Neural Language Models

Week 2: Word Embeddings and Word2Vec

NLP Course 2025

Professional Template Edition

September 29, 2025

Learning Path: From discrete word IDs to continuous semantic vectors. Master how neural networks learn word meaning through context, leading to the Word2Vec revolution that powers modern NLP.

Part 1: Introduction & Motivation

Interactive: Word Association Game

Fill in the blank - What word naturally comes next?

1. The cat sat on the ____________________

Interactive: Word Association Game

Fill in the blank - What word naturally comes next?

1. The cat sat on the ________________ → mat, floor, chair (physical objects)

2. I drink my coffee with milk and ________________

Interactive: Word Association Game

Fill in the blank - What word naturally comes next?

1. The cat sat on the ____________________ $\rightarrow$ mat, floor, chair (physical objects)
2. I drink my coffee with milk and ____________________ $\rightarrow$ sugar, cream, honey (additives)
3. The capital of France is ____________________

Interactive: Word Association Game

Fill in the blank - What word naturally comes next?

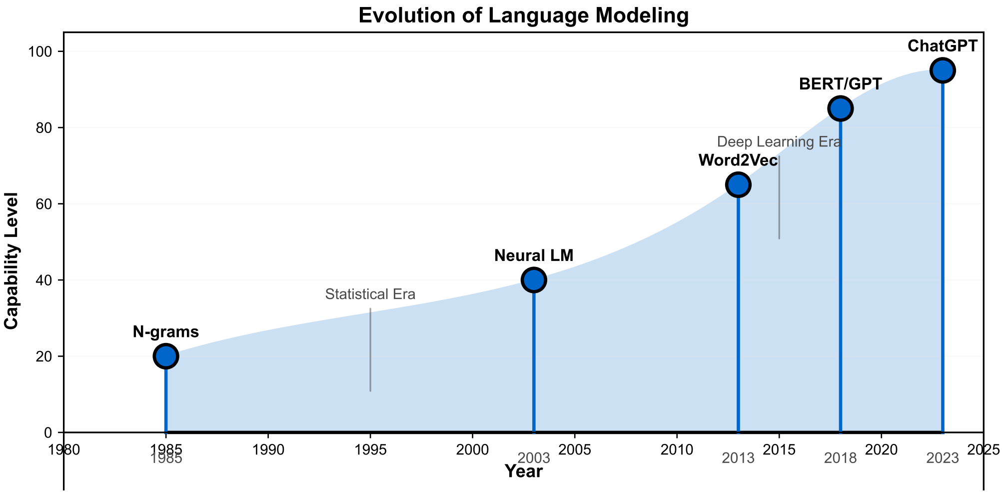
1. The cat sat on the ________________ $\rightarrow$ mat, floor, chair (physical objects)
2. I drink my coffee with milk and ________________ $\rightarrow$ sugar, cream, honey (additives)
3. The capital of France is ________________ $\rightarrow$ Paris (factual knowledge)
4. She was happy but also felt ________________

Interactive: Word Association Game

Fill in the blank - What word naturally comes next?

1. The cat sat on the ________________ $\rightarrow$ mat, floor, chair (physical objects)
2. I drink my coffee with milk and ________________ $\rightarrow$ sugar, cream, honey (additives)
3. The capital of France is ________________ $\rightarrow$ Paris (factual knowledge)
4. She was happy but also felt ________________ $\rightarrow$ sad, anxious, confused (emotions)

Humans predict words using semantic understanding - how can computers learn this?



Four Major Eras in Next-Word Prediction:

The Fundamental Problem

Traditional Approach: One-Hot Encoding

- Words as discrete IDs
- Vocabulary size: 10,000 words
- "cat" = [0,0,1,0,...,0] (position 3)
- "dog" = [0,0,0,0,1,...,0] (position 5)

Problems:

- No notion of similarity
- distance(cat, dog) = distance(cat, democracy)
- Can't generalize knowledge
- Huge, sparse vectors

Solution: Dense Embeddings

- Words as dense vectors
- Dimension: 100-300 (not 10,000!)
- "cat" = [0.2, -0.4, 0.7, ...]
- "dog" = [0.3, -0.3, 0.8, ...]

Benefits:

- Similar words have similar vectors
- distance(cat, dog) ¡ distance(cat, democracy)
- Knowledge transfers between similar words
- Compact, meaningful representation

Key Insight: Learn representations where geometric distance = semantic distance

Real-World Impact

Search Engines

- Semantic search
- Query understanding
- "car" finds "automobile"
- Intent matching

Used by:

- Google Search
- Bing
- DuckDuckGo

Recommendations

- Content similarity
- User preferences
- Cross-lingual matching
- Cold-start solutions

Used by:

- Netflix
- Spotify
- Amazon

Language AI

- Machine translation
- Sentiment analysis
- Chatbots
- Foundation for LLMs

Used by:

- ChatGPT
- Google Translate
- Grammarly

Market Impact: Word embeddings power $100B+ in NLP applications worldwide

Word2Vec papers cited 50,000+ times - one of the most influential ML innovations

Part 2: Core Concepts

"You shall know a word by the company it keeps"

- J.R. Firth (1957)

Example Context Windows:

- The **cat** sat on the mat
- The **dog** sat on the floor
- A **cat** chased the mouse
- A **dog** chased the ball

Shared contexts:

- Both appear after "The" and "A"
- Both appear before "sat", "chased"
- Both are subjects of similar actions

Mathematical Formulation:

Context window of size 2:

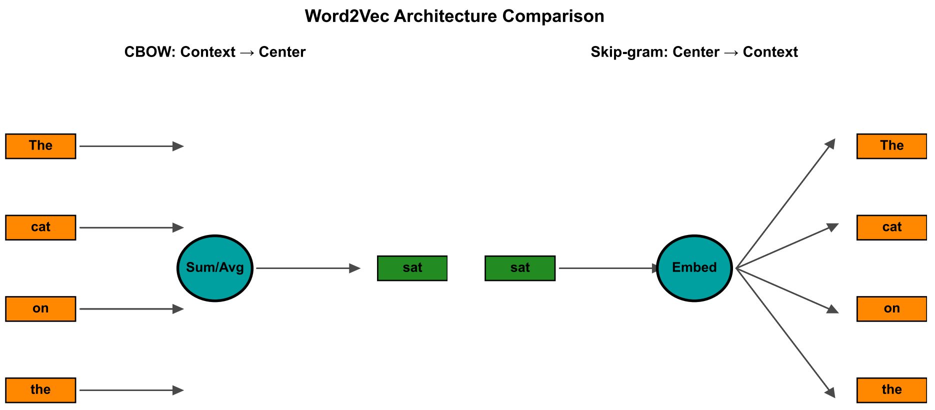
$$\text{context}(\text{cat}) = \{\text{The, sat, on, the}\}$$
$$\text{context}(\text{dog}) = \{\text{The, sat, on, the}\}$$

Key Insight:

- Similar contexts $\Rightarrow$ Similar meanings
- Learn vectors to predict context
- Vectors capture semantic similarity

This simple idea - words with similar contexts have similar meanings - drives all embeddings

Word2Vec: Two Revolutionary Architectures



CBOW (Continuous Bag-of-Words)

- Predict center word from context

Skip-gram

- Predict context from center word

Skip-gram Objective Function:

Maximize the probability of context words given center word:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{t+j} \mid w_t)$$

Probability Calculation using Softmax:

$$P(w_O \mid w_I) = \frac{\exp(v_{w_O}^T \cdot v_{w_I})}{\sum_{w=1}^{V} \exp(v_w^T \cdot v_{w_I})}$$

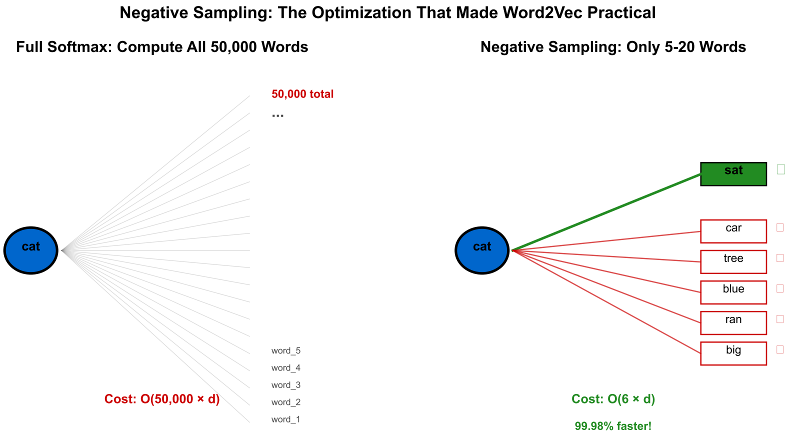
Where:

- v_{w_I}: Input vector for center word
- v_{w_O}: Output vector for context word
- V: Vocabulary size
- c: Context window size

Problem: Denominator sums over entire vocabulary (expensive!)

Computing softmax over 50,000 words for every training example is computationally prohibitive

Negative Sampling: Making Training Feasible



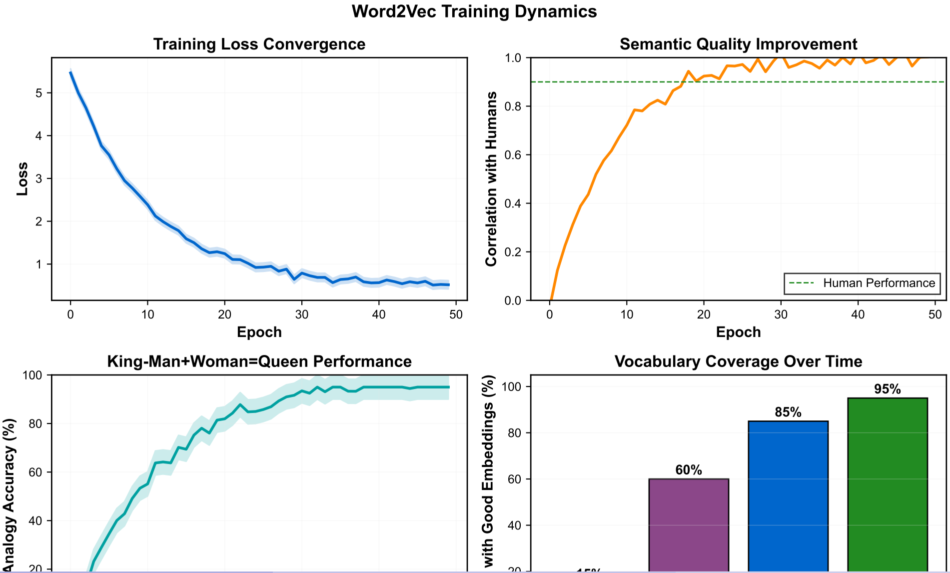
Convert to Binary Classification:

Instead of softmax over all words:

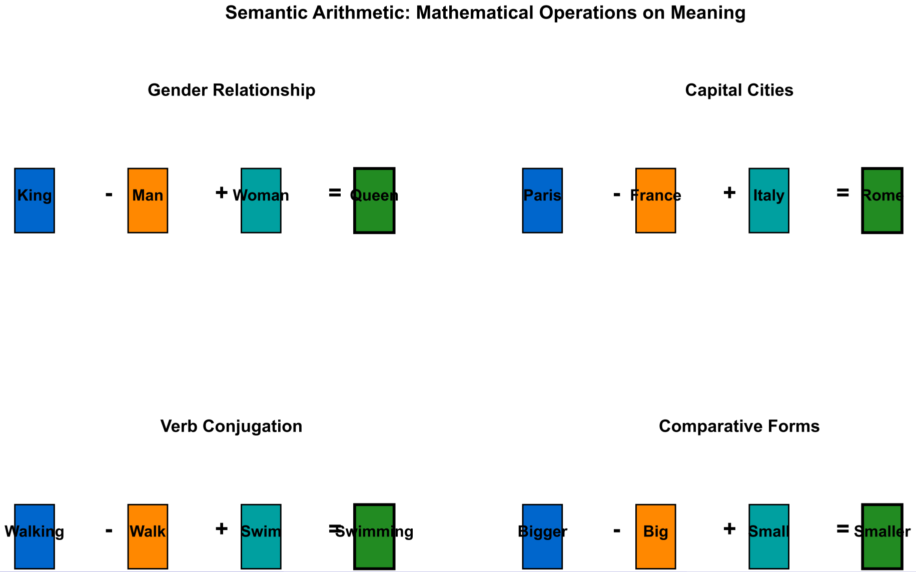
New Objective:

- Positive sample: (cat, sat) → 1

Part 3: Training & Solutions



Word2Vec Training Dynamics
Training Loss Convergence
Loss
Epoch
Semantic Quality Improvement
Correlation with Humans
Epoch
Human Performance
King-Man+Woman=Queen Performance
Analogy Accuracy (%)
Vocabulary Coverage Over Time
with Good Embeddings (%)
60%
85%
95%


Semantic Arithmetic: Mathematical Operations on Meaning
Gender Relationship
King - Man + Woman = Queen
Capital Cities
Paris - France + Italy = Rome
Verb Conjugation
Walking - Walk + Swim = Swimming
Comparative Forms
Bigger - Big + Small = Smaller

Evaluating Word Embeddings

Intrinsic Evaluation

- Word similarity tasks
- Analogy completion
- Clustering quality

Benchmarks:

- WordSim-353
- Google Analogy Test
- SimLex-999

Metrics:

- Spearman correlation
- Accuracy@1, @5
- Silhouette score

Extrinsic Evaluation

- Downstream task performance
- NER improvement
- Sentiment accuracy

Tasks:

- Text classification
- Machine translation
- Question answering

Metrics:

- F1 score improvement
- BLEU score gain
- Task-specific metrics

Visualization

- t-SNE projections
- PCA analysis
- Nearest neighbors

Qualitative:

- Semantic coherence
- Cluster separation
- Outlier detection

Tools:

- TensorBoard
- Embedding Projector
- Custom visualizations

Best Practice: Combine all three - numbers alone don't tell the whole story

Good embeddings show 0.6+ correlation on similarity tasks and 3-5% improvement on downstream tasks

Limitations and Advanced Solutions

Fundamental Limitations:

- Out-of-vocabulary words
  $\rightarrow$ FastText with subword units
- Single vector per word
  $\rightarrow$ Contextual embeddings (ELMo, BERT)
- No word order information
  $\rightarrow$ Position encodings
- Bias in training data
  $\rightarrow$ Debiasing techniques
- Fixed after training
  $\rightarrow$ Fine-tunable embeddings

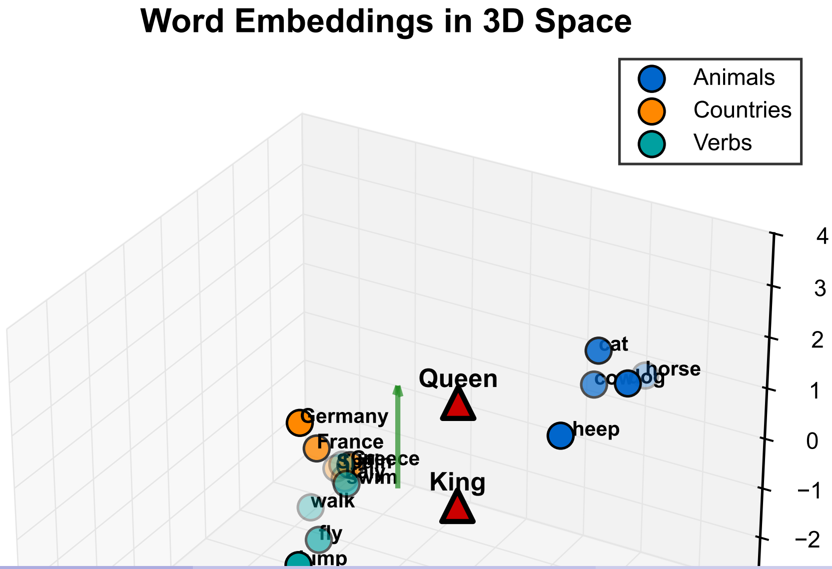
Advanced Techniques:

- **GloVe (2014):**
  Combines global statistics + local context
- **FastText (2016):**
  Character n-grams for OOV handling
- **ELMo (2018):**
  Context-dependent embeddings
- **BERT (2018):**
  Bidirectional contextual representations
- **GPT (2018+):**
  Autoregressive language modeling

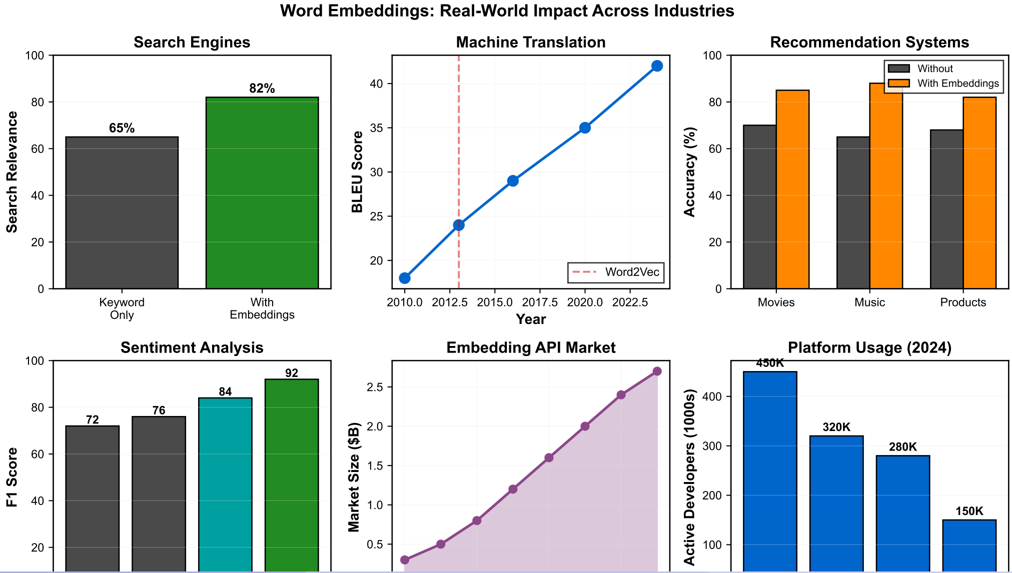
Word2Vec's limitations led directly to the transformer revolution in NLP

Part 4: Applications & Future

Visualizing the Embedding Space



Modern Applications Dashboard

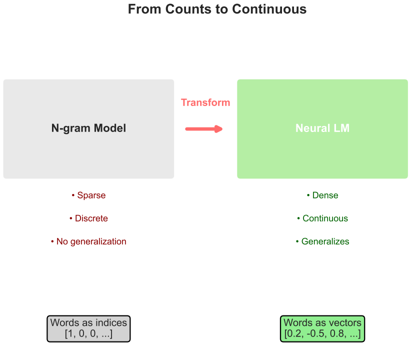


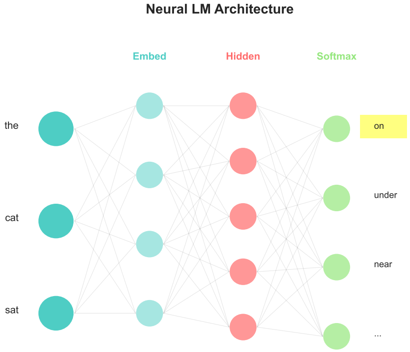
Hands-On: Using Word2Vec in Practice

```python
from gensim.models import Word2Vec
import numpy as np

# Train Word2Vec model
sentences = [["the", "cat", "sat", "on", "the", "mat"],
             ["the", "dog", "sat", "on", "the", "floor"]]

model = Word2Vec(sentences, vector_size=100, window=5,
                 min_count=1, sg=1)  # sg=1 for skip-gram

# Get word vectors
cat_vector = model.wv['cat']
dog_vector = model.wv['dog']

# Compute similarity
similarity = model.wv.similarity('cat', 'dog')
print(f"Similarity(cat, dog) = {similarity:.3f}")

# Find similar words
similar_words = model.wv.most_similar('cat', topn=3)
print(f"Words similar to 'cat': {similar_words}")

# Word arithmetic
result = model.wv.most_similar(positive=['king', 'woman'],
                                negative=['man'], topn=1)
print(f"king - man + woman = {result[0][0]}")
```

Gensim makes Word2Vec incredibly easy to use - training takes just minutes

From Word2Vec to Modern Transformers





Word2Vec's Legacy:

- Proved semantic learning possible

Modern Evolution:

- BERT: Contextual embeddings

Summary: The Word Embedding Revolution

What We Learned:

Key Concepts

- Distributional hypothesis
- Dense vector representations
- Skip-gram vs CBOW
- Negative sampling optimization
- Semantic arithmetic

Technical Skills

- Training Word2Vec models
- Evaluating embedding quality
- Visualizing semantic spaces
- Applying to downstream tasks

Practical Impact

- Powers modern search engines
- Enables machine translation
- Foundation for ChatGPT/Claude
- $100B+ market impact

Historical Significance

- 50,000+ citations
- Revolutionized NLP (2013)
- Led to transformer era
- Still widely used today

Core Insight: Words are not just symbols - they carry meaning in their

Quick Quiz: Test Your Understanding

Answer these questions to check your understanding:

1. What is the key insight of the distributional hypothesis?

Quick Quiz: Test Your Understanding

Answer these questions to check your understanding:

1. What is the key insight of the distributional hypothesis? → Words with similar contexts have similar meanings
2. Why is negative sampling needed in Word2Vec?

Quick Quiz: Test Your Understanding

Answer these questions to check your understanding:

1. What is the key insight of the distributional hypothesis? → Words with similar contexts have similar meanings
2. Why is negative sampling needed in Word2Vec? → To avoid expensive softmax over entire vocabulary
3. What's the difference between Skip-gram and CBOW?

Quick Quiz: Test Your Understanding

Answer these questions to check your understanding:

1. What is the key insight of the distributional hypothesis? $\rightarrow$ Words with similar contexts have similar meanings
2. Why is negative sampling needed in Word2Vec? $\rightarrow$ To avoid expensive softmax over entire vocabulary
3. What's the difference between Skip-gram and CBOW? $\rightarrow$ Skip-gram: word$\rightarrow$context, CBOW: context$\rightarrow$word
4. Why does "king - man + woman = queen" work?

Quick Quiz: Test Your Understanding

Answer these questions to check your understanding:

1. What is the key insight of the distributional hypothesis? $\rightarrow$ Words with similar contexts have similar meanings
2. Why is negative sampling needed in Word2Vec? $\rightarrow$ To avoid expensive softmax over entire vocabulary
3. What's the difference between Skip-gram and CBOW? $\rightarrow$ Skip-gram: word$\rightarrow$context, CBOW: context$\rightarrow$word
4. Why does "king - man + woman = queen" work? $\rightarrow$ Relationships are encoded as vector directions
5. What's the main limitation of Word2Vec?

Quick Quiz: Test Your Understanding

Answer these questions to check your understanding:

1. What is the key insight of the distributional hypothesis? → Words with similar contexts have similar meanings
2. Why is negative sampling needed in Word2Vec? → To avoid expensive softmax over entire vocabulary
3. What's the difference between Skip-gram and CBOW? → Skip-gram: word→context, CBOW: context→word
4. Why does "king - man + woman = queen" work? → Relationships are encoded as vector directions
5. What's the main limitation of Word2Vec? → Single vector per word (no context dependence)

If you can answer these, you understand the core of word embeddings!

Resources for Deep Dive

Essential Papers

- Mikolov et al. (2013a): Efficient Estimation
- Mikolov et al. (2013b): Distributed Representations
- Goldberg & Levy (2014): word2vec Explained
- Pennington et al. (2014): GloVe

Implementations

- Gensim (Python)
- TensorFlow Embeddings
- PyTorch nn.Embedding
- FastText library

Datasets & Tools

- Google News vectors
- GloVe pre-trained
- Embedding Projector
- Word2Vec demos

Lab Session Preview:

- Train Word2Vec on real corpus
- Explore semantic relationships
- Build a similarity search engine
- Visualize your own embeddings

Lab notebook: week02_word_embeddings_lab.ipynb