t-SNE: Visualizing High-Dimensional Sentence Embeddings

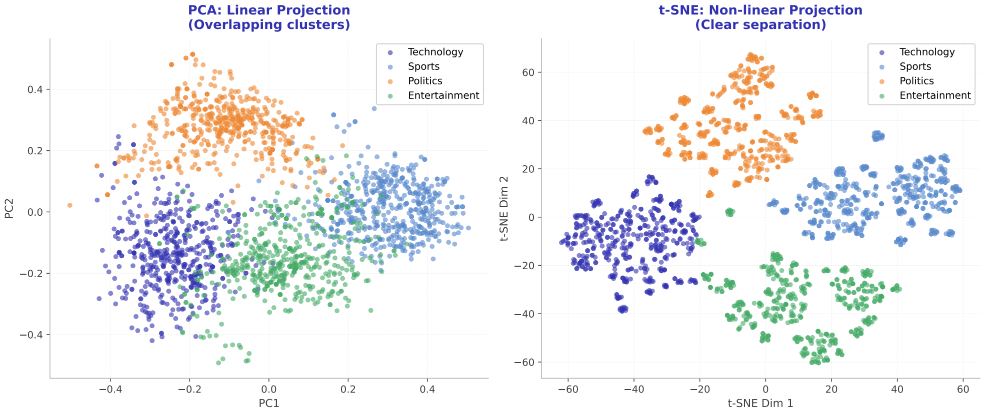
Natural Language Processing

October 3, 2025

From 384 Dimensions to 2D Visualization

Understanding how we visualize meaning captured in numbers

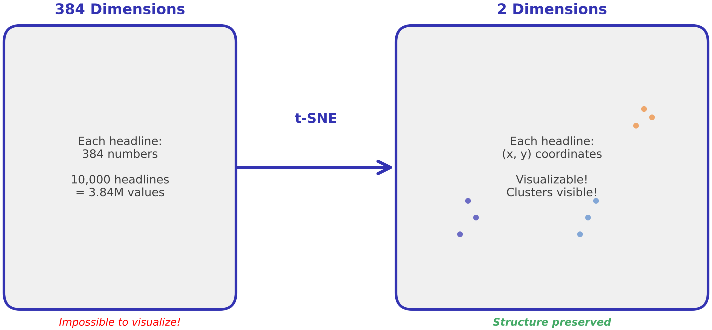
Same Embeddings, Dramatically Different Insights



Key Insight: Non-linear methods reveal structure that linear methods miss

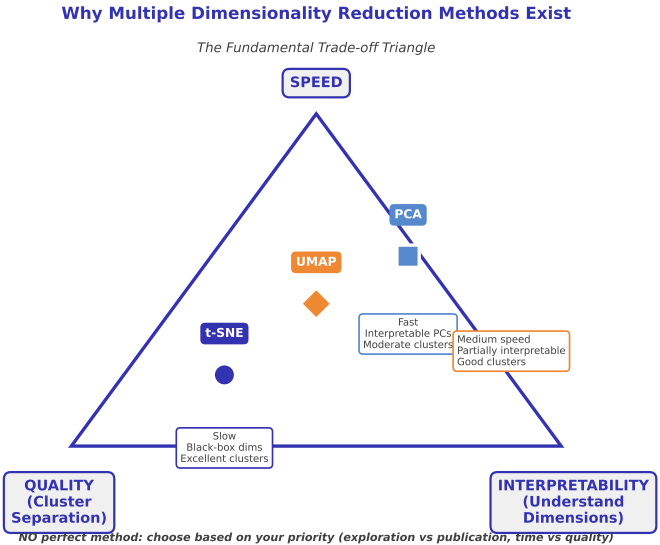
The choice of visualization method dramatically affects what patterns we can discover

The Fundamental Challenge and Trade-offs



The Problem: 384 dimensions cannot be visualized directly

Dimensionality reduction is essential – but which method to choose?


Why Multiple Dimensionality Reduction Methods Exist
The Fundamental Trade-off Triangle
SPEED
PCA
UMAP
t-SNE
Fast
Interpretable PCs
Moderate clusters
Medium speed
Partially interpretable
Good clusters
Slow
Black-box dims
Excellent clusters
QUALITY (Cluster Separation)
INTERPRETABILITY (Understand Dimensions)
NO perfect method: choose based on your priority (exploration vs publication, time vs quality)

Comparing Approaches: Understanding the Trade-offs

Linear Methods (PCA)

Strengths:

- Very fast (seconds for 10,000 points)
- Deterministic (same result every time)
- Interpretable components (axes have meaning)
- Scalable to millions of points
- Preserves global distances

Weaknesses:

- Only captures linear relationships
- May miss complex patterns
- Clusters often overlap
- Moderate cluster separation

Best For:

- Quick exploration
- Large datasets
- When you need interpretable dimensions
- Real-time applications

Non-linear Methods (t-SNE)

Strengths:

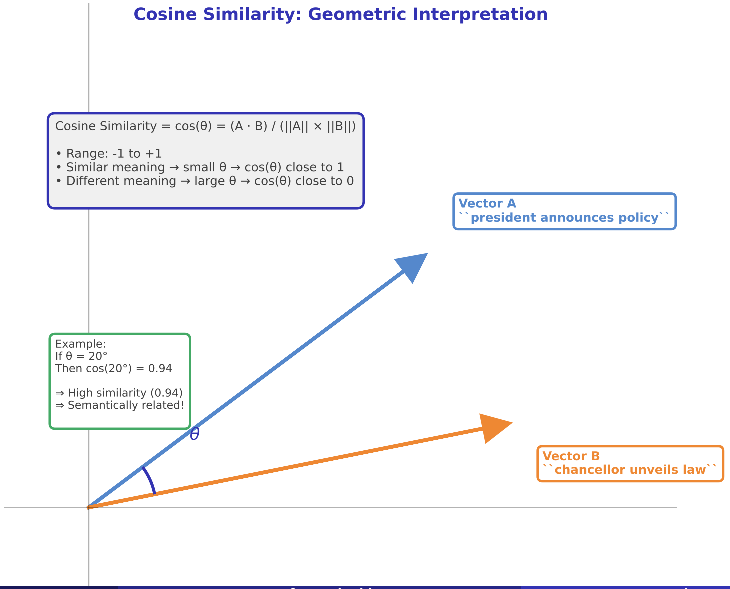
- Captures non-linear patterns
- Excellent cluster separation
- Reveals hidden semantic structure
- Beautiful publication visuals
- Preserves local neighborhoods

Weaknesses:

- Slower (minutes for 2,000 points)
- Stochastic (slight variations per run)
- Black-box dimensions (no interpretation)
- Global distances not meaningful
- Limited scalability

Best For:

- Final visualizations
- Discovering clusters
- Publication figures


Cosine Similarity: Geometric Interpretation
Cosine Similarity = cos(θ) = (A · B) / (||A|| × ||B||)
• Range: -1 to +1
• Similar meaning → small θ → cos(θ) close to 1
• Different meaning → large θ → cos(θ) close to 0
Vector A
``president announces policy``
Example:
If θ = 20°
Then cos(20°) = 0.94
⇒ High similarity (0.94)
⇒ Semantically related!
θ
Vector B
``chancellor unveils law``

High-Dimensional Similarity

Process:

1. Compute all pairwise similarities
2. Each headline: 384D vector
3. Cosine similarity: angle between vectors
4. Similar headlines: high similarity (0.8-1.0)
5. Dissimilar headlines: low similarity (0.0-0.4)

Example:

- A: "president announces policy"
- B: "chancellor unveils law"
- Similarity: 0.89 (very similar!)
- Share semantic meaning

Challenge:

- 10,000 headlines = 50 million pairs
- Cannot visualize 384D space
- Need to reduce to 2D
- But preserve similarity structure

Neighborhoods

Concept:

- Each point has "neighbors" (similar points)
- Neighborhood = K most similar points
- In 384D: neighbors are semantically related
- Goal: Keep neighbors together in 2D

PCA Approach:

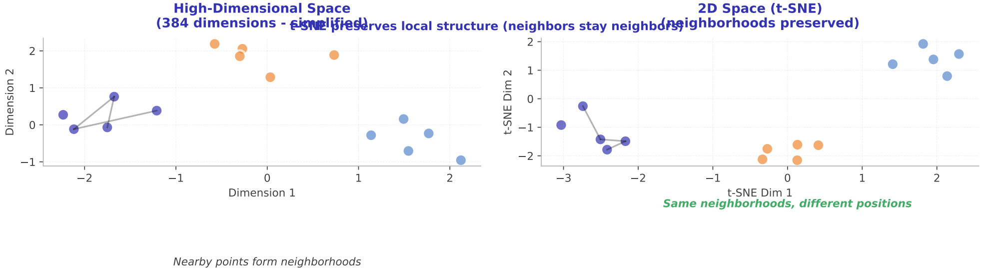
- Find directions of maximum variance
- Project onto these directions
- Linear transformation
- Approximately preserves neighborhoods

t-SNE Approach:

- Convert similarities to probabilities
- Optimize 2D layout to match probabilities
- Non-linear transformation
- Strongly preserves neighborhoods

Key Difference:

t-SNE Core Concept: Neighborhood Preservation



Key Insight: If two points are close in 384D, t-SNE keeps them close in 2D

Local structure (neighborhoods) matters more than global distances in t-SNE

How t-SNE Works: The Algorithm

The Three-Step Process

Step 1: Measure High-D Similarities

- For each pair (i, j) in 384D
- Compute probability p_{ij} they are neighbors
- Uses Gaussian distribution
- Nearby points: high p
- Distant points: low p
- Controlled by perplexity parameter

Step 2: Random 2D Initialization

- Place all points randomly in 2D
- This is our starting configuration
- Pure chaos initially

Step 3: Optimize Layout (1000 iterations)

- Compute 2D similarities q_{ij} (t-distribution)
- Measure difference: $\text{KL}(P||Q)$
- KL divergence = how different are the distributions?
- Move points via gradient descent

Concrete Example

Given: Two headlines

- A: "president announces policy"
- B: "chancellor unveils law"
- 384D embeddings: $\vec{a}$, $\vec{b}$

Step 1: High-D Similarity

$$
\begin{aligned}
d &= ||\vec{a} - \vec{b}|| = 0.35 \\
p_{AB} &= \exp(-d^2/2\sigma^2)/Z \\
&= 0.89 \text{ (very similar!)}
\end{aligned}
$$

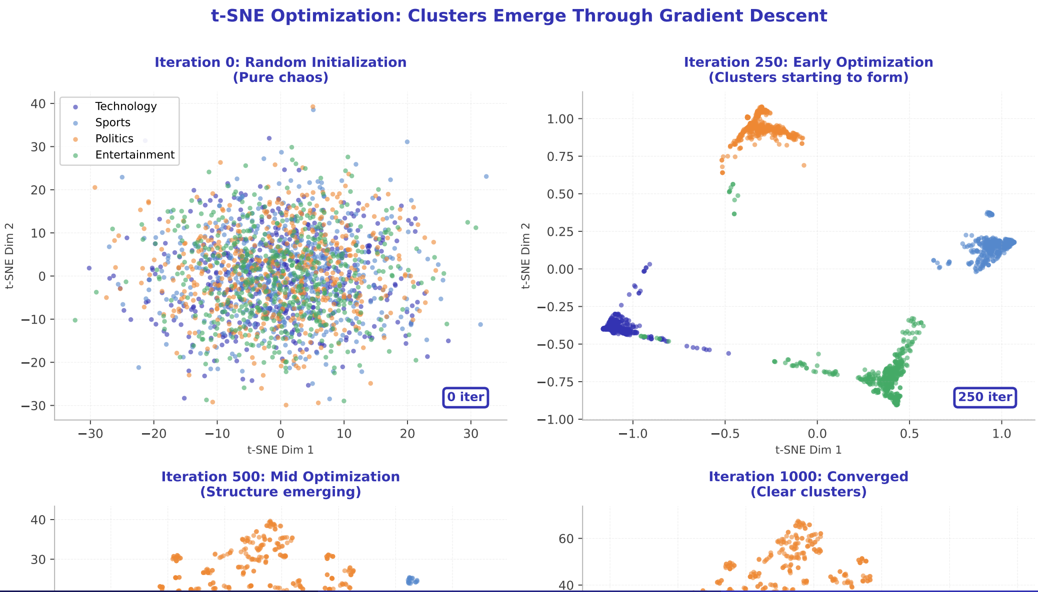
Step 2: Random Init

$$
\begin{aligned}
\vec{a}_{2D} &= (0.2, 0.5) \text{ (random)} \\
\vec{b}_{2D} &= (-0.3, 0.1) \text{ (random)} \\
d_{2D} &= 0.64 \text{ (far apart!)}
\end{aligned}
$$

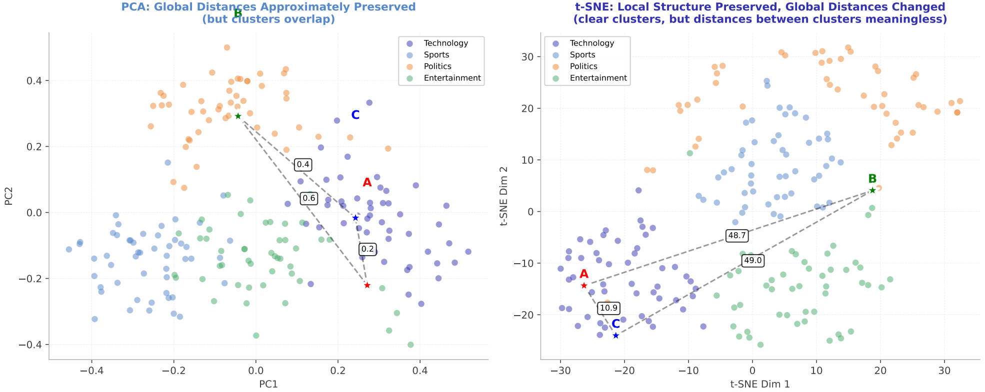
Step 3: Optimize

- Current 2D similarity: $q_{AB} = 0.12$ (low)
- Want: $q_{AB} \approx p_{AB} = 0.89$ (high)

Watching t-SNE Converge: Clusters Emerge from Chaos



What t-SNE Preserves vs What It Sacrifices

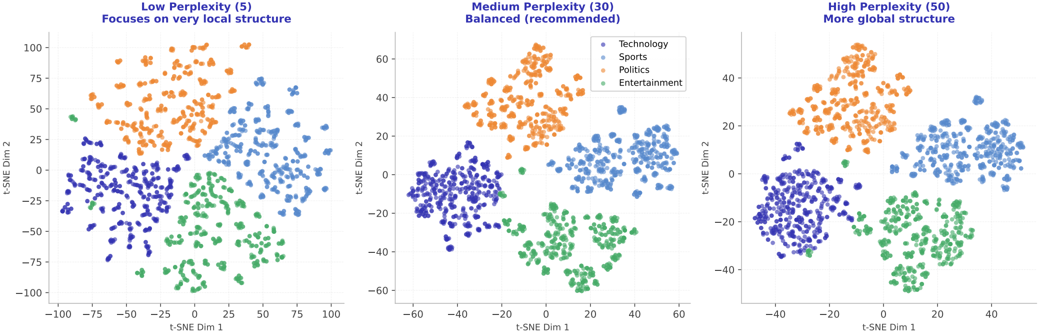


Notice: Same points A, B, C have DIFFERENT relative distances - t-SNE optimizes for local neighborhoods, not global geometry

Key Insight: t-SNE optimizes for local neighborhoods – global distances between clusters are NOT meaningful

Never interpret the distance between clusters in t-SNE – only within-cluster structure matters

Parameter Sensitivity: Perplexity Controls Local vs Global



Key Insight: Perplexity balances local vs global structure (30 is recommended for most datasets)

Too low: overfits noise; too high: misses fine structure; default 30 works well

Our Implementation: Code and Parameters

Code from generate_charts.py:

```python
from sklearn.manifold import TSNE
import numpy as np

# Use subset for computational efficiency
n_samples = 2000
np.random.seed(42)
sample_indices = np.random.choice(len(embeddings), n_samples, replace=False)

# Initialize t-SNE with optimized parameters
tsne = TSNE(
    n_components=2,       # Target: 2D visualization
    random_state=42,      # Reproducibility (fix random seed)
    perplexity=30,        # Balance: not too local, not too global
    max_iter=1000         # Iterations: ensure full convergence
)

# Apply transformation: 384D -> 2D
embeddings_2d = tsne.fit_transform(embeddings[sample_indices])
# Result: (2000, 2) array with (x, y) coordinates
```

Parameter Choices Explained:

- **n_components=2**: We want 2D plots
- **perplexity=30**: Recommended (range: 5-50)
- **max_iter=1000**: Ensure convergence
- **random_state=42**: Reproducibility

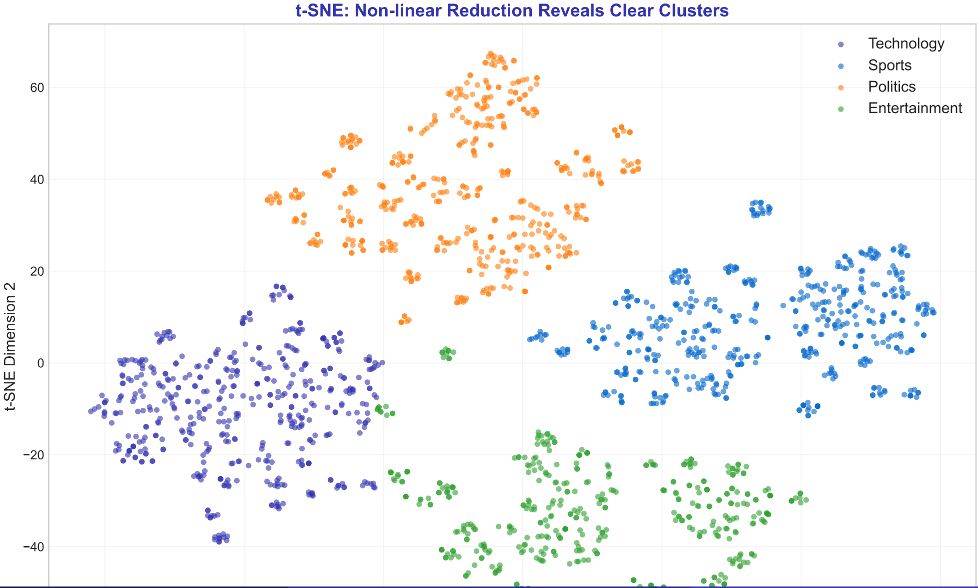
Why 2,000 Samples?:

- Computational efficiency
- Still representative of structure
- Faster iteration during exploration
- Can increase for final figures

Computational Complexity:

Best Practices:

Results: t-SNE Reveals Clear Semantic Clusters



Interpreting Results and Practical Applications

What We Observe

Visual Patterns:

- Four distinct clusters
- Clear separation between categories
- Some meaningful overlap at boundaries
- Clusters discovered without supervision

Quantitative Validation:

- K-means clustering: 97%+ accuracy
- Within-category similarity: 0.62 avg
- Between-category similarity: 0.46 avg
- Clear separation in embedding space

What This Tells Us:

1. Embeddings capture semantic structure
2. Model (all-MiniLM-L6-v2) is high quality
3. Categories are semantically distinct
4. Similar meanings cluster naturally

NLP Applications

Model Development:

- Verify embeddings capture meaning
- Compare different embedding models
- Visualize attention patterns
- Diagnose model failures

Data Analysis:

- Discover themes in large corpora
- Find outliers and mislabeled data
- Identify natural groupings
- Track semantic drift over time

Research Uses:

- Publication-quality visualizations
- Cross-lingual embedding comparison
- Study semantic spaces
- Present findings visually

Example Success:

Decision Guide: When to Use t-SNE

When to Use t-SNE

Ideal Scenarios:

- Goal: Visualize clusters and patterns
- Data size: $<$10,000 points
- Priority: Quality over speed
- Purpose: Final visualization, publication
- Need: Discover hidden structure
- Context: Have 2-5 minutes for computation
- Audience: Need visual proof

Success Criteria:

- Clear cluster separation desired
- Local structure more important than global
- Beautiful visuals matter
- Willing to try multiple parameters

When NOT to Use t-SNE

Avoid If:

- Large data: $>$10,000 points (too slow)

Common Pitfalls and Solutions

Pitfall 1: Interpreting Between-Cluster Distances

- Wrong: "Cluster A is 2x farther from B than C"
- Right: "Points within clusters are similar"
- Solution: Only interpret local structure

Pitfall 2: Expecting Identical Runs

- Problem: Stochastic initialization
- Solution: Always set `random_state=42`
- Result: Reproducible visualizations

Pitfall 3: Wrong Perplexity

- Too low (5): Over-fragments data
- Too high (100): Loses fine structure
- Solution: Try 5, 30, 50; pick best

Pitfall 4: Insufficient Iterations

- 250 iterations: Often insufficient
- 1000 iterations: Recommended
- Solution: Always use max_iter=1000