic example demonstrates vector arithmetic:

$$\phi(\text{king}) - \phi(\text{man}) + \phi(\text{woman}) \approx \phi(\text{queen}) \tag{7}$$

$$\phi(\text{Paris}) - \phi(\text{France}) + \phi(\text{Italy}) \approx \phi(\text{Rome}) \tag{8}$$

5 Implementation Details

5.1 The Embedding Function

Example: Cache Operation in Practice

Initial state: cache = {}
First call: embed_text(”hello”)

- ”hello” not in cache → compute embedding
- $\phi(\text{"hello"}) = [0.1, 0.5, -0.2, \ldots]$
- cache[”hello”] = [0.1, 0.5, -0.2, . . .]
- Return: [0.1, 0.5, -0.2, . . .]

Second call: embed_text(”hello”)

- ”hello” in cache → skip computation
- Return: [0.1, 0.5, -0.2, . . .] (instantaneous)

Performance: 1000 calls with 10 unique texts

- Without cache: 1000 computations
- With cache: 10 computations + 990 lookups
- 100x speedup!

The provided Python code implements an efficient embedding system with caching:

Algorithm 1 Text Embedding with Caching

```
function EMBED_TEXT(text)
    if text ∉ cache then
        v ← encoder.encode(text)
        cache[text] ← v
    end if
    return cache[text]
end function
```

5.2 Mathematical Formulation of the Implementation

Example: Mathematical Cache Operation

Let embedding computation time $t_e = 50$ms, cache lookup $t_c = 0.5$ms. For document processing with texts: [”hello”, ”world”, ”hello”, ”ai”, ”world”]

$$\begin{aligned} \text{Unique texts} &= 3 \quad \text{(hello, world, ai)} \\ \text{Total calls} &= 5 \\ \text{Time without cache} &= 5 \times 50 = 250\text{ms} \\ \text{Time with cache} &= 3 \times 50 + 2 \times 0.5 = 151\text{ms} \\ \text{Speedup} &= \frac{250}{151} \approx 1.66\times \end{aligned}$$

For larger scale: 1000 calls, 100 unique texts:

$$\text{Speedup} = \frac{1000 \times 50}{100 \times 50 + 900 \times 0.5} = \frac{50000}{5000 + 450} \approx 9.2\times$$

Let E be our embedding model, C our cache, and T our input text:

$$\text{embed\_text}(T) = \begin{cases} E(T) & \text{if } T \notin C \\ C[T] & \text{otherwise} \end{cases} \tag{9}$$

The cache update rule:

$$C[T] \leftarrow E(T) \quad \text{when } T \notin C \tag{10}$$

6 How Embeddings are Computed

6.1 Transformer-based Embeddings

Example: Sentence Encoding Process

Encoding the sentence: ”The cat sat on the mat”
Step 1: Tokenization

- Input: ”The cat sat on the mat”
- Tokens: [”The”, ”cat”, ”sat”, ”on”, ”the”, ”mat”]
- Token IDs: [1, 542, 1234, 56, 1, 7890]

Step 2: Forward Pass

$$\mathbf{H} = \text{Transformer}(\mathbf{X})$$
$$\mathbf{H} \in \mathbb{R}^{6\times 768} \quad \text{(6 tokens, 768 dimensions)}$$

Step 3: Pooling

$$\mathbf{v} = \text{mean-pool}(\mathbf{H}) = \frac{1}{6}\sum_{i=1}^{6} \mathbf{h}_i$$
$$\mathbf{v} \in \mathbb{R}^{768} \quad \text{(sentence embedding)}$$

Modern embeddings use transformer architectures:

$$\mathbf{H} = \text{Transformer}(\mathbf{X}) \tag{11}$$

where **X** is the input token sequence and **H** is the hidden state matrix.

6.2 The Encoding Process

Example: Complete Pipeline for "I love AI"

1. **Input**: "I love AI"
2. **Tokenization**: ["I", "love", "AI"] → [100, 205, 3001]
3. **Embedding Lookup**:

$$\mathbf{E}_{\text{I}} = [0.1, 0.2, \dots]$$
$$\mathbf{E}_{\text{love}} = [0.8, 0.1, \dots]$$
$$\mathbf{E}_{\text{AI}} = [0.9, 0.8, \dots]$$

4. **Positional Encoding**: Add position information
5. **Transformer Layers**: 12 layers of self-attention
6. **Output**: Contextualized token representations
7. **Pooling**: Average all token representations
8. **Final**: $\phi(\text{"I love AI"}) = [0.6, 0.37, \dots]$

For a sentence $S = [t_1, t_2, \dots, t_n]$:

1. **Tokenization**: Convert text to tokens
2. **Positional Encoding**: Add position information
3. **Multi-head Attention**: Compute contextual representations
4. **Pooling**: Aggregate token representations

6.2.1 Multi-head Attention

Example: Single Attention Head Calculation

For token "cat" in "The cat sat", with 4-dimensional embeddings:

$$\mathbf{Q}_{\text{cat}} = [1.2, 0.8, -0.5, 0.3]$$
$$\mathbf{K}_{\text{The}} = [0.9, 0.1, 0.2, -0.3]$$
$$\mathbf{K}_{\text{cat}} = [1.1, 0.9, -0.4, 0.2]$$
$$\mathbf{K}_{\text{sat}} = [0.8, 0.7, -0.6, 0.4]$$

Compute attention scores:

$$\text{score}_{\text{The}} = \frac{\mathbf{Q}_{\text{cat}} \cdot \mathbf{K}_{\text{The}}}{\sqrt{4}} = \frac{1.08}{2} = 0.54$$
$$\text{score}_{\text{cat}} = \frac{\mathbf{Q}_{\text{cat}} \cdot \mathbf{K}_{\text{cat}}}{\sqrt{4}} = \frac{2.16}{2} = 1.08$$
$$\text{score}_{\text{sat}} = \frac{\mathbf{Q}_{\text{cat}} \cdot \mathbf{K}_{\text{sat}}}{\sqrt{4}} = \frac{1.56}{2} = 0.78$$

Softmax: [0.24, 0.48, 0.28] → "cat" pays most attention to itself!

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{12}$$
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{13}$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{14}$$

7 Concrete Example with Mathematics

7.1 2D Conceptual Example

Example: Complete 2D Semantic Space

Let's build a comprehensive 2D space:

- X-axis: Royalty (0.0 = common, 1.0 = royal)
- Y-axis: Gender (0.0 = feminine, 1.0 = masculine)

$$
\begin{aligned}
\phi(\text{"king"}) &= [0.95, 0.90] \\
\phi(\text{"queen"}) &= [0.95, 0.10] \\
\phi(\text{"prince"}) &= [0.85, 0.80] \\
\phi(\text{"princess"}) &= [0.85, 0.15] \\
\phi(\text{"man"}) &= [0.10, 0.85] \\
\phi(\text{"woman"}) &= [0.10, 0.15] \\
\phi(\text{"boy"}) &= [0.08, 0.75] \\
\phi(\text{"girl"}) &= [0.08, 0.20]
\end{aligned}
$$

Visualize: Royalty on X, Gender on Y - clear clusters emerge!

Let's define a simplified 2D embedding space:

$$\phi(\text{king}) = [0.9, 0.8] \tag{15}$$
$$\phi(\text{queen}) = [0.9, 0.2] \tag{16}$$
$$\phi(\text{man}) = [0.1, 0.7] \tag{17}$$
$$\phi(\text{woman}) = [0.1, 0.3] \tag{18}$$

7.2 Vector Arithmetic Verification

Example: Multiple Analogies Verification

Analogy 1: Royal gender change

$$\phi(\text{"king"}) - \phi(\text{"queen"}) = [0.00, 0.80] \quad \text{(male}\rightarrow\text{female)}$$
$$\phi(\text{"man"}) - \phi(\text{"woman"}) = [0.00, 0.70] \quad \text{(same direction!)}$$

Analogy 2: Age relationship

$$\phi(\text{"man"}) - \phi(\text{"boy"}) = [0.02, 0.10] \quad \text{(adult}\rightarrow\text{child)}$$
$$\phi(\text{"woman"}) - \phi(\text{"girl"}) = [0.02, -0.05] \quad \text{(similar!)}$$

Analogy 3: Royal to common

$$\phi(\text{"king"}) - \phi(\text{"man"}) = [0.85, 0.05] \quad \text{(royal}\rightarrow\text{common)}$$
$$\phi(\text{"queen"}) - \phi(\text{"woman"}) = [0.85, -0.05] \quad \text{(similar!)}$$

All semantic relationships captured as vector directions!

$$\phi(\text{king}) - \phi(\text{man}) = [0.8, 0.1] \tag{19}$$
$$\phi(\text{queen}) - \phi(\text{woman}) = [0.8, -0.1] \tag{20}$$
$$\cos([0.8, 0.1], [0.8, -0.1]) = \frac{0.64 - 0.01}{\sqrt{0.65}\sqrt{0.65}} \approx 0.97 \tag{21}$$

High cosine similarity confirms the relationship is captured.

8 Real-world Implementation Mathematics

8.1 The Encoder Function

Example: BERT-base Embedding Dimensions

For BERT-base model:

- Vocabulary size: 30,522 tokens
- Hidden dimension: 768
- Layers: 12
- Attention heads: 12
- Total parameters: 110 million

  Single sentence processing:

$$\begin{aligned} \text{Input: } & \text{"The quick brown fox"} \\ \text{Tokens: } & [\text{"The"}, \text{"quick"}, \text{"brown"}, \text{"fox"}] \\ \text{Token embeddings: } & \mathbb{R}^{4\times768} \\ \text{Output: } & \mathbb{R}^{4\times768} \text{ contextual embeddings} \\ \text{Sentence embedding: } & \mathbb{R}^{768} \text{ (mean pooled)} \end{aligned}$$

The `encoder.encode()` function typically implements:

$$E(T) = \text{Pool}(\text{Transformer}(\text{Tokenize}(T))) \tag{22}$$

8.2 Pooling Strategies

Example: Comparing Pooling Strategies

Sentence: "AI is amazing" with token embeddings:

$$\begin{aligned}\mathbf{h}_1 &= [0.1, 0.2, 0.3] \quad \text{(AI)}\\ \mathbf{h}_2 &= [0.4, 0.1, 0.5] \quad \text{(is)}\\ \mathbf{h}_3 &= [0.9, 0.8, 0.7] \quad \text{(amazing)}\end{aligned}$$

Mean Pooling:

$$\begin{aligned}\mathbf{v} &= \frac{1}{3}([0.1, 0.2, 0.3] + [0.4, 0.1, 0.5] + [0.9, 0.8, 0.7])\\ &= \frac{1}{3}[1.4, 1.1, 1.5] = [0.467, 0.367, 0.5]\end{aligned}$$

Max Pooling:

$$\begin{aligned}\mathbf{v} &= [\max(0.1, 0.4, 0.9), \max(0.2, 0.1, 0.8), \max(0.3, 0.5, 0.7)]\\ &= [0.9, 0.8, 0.7]\end{aligned}$$

CLS Token: Use first token's embedding: $[0.1, 0.2, 0.3]$

- **Mean Pooling**: $\mathbf{v} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{h}_i$
- **CLS Token**: $\mathbf{v} = \mathbf{h}_{\text{CLS}}$
- **Max Pooling**: $v_j = \max_i h_{ij}$

9 Performance Optimization

9.1 Cache Efficiency

Example: Real-world Cache Performance

Scenario: Chatbot processing user messages

- Embedding computation: $t_e = 20\text{ms}$
- Cache lookup: $t_c = 0.1\text{ms}$
- User messages: 1000 total, 200 unique phrases

Without cache:

$$\text{Total time} = 1000 \times 20\text{ms} = 20,000\text{ms} = 20 \text{ seconds}$$

With cache:

$$\begin{aligned}\text{Total time} &= 200 \times 20\text{ms} + 800 \times 0.1\text{ms} \\ &= 4000\text{ms} + 80\text{ms} = 4080\text{ms} \approx 4 \text{ seconds}\end{aligned}$$

Speedup: $20s/4s = 5\times$ faster!
Memory usage (768-dim float32):

$$\begin{aligned}200 \text{ embeddings} &= 200 \times 768 \times 4\text{bytes} \\ &= 614,400\text{bytes} \approx 600\text{KB} \quad \text{(tiny!)}\end{aligned}$$

Let t_e be embedding computation time and t_c be cache lookup time. The speedup factor is:

$$S = \frac{t_e}{t_c} \approx 100 - 1000\times \tag{23}$$

For n unique texts and m total calls ($m \gg n$):

$$\text{Total time} = n \cdot t_e + (m - n) \cdot t_c \tag{24}$$

9.2 Memory Complexity

Example: Cache Memory Calculation

System specifications:

- Embedding dimension: $d = 768$
- Float size: 4 bytes
- Average text length: 50 characters (2 bytes per char UTF-16)
- Cache entries: 10,000

Memory calculation:

$$\text{Text storage} = 10,000 \times 50 \times 2 = 1,000,000 \text{ bytes} \approx 1\text{MB}$$
$$\text{Embedding storage} = 10,000 \times 768 \times 4 = 30,720,000 \text{ bytes} \approx 30\text{MB}$$

Total cache memory $\approx$ 31MB

Comparison: Modern systems have 16GB+ RAM, so 31MB is only 0.2% of memory!
Scaling: To store 1 million embeddings:

$$\text{Memory required} \approx 3.1\text{GB} \quad \text{Still manageable!}$$

Cache memory usage:

$$M = \sum_{T \in C} (|T| + d \cdot \text{sizeof(float)}) \tag{25}$$

10 Applications and Use Cases

10.1 Semantic Search

Example: Document Search System

Query: ”machine learning tutorials”
Document database:

1. ”Introduction to neural networks”
2. ”Python programming guide”
3. ”Deep learning course materials”
4. ”Cooking recipes for beginners”
5. ”ML tutorial for beginners”

Embedding similarities:

$$\begin{aligned}
\cos(\phi(\text{query}), \phi(\text{doc1})) &= 0.85 \\
\cos(\phi(\text{query}), \phi(\text{doc2})) &= 0.45 \\
\cos(\phi(\text{query}), \phi(\text{doc3})) &= 0.92 \\
\cos(\phi(\text{query}), \phi(\text{doc4})) &= 0.12 \\
\cos(\phi(\text{query}), \phi(\text{doc5})) &= 0.88
\end{aligned}$$

Search results ranking:

1. ”Deep learning course materials” (0.92)
2. ”ML tutorial for beginners” (0.88)
3. ”Introduction to neural networks” (0.85)
4. ”Python programming guide” (0.45)
5. ”Cooking recipes” (0.12)

Semantic search finds relevant documents even without keyword matches!

Given query q and documents $D = \{d_1, d_2, \ldots, d_k\}$:

$$\text{score}(q, d_i) = \cos(\phi(q), \phi(d_i)) \tag{26}$$

10.2 Text Classification

Example: Sentiment Analysis

Task: Classify movie reviews as positive/negative
Training data:

- Positive: "Great movie with amazing acting" → label 1
- Negative: "Terrible plot and bad acting" → label 0

Model:

$$\mathbf{v} = \phi(\text{review}) \in \mathbb{R}^{768}$$
$$\mathbf{z} = W\mathbf{v} + b \quad (W \in \mathbb{R}^{2\times 768}, b \in \mathbb{R}^2)$$
$$\hat{y} = \text{softmax}(\mathbf{z}) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Prediction for new review:

$$\phi(\text{"Loved the characters and story"}) = [0.8, -0.2, \ldots]$$
$$\hat{y} = [0.85, 0.15] \quad \text{(85\% positive, 15\% negative)}$$

$$\hat{y} = \text{softmax}(W\phi(T) + b) \tag{27}$$

10.3 Clustering

Example: News Article Clustering

Articles to cluster:

1. "Stock market reaches all-time high"
2. "Basketball team wins championship"
3. "New AI model breaks records"
4. "Football league finals this weekend"
5. "Tech company earnings exceed expectations"
6. "Baseball season opener results"

Clustering result:

- **Cluster 1 (Sports)**: 2, 4, 6
- **Cluster 2 (Technology)**: 3, 5
- **Cluster 3 (Finance)**: 1

Cluster centers:

$$\begin{aligned}\mu_{\text{sports}} &= \text{mean}(\phi(\text{article 2}), \phi(\text{article 4}), \phi(\text{article 6})) \\ \mu_{\text{tech}} &= \text{mean}(\phi(\text{article 3}), \phi(\text{article 5})) \\ \mu_{\text{finance}} &= \phi(\text{article 1})\end{aligned}$$

Articles automatically grouped by semantic content!

Group texts based on embedding proximity using algorithms like K-means:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{k} \sum_{T \in C_i} \|\phi(T) - \mu_i\|^2 \tag{28}$$

11 Advanced Mathematical Concepts

11.1 Geometric Interpretation

Example: Semantic Geometry in Action

Consider our 2D royalty-gender space:

- **Distance**:

$$d(\text{"king"}, \text{"queen"}) = 0.8 \quad \text{(small - same category)}$$
$$d(\text{"king"}, \text{"car"}) = 1.5 \quad \text{(large - different categories)}$$

- **Direction**:

$$\phi(\text{"king"}) - \phi(\text{"queen"}) = [0.0, 0.8] \quad \text{(gender axis)}$$
$$\phi(\text{"king"}) - \phi(\text{"man"}) = [0.85, 0.05] \quad \text{(royalty axis)}$$

- **Clusters**:
  - Royal cluster: king, queen, prince, princess
  - Common male cluster: man, boy
  - Common female cluster: woman, girl
  - Objects cluster: car, house, book (not shown)

The vector space becomes a "semantic map" of concepts!

Embeddings create a semantic geometry where:

- Distance $\leftrightarrow$ Semantic dissimilarity
- Direction $\leftrightarrow$ Semantic relationships
- Clusters $\leftrightarrow$ Semantic categories

11.2 Manifold Hypothesis

Example: Understanding the Manifold

High-dimensional space: $\mathbb{R}^{768}$ (BERT embeddings)
Actual data manifold: Much lower intrinsic dimension

- All English sentences lie on a complex surface
- This surface has much lower dimension than 768
- The manifold captures grammatical and semantic rules

Analogy: Think of a spiral in 3D space:

- Ambient space: 3 dimensions
- Actual spiral: 1-dimensional curve
- Similarly, text embeddings lie on low-dimensional surfaces in high-dimensional space

Implication: We can compress embeddings without losing much information!

Text embeddings typically lie on a low-dimensional manifold within $\mathbb{R}^d$:

$$\mathcal{M} \subset \mathbb{R}^d \quad \text{where} \quad \dim(\mathcal{M}) \ll d \tag{29}$$

12 Conclusion

Example: Complete System Overview

Building a semantic search engine:

1. **Document processing**: Convert all documents to embeddings using our cached `embed_text()` function
2. **Query handling**: Convert user query to embedding (cached)
3. **Similarity search**: Compute cosine similarities between query and all document embeddings
4. **Ranking**: Return top-K most similar documents

   Performance:

   - 1 million documents $\rightarrow$ 1 million embeddings (4GB)
   - Query processing: 20ms (including cache benefits)
   - Scalable to web-scale applications!

   Mathematical elegance: Pure linear algebra operations powering understanding of human language!

Text embeddings provide a powerful mathematical framework for representing semantic information in a computationally tractable form. The combination of deep learning architectures with efficient caching mechanisms enables practical applications across natural language processing.

The key insight is that semantic relationships can be encoded as geometric relationships in high-dimensional vector spaces, enabling mathematical operations on concepts and meanings.