From Headlines to Narratives

A Complete Course in LLM-Based Narrative Extraction

Prof. Dr. Joerg Osterrieder

Graduate Course in Advanced NLP and Machine Learning

September 19, 2025

Course Overview: Comprehensive treatment of Large Language Models for narrative extraction from financial news. Mathematical foundations, practical implementations, and real-world applications.

Course Learning Objectives

By the end of this course, you will understand:

1. **Text Representation:** How raw text becomes mathematical objects (embeddings)
2. **Semantic Discovery:** How machines discover topics and themes in text
3. **Information Aggregation:** How individual headlines become coherent narratives
4. **Deep Learning Architecture:** How Transformers process and understand language
5. **Generation Theory:** How LLMs create new text and control their outputs
6. **Advanced NLP:** How machines understand complex linguistic phenomena
7. **Optimization:** How to train large language models effectively

Prerequisites: Linear algebra, probability theory, basic machine learning, Python programming

Course Roadmap

Pedagogical Approach: Each topic builds on previous concepts. We start with fundamental representations and progress to sophisticated generation models. Mathematical rigor combined with intuitive explanations and practical examples.

Part I: Mathematical Foundations of Text Embeddings

Learning Goals: Understand how text becomes numbers, why embeddings work, and the mathematical principles behind different embedding methods.

Why Do We Need Text Embeddings?

The Fundamental Problem:
Computers work with numbers, but language consists of discrete symbols (words).

Key Concept: One-Hot Encoding Limitations

Traditional approach: Represent each word as a binary vector with single 1.

- Vocabulary size = 50,000 words $\Rightarrow$ 50,000-dimensional vectors
- No semantic similarity: "king" and "queen" are orthogonal
- Sparse, inefficient, no generalization

Intuition

We want representations where semantically similar words have similar mathematical representations. This allows machines to understand that "king" and "queen" are more related than "king" and "banana".

The Solution: Dense vector representations (embeddings) that capture semantic relationships.

The Distributional Hypothesis

Key Concept: Firth's Principle (1957)

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

Mathematical Formulation: Words appearing in similar contexts should have similar representations.

Context Window Example:

- "The king ruled the kingdom wisely"
- "The queen governed the nation fairly"
- "The president led the country effectively"

Intuition

Words surrounded by similar words (ruled/governed/led, kingdom/nation/country) should have similar embeddings. This is the core insight behind all embedding methods.

Implications:

- Semantic similarity emerges from distributional patterns
- No manual feature engineering required
- Works across languages and domains

Historical Development of Embeddings

Timeline of Major Developments:

- **1986:** Rumelhart et al. - Distributed representations
- **1990s:** Latent Semantic Analysis (LSA) - SVD on term-document matrices
- **2003:** Neural Language Models (Bengio et al.)
- **2008:** Collobert & Weston - Multi-task learning for NLP
- **2013:** Word2Vec (Mikolov et al.) - Efficient neural embeddings
- **2014:** GloVe (Pennington et al.) - Global statistics + local context
- **2017:** Transformer (Vaswani et al.) - Self-attention revolution
- **2018:** BERT - Bidirectional contextual representations
- **2019+:** Large Language Models (GPT, T5, etc.)

Intuition

Each development solved limitations of previous approaches while building on their insights.

Section I Learning Objectives

After this section, you will be able to:

1. **Explain** the distributional hypothesis and why it enables semantic representations
2. **Derive** the Word2Vec objective functions from first principles
3. **Understand** the computational tricks (negative sampling, hierarchical softmax)
4. **Compare** different embedding methods and their trade-offs
5. **Implement** basic embedding algorithms
6. **Evaluate** embedding quality using intrinsic and extrinsic tasks
7. **Apply** embeddings to narrative extraction problems

Key Mathematical Tools:

- Probability distributions and conditional probability
- Gradient descent and backpropagation
- Matrix factorization and singular value decomposition
- Information theory (entropy, mutual information)

Word2Vec: The Breakthrough

Why Word2Vec Revolutionized NLP:

Key Concept: Core Innovation

Instead of complex neural architectures, use a simple objective: predict context words from target word (or vice versa).

Two Main Architectures:

1. **Skip-gram:** Given center word, predict surrounding words
   - Input: "king"
   - Output: "the", "ruled", "kingdom", "wisely"
2. **CBOW (Continuous Bag of Words):** Given context, predict center word
   - Input: "the", "ruled", "kingdom", "wisely"
   - Output: "king"

Intuition

If we can predict context from word (or word from context), then our learned representations must capture semantic meaning.

Skip-gram Architecture Deep Dive

Neural Network Structure:

1. **Input Layer:** One-hot encoded word $w_I \in \{0,1\}^V$
2. **Hidden Layer:** Dense embedding $h = W^T w_I \in \mathbb{R}^N$
3. **Output Layer:** Context word probabilities $y \in \mathbb{R}^V$

Key Insight: The hidden layer weights $W \in \mathbb{R}^{V \times N}$ become our word embeddings!

Key Concept: Embedding Matrix

Each row of W is the embedding vector for a word. Training the network to predict context automatically learns semantic representations.

Mathematical Objective:

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

where T = corpus size, c = context window size.

The Softmax Problem

Probability Computation:
For each context position, we need:

$$p(w_O|w_I) = \frac{\exp(v_{w_O}^T v_{w_I})}{\sum_{w=1}^{W} \exp(v_w^T v_{w_I})} \tag{2}$$

Computational Challenge:

- Denominator requires summing over entire vocabulary ($W \approx 50,000$)
- For each training example, for each context position
- Makes training prohibitively expensive

Key Concept: The Bottleneck

Computing softmax over large vocabularies is the main computational bottleneck in language modeling.

Two Solutions:

1. **Hierarchical Softmax:** Use binary tree structure
2. **Negative Sampling:** Approximate softmax with sampling

Negative Sampling: Elegant Solution

Key Insight: Instead of computing probabilities over all words, distinguish target from random "negative" samples.

Key Concept: Reformulation

Transform multi-class classification into binary classification problems.

Objective Function:

$$\log \sigma(v_{w_O}^T v_{w_I}) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P_n(w)} [\log \sigma(-v_{w_k}^T v_{w_I})] \tag{3}$$

Breaking it down:

- $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function
- First term: Maximize probability of actual context word
- Second term: Minimize probability of random "negative" words
- K = number of negative samples (typically 5-20)

Noise Distribution: $P_n(w) = \frac{U(w)^{3/4}}{\sum_w U(w)^{3/4}}$ favors frequent words

Training Word2Vec: Step by Step

Algorithm:

Initialize embedding matrices W_{in}, W_{out} randomly
for each training sentence **do**
  for each word w_t in sentence **do**
    for each context position $j \in [-c, c], j \neq 0$ **do**
      Sample K negative words: $w_{neg,1}, \ldots, w_{neg,K} \sim P_n$
      Compute gradients for positive pair (w_t, w_{t+j})
      Compute gradients for negative pairs $(w_t, w_{neg,k})$
      Update embeddings using SGD
    end for
  end for
end for

Gradient Computation: For positive pair: $\frac{\partial}{\partial v_{w_O}} \log \sigma(v_{w_O}^T v_{w_I}) = (\sigma(v_{w_O}^T v_{w_I}) - 1) v_{w_I}$

Learning Rate: Typically starts at 0.025, decreases during training

Word2Vec: What Did We Learn?

Emergent Properties:

1. **Semantic Similarity:** Similar words cluster together
   - Countries: France, Germany, Italy
   - Colors: red, blue, green
   - Emotions: happy, sad, angry
2. **Analogical Reasoning:** Linear relationships in embedding space
   - King - Man + Woman Queen
   - Paris - France + Germany Berlin
   - Walking - Walk + Swam Swimming
3. **Syntactic Patterns:** Grammatical relationships preserved
   - Verb tenses: run/running, eat/eating
   - Pluralization: cat/cats, mouse/mice

Intuition

The embedding space automatically organizes itself to reflect human conceptual knowledge, purely from distributional statistics!

Limitations of Word2Vec

What Word2Vec Cannot Do:

1. **Polysemy:** Same word, different meanings
   - "bank" (financial institution) vs "bank" (river side)
   - Single embedding cannot capture both meanings
2. **Context Dependency:** Meaning changes with context
   - "The bank was closed" vs "The river bank was muddy"
   - Static embeddings miss contextual nuances
3. **Out-of-Vocabulary:** Cannot handle unknown words
   - New words, typos, rare words
   - No representation available

Impact on Narrative Extraction:

- Financial terms often have multiple meanings
- Context crucial for disambiguation
- Need more sophisticated approaches

Solution Preview: Contextual embeddings (BERT, GPT)

GloVe: Combining Global and Local Statistics

Motivation: Word2Vec uses only local context windows. Can we also leverage global corpus statistics?

Key Concept: Co-occurrence Matrix

X_{ij} = number of times word j appears in context of word i across entire corpus.

Key Insight: Ratios of co-occurrence probabilities reveal semantic relationships.
Example: Ice vs Steam

- $P(solid|ice)/P(solid|steam)$ is large (ice is solid, steam is not)
- $P(water|ice)/P(water|steam)$ is close to 1 (both related to water)
- $P(fashion|ice)/P(fashion|steam)$ is close to 1 (neither related to fashion)

Mathematical Goal: Learn embeddings where $w_i^T w_j \approx \log P(j|i)$
This leads to the GloVe objective function.

GloVe Objective Function Derivation

Starting Point: We want $w_i^T w_j = \log P(j|i) = \log X_{ij} - \log X_i$
Problem: Different words have different total frequencies $X_i = \sum_k X_{ik}$
Solution: Add bias terms to absorb frequency differences.
Target: $w_i^T \tilde{w}_j + b_i + \tilde{b}_j = \log X_{ij}$
Least Squares Objective:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \tag{4}$$

Weighting Function: $f(x)$ handles zero entries and reduces influence of very common pairs.

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

Typically: $x_{max} = 100$, $\alpha = 0.75$

FastText: Handling the Unknown

Motivation: What about words not seen during training?
Examples in financial news:

- Company names: "Anthropic", "OpenAI"
- Technical terms: "DeFi", "stablecoin"
- Misspellings: "sentement" instead of "sentiment"

Key Concept: Subword Representations

Represent each word as sum of character n-gram embeddings.

Example: Word "banking" with trigrams:

- Special markers: "¡ba", "ban", "ank", "nki", "kin", "ing", "ng¿"
- Each n-gram has its own embedding z_g
- Word embedding: $v_{banking} = \sum_{g \in G_{banking}} z_g$

Mathematical Formulation:

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c \tag{6}$$

Advantages:

- Handles out-of-vocabulary words
- Captures morphological patterns
- Works well for morphologically rich languages

The Leap to Contextual Embeddings

The Polysemy Problem Revisited:
Traditional embeddings give one representation per word type.
Example: "bank"

- "I deposited money at the bank" (financial institution)
- "We sat by the river bank" (shore)
- "The pilot had to bank the airplane" (turn)

Key Concept: Contextual Embeddings

Generate different representations for the same word based on its context.

Mathematical Shift:

- Traditional: $\text{embed}(\text{"bank"}) = v_{bank}$ (fixed)
- Contextual: $\text{embed}(\text{"bank"}, \text{context}) = f(\text{context})$ (dynamic)

This requires more sophisticated architectures that can process entire sequences.

ELMo: Bidirectional Context

Embeddings from Language Models (ELMo)

Key Concept: Bidirectional Processing

To understand a word, look at both left and right context simultaneously.

Architecture:

1. Forward LSTM: $\vec{h}_{LM} = \text{LSTM}(w_1, ..., w_t)$
2. Backward LSTM: $\overleftarrow{h}_{LM} = \text{LSTM}(w_n, ..., w_t)$
3. Concatenation: $h_{LM} = [\vec{h}_{LM}; \overleftarrow{h}_{LM}]$

Training Objective: Maximize likelihood of predicting next/previous words:

$$\mathcal{L}_{forward} = \sum_{t=1}^{T} \log P(w_t | w_1, ..., w_{t-1}) \tag{7}$$

$$\mathcal{L}_{backward} = \sum_{t=1}^{T} \log P(w_t | w_{t+1}, ..., w_T) \tag{8}$$

Usage: Extract representations from different layers for downstream tasks.

BERT: Bidirectional Transformer Revolution

The Transformer Advantage:

Key Concept: Self-Attention vs RNNs

Instead of processing sequences left-to-right, attend to all positions simultaneously.

Key Innovations:

1. **Masked Language Modeling:** Randomly mask 15% of words, predict them
   - Input: "The [MASK] was very successful"
   - Target: Predict "company"
2. **Next Sentence Prediction:** Predict if sentence B follows sentence A
   - Helps model understand document structure

Mathematical Objectives:

MLM: $\mathcal{L}_{MLM} = -\mathbb{E}_{\mathcal{D}} \sum_{m \in M} \log P(x_m | x_{\setminus M})$

NSP: $\mathcal{L}_{NSP} = -\mathbb{E}_{(S_A, S_B)} \log P(y | [CLS], S_A, [SEP], S_B)$

Total: $\mathcal{L} = \mathcal{L}_{MLM} + \mathcal{L}_{NSP}$

Why BERT Works So Well

Bidirectional Context Benefits:

Key Concept: Full Context Access

Unlike left-to-right models, BERT sees entire sentence during training.

Example Analysis: "The animal didn't cross the street because it was too tired."

- Left-only model: Only sees "The animal didn't cross the street because it was too"
- BERT: Sees full sentence, understands "it" refers to "animal"

Mathematical Intuition: Masking forces model to use bidirectional context:

$$P(w_i|w_1, ..., w_{i-1}, [MASK], w_{i+1}, ..., w_n) \tag{9}$$

Transfer Learning Power:

- Pre-train on large unlabeled corpus
- Fine-tune on specific tasks with small labeled data
- Achieves state-of-the-art on multiple NLP benchmarks

Sentence-Level Embeddings

From Words to Sentences:

Key Concept: Compositional Challenge

How do we get embeddings for entire sentences or documents?

Naive Approaches (Don't Work Well):

- Average word embeddings: $\text{sent} = \frac{1}{n}\sum_{i=1}^{n} w_i$
- TF-IDF weighted average
- Simple concatenation

Why They Fail: Lose word order, ignore syntax, no interaction modeling.
Better Solutions:

1. **Doc2Vec:** Treat document as additional "word" in vocabulary
2. **Universal Sentence Encoder:** Deep averaging network or Transformer
3. **Sentence-BERT:** Fine-tune BERT for sentence similarity

Key Insight: Need architectures designed specifically for sentence-level tasks.

Sentence-BERT Architecture

The BERT Limitation:
Original BERT requires comparing every sentence pair individually:

- Input: [CLS] Sentence A [SEP] Sentence B [SEP]
- For n sentences: $O(n^2)$ comparisons
- Impractical for large-scale similarity search

Key Concept: Siamese Networks Solution

Process sentences independently, then compare embeddings.

Architecture:

1. Encode sentences separately: $u = \text{BERT}(S_A)$, $v = \text{BERT}(S_B)$
2. Pool representations (mean, max, or CLS token)
3. Compute similarity: $\text{sim}(u, v) = \frac{u \cdot v}{||u|| \cdot ||v||}$

Training Strategies:

- Classification: $o = \text{softmax}(W[u; v; |u - v|])$
- Regression: Direct cosine similarity
- Triplet loss: Anchor, positive, negative examples

Embedding Quality Evaluation

How Do We Know If Embeddings Are Good?

Key Concept: Intrinsic vs Extrinsic Evaluation

- Intrinsic: Test embedding properties directly
- Extrinsic: Use embeddings in downstream tasks

Intrinsic Tasks:

1. **Word Similarity:** Compare with human judgments
   - WordSim-353, SimLex-999 datasets
   - Spearman correlation with human ratings
2. **Analogical Reasoning:** a:b :: c:? problems
   - Google analogy dataset
   - Accuracy on completing analogies
3. **Clustering:** Do similar words cluster together?

Extrinsic Tasks:

- Text classification accuracy
- Named entity recognition performance
- Sentiment analysis improvement

Understanding the Structure of Embedding Spaces

Key Concept: Manifold Hypothesis

High-dimensional embeddings actually lie on lower-dimensional manifolds.

Intrinsic Dimensionality:

$$ID = \lim_{r \to 0} \frac{\log \mathbb{E}[N(r)]}{\log r} \tag{10}$$

Intuitive Explanation:

- $N(r)$ = number of points within distance r
- If data lies on d-dimensional manifold, $N(r) \propto r^d$
- Most 300D embeddings have intrinsic dimension 10-50

Anisotropy Problem: Embeddings often concentrated in narrow cone:

$$\text{avg-cos} = \frac{2}{n(n-1)} \sum_{i<j} \cos(v_i, v_j) \tag{11}$$

When avg-cos is high, all embeddings are similar $\Rightarrow$ poor discriminability.

Distance Metrics for Text Similarity

How Do We Measure Similarity Between Text Embeddings?

1. **Cosine Similarity:** Most common in NLP

$$\cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}} \tag{12}$$

   Why cosine? Ignores magnitude, focuses on direction.

2. **Euclidean Distance:** $d(A, B) = \sqrt{\sum_i (a_i - b_i)^2}$
   When to use? When magnitude matters (frequency-based features).

3. **Earth Mover's Distance:** Optimal transport between distributions

$$EMD(P, Q) = \min_{\gamma \in \Pi(P,Q)} \sum_{i,j} \gamma_{ij} ||x_i - y_j|| \tag{13}$$

   Intuition: Minimum cost to transform one distribution into another.

Section I Summary and Key Takeaways

What We've Learned About Text Embeddings:

1. **Distributional Hypothesis:** Semantic similarity from contextual similarity
2. **Word2Vec:** Efficient neural embeddings via prediction tasks
3. **GloVe:** Combining local and global statistics
4. **FastText:** Subword representations for robustness
5. **Contextual Models:** ELMo, BERT for dynamic representations
6. **Distance Metrics:** How to measure semantic similarity

Key Mathematical Insights:

- Prediction objectives naturally capture semantics
- Matrix factorization perspectives
- Probabilistic interpretations of embeddings
- Geometric properties of embedding spaces

Applications to Narrative Extraction:

- Represent headlines as dense vectors
- Cluster semantically similar headlines
- Build narrative coherence measures

Part II: Topic Modeling and Narrative Discovery

Learning Goals: Understand how machines discover hidden thematic structure in text collections and extract coherent narratives from news headlines.

What Is a Topic? Intuitive Understanding

Human Perspective: When we read news articles, we naturally group them by themes:

- Economic news: GDP, inflation, unemployment, interest rates
- Political news: elections, policies, candidates, polls
- Technology news: AI, startups, innovations, regulations

Key Concept: Computational Topic Definition

A topic is a probability distribution over words that captures a coherent theme.

Example Topic Distributions:
Topic 1 (Economics): market (0.08), economy (0.06), growth (0.05), GDP (0.04)...
Topic 2 (Politics): election (0.09), candidate (0.07), vote (0.05), poll (0.04)...
Goal: Automatically discover these topics from raw text without supervision.

The Generative Story: How Documents Are "Born"

Latent Dirichlet Allocation (LDA) Intuition:
Imagine an author writing a document:

1. **Choose Topic Mix:** "I'll write 60% about economics, 30% politics, 10% technology"
   - Draw from Dirichlet: $\theta_d \sim \text{Dir}(\alpha)$
2. **For Each Word:**
   - Choose which topic: $z_{dn} \sim \text{Multinomial}(\theta_d)$
   - Pick word from that topic: $w_{dn} \sim \text{Multinomial}(\phi_{z_{dn}})$

Key Concept: Generative Process

LDA assumes documents are created by this probabilistic process. Our job is to reverse-engineer the hidden topics and mixtures.

Key Assumption: Documents are "bags of words" - order doesn't matter for topic assignment.

LDA Mathematical Framework

The Complete Generative Process:
For the entire corpus:

1. For each topic $k = 1, ..., K$:
   - Draw word distribution: $\phi_k \sim \text{Dir}(\beta)$

For each document $d = 1, ..., D$:

1. Draw topic distribution: $\theta_d \sim \text{Dir}(\alpha)$
2. For each word position $n = 1, ..., N_d$:
   - Choose topic: $z_{dn} \sim \text{Multinomial}(\theta_d)$
   - Choose word: $w_{dn} \sim \text{Multinomial}(\phi_{z_{dn}})$

Joint Probability:

$$p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi} | \alpha, \beta) = \prod_{k=1}^{K} p(\phi_k | \beta) \prod_{d=1}^{D} p(\theta_d | \alpha) \prod_{n=1}^{N_d} p(z_{dn} | \theta_d) p(w_{dn} | z_{dn}, \phi_{z_{dn}}) \tag{14}$$

The Inference Challenge

What We Want to Compute:
Given observed words $\boldsymbol{w}$, find posterior distributions:

$$p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{w}, \alpha, \beta) = \frac{p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi}|\alpha, \beta)}{p(\boldsymbol{w}|\alpha, \beta)} \tag{15}$$

The Problem: Denominator is intractable!

$$p(\boldsymbol{w}|\alpha, \beta) = \int \sum_{\boldsymbol{z}} p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi}|\alpha, \beta) d\boldsymbol{\theta} d\boldsymbol{\phi} \tag{16}$$

This integral over all possible topic assignments and parameters cannot be computed exactly.

Key Concept: Approximate Inference

We need approximation methods: variational inference or sampling.

Two Main Approaches:

1. **Variational Bayes:** Approximate with simpler distributions
2. **Gibbs Sampling:** Sample from posterior using MCMC

Variational Inference: The Approximation

Core Idea: Replace intractable posterior with tractable approximation.

Key Concept: Mean Field Approximation

Assume independence between latent variables:

$$q(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi} | \boldsymbol{\gamma}, \boldsymbol{\xi}, \boldsymbol{\lambda}) = \prod_{d=1}^{D} q(\theta_d | \gamma_d) \prod_{k=1}^{K} q(\phi_k | \lambda_k) \prod_{d=1}^{D} \prod_{n=1}^{N_d} q(z_{dn} | \xi_{dn}) \tag{17}$$

Variational Objective (ELBO):

$$\mathcal{L} = \mathbb{E}_q[\log p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi} | \alpha, \beta)] - \mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi})] \tag{18}$$

Intuition: Maximize likelihood while keeping approximation close to prior.
Coordinate Ascent: Optimize each parameter while holding others fixed.

LDA Update Equations

Coordinate Ascent Updates:
Topic Assignment:

$$\xi_{dni} \propto \beta_{iw_{dn}} \exp(\Psi(\gamma_{di}) - \Psi(\sum_{j} \gamma_{dj})) \tag{19}$$

Document-Topic Distribution:

$$\gamma_{di} = \alpha_i + \sum_{n=1}^{N_d} \xi_{dni} \tag{20}$$

Topic-Word Distribution:

$$\lambda_{kw} = \beta_w + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \mathbf{1}[w_{dn} = w] \xi_{dnk} \tag{21}$$

What's Happening:

- ξ: Soft topic assignments for each word
- γ: Document's topic mixture (pseudo-counts)
- λ: Topic's word mixture (pseudo-counts)
- Ψ: Digamma function (derivative of log-gamma)

Neural Topic Models: Why Go Neural?

Limitations of Classical Topic Models:

1. **Bag of Words:** Ignores word order and syntax
2. **Discrete Topics:** Hard assignments, no continuous representations
3. **Fixed Vocabulary:** Cannot handle out-of-vocabulary words
4. **No Transfer Learning:** Each corpus modeled independently

Key Concept: Neural Solution

Use neural networks to learn topic representations that can leverage pre-trained embeddings and capture more complex patterns.

Variational Autoencoder Approach:

- **Encoder:** Maps documents to topic distributions
- **Decoder:** Reconstructs documents from topics
- **Training:** End-to-end optimization via reparameterization trick

Advantages:

- Incorporates pre-trained word embeddings
- Handles continuous topic representations
- Scales to large corpora
- Enables transfer learning

VAE-LDA Architecture Deep Dive

Encoder Network (Inference):
Document representation: $\boldsymbol{d} \in \mathbb{R}^V$ (bag-of-words or TF-IDF)

1. Hidden layers: $h_1 = \text{ReLU}(W_1 \boldsymbol{d} + b_1)$
2. Output parameters:

$$\mu = W_\mu h_1 + b_\mu \tag{22}$$

$$\log \sigma^2 = W_\sigma h_1 + b_\sigma \tag{23}$$

3. Approximate posterior: $q_\phi(\theta|d) = \mathcal{N}(\mu, \text{diag}(\sigma^2))$

Reparameterization Trick:

$$\theta = \mu + \epsilon \odot \sigma, \quad \epsilon \sim \mathcal{N}(0, I) \tag{24}$$

Why This Works: Allows gradients to flow through random sampling.
Decoder Network (Generation):

$$p_\psi(w|\theta) = \text{Softmax}(W\theta + b) \tag{25}$$

The BERTopic Pipeline Explained:

Key Concept: Three-Stage Process

1) Embed documents, 2) Cluster embeddings, 3) Extract topic words

Stage 1: Document Embeddings

$$e_i = \text{BERT}(\text{document}_i) \tag{26}$$

Use sentence-transformers for high-quality document representations.

Stage 2: Dimensionality Reduction + Clustering

UMAP: $\mathcal{L} = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} + (1 - p_{ij}) \log \frac{1-p_{ij}}{1-q_{ij}}$

HDBSCAN: Density-based clustering with hierarchy.

Stage 3: Topic Word Extraction

Class-based TF-IDF:

$$w_{t,c} = tf_{t,c} \cdot \log \left(1 + \frac{|\text{documents}|}{|\text{documents containing } t|} \right) \tag{27}$$

Advantage: Leverages powerful pre-trained representations while maintaining interpretability.

Section Summary: Topic Modeling

Key Concepts Covered:

1. **Probabilistic Topic Models:** LDA as the foundation
2. **Inference Methods:** Variational Bayes and Gibbs sampling
3. **Neural Extensions:** VAE-based topic models
4. **Modern Approaches:** BERTopic and transformer-based methods
5. **Evaluation:** Coherence metrics and human interpretability

Mathematical Tools Learned:

- Dirichlet distributions and conjugacy
- Variational inference and ELBO
- Coordinate ascent optimization
- Reparameterization trick for neural models

Narrative Applications:

- Discover themes in financial news
- Track narrative evolution over time
- Identify emerging market concerns
- Cluster related news stories

Next: How to aggregate individual headlines into coherent narrative threads.

Part III: From Headlines to Narratives

Learning Goals: Understand how individual news headlines are aggregated into coherent narrative threads using clustering, summarization, and information fusion techniques.

The Aggregation Challenge

From Individual Headlines to Coherent Stories:
Input: Thousands of individual headlines per day

- "Fed raises interest rates by 0.75 basis points"
- "Markets tumble following Fed announcement"
- "Inflation concerns drive monetary policy shift"
- "Dollar strengthens on hawkish Fed stance"

Goal: Group into coherent narratives

- **Narrative:** "Federal Reserve tightening cycle"
- **Key events:** Rate hikes, market reactions, policy statements
- **Temporal evolution:** How story develops over time

Key Concept: Aggregation Tasks

1) Clustering similar headlines, 2) Extracting key information, 3) Building temporal threads, 4) Resolving contradictions

Why Graphs for Text Clustering?

Key Concept: Similarity Networks

Represent documents as nodes, similarity as edge weights. Clustering becomes graph partitioning.

Similarity Graph Construction:

1. **Nodes:** Each headline/document
2. **Edges:** Similarity scores between documents
3. **Weights:** $w_{ij} = \text{similarity}(\text{doc}_i, \text{doc}_j)$

Using BERT Embeddings:

$$w_{ij} = \frac{\text{BERT}(h_i) \cdot \text{BERT}(h_j)}{||\text{BERT}(h_i)|| \cdot ||\text{BERT}(h_j)||} \tag{28}$$

Graph Properties:

- Dense clusters = coherent topics
- Bridge nodes = documents spanning multiple topics
- Isolated nodes = unique/outlier content

Spectral Clustering: Mathematical Foundation

Graph Partitioning as Optimization:

Key Concept: Graph Cut Problem

Find partition that minimizes connections between groups while maximizing internal connections.

Graph Laplacian: $L = D - W$

- $D_{ii} = \sum_j W_{ij}$ (degree matrix)
- W_{ij} = similarity weights

Normalized Cut Objective:

$$\text{NCut}(A, B) = \frac{\text{cut}(A, B)}{\text{vol}(A)} + \frac{\text{cut}(A, B)}{\text{vol}(B)} \tag{29}$$

where $\text{vol}(A) = \sum_{i \in A} d_i$ is the volume.
Relaxed Solution: Eigenvectors of normalized Laplacian

$$\min_H \text{tr}(H^T L_{norm} H) \quad \text{s.t.} \quad H^T H = I \tag{30}$$

Algorithm: Use k smallest eigenvectors as features, then apply k-means.

The Summarization Challenge:
Given cluster of related headlines, extract key information without redundancy.

Key Concept: Optimization Formulation

Balance informativeness, coverage, and diversity.

Integer Linear Programming Approach:
Binary variables: $x_i = 1$ if sentence i selected
Objective Function:

$$\max \sum_i w_i x_i - \lambda \sum_{i,j} s_{ij} x_i x_j \tag{31}$$

Terms Explained:

- w_i: Importance score of sentence i
- s_{ij}: Similarity between sentences i and j
- λ: Penalty for redundancy

Constraints:

$$\sum_i l_i x_i \leq L \quad \text{(length limit)} \tag{32}$$

$$\sum_{i \in C_k} x_i \geq 1 \quad \forall k \quad \text{(coverage)} \tag{33}$$

Event Chain Extraction

Building Temporal Narrative Threads:

Key Concept: Narrative Chains

Sequences of events that typically occur together in stories.

Example Financial Narrative Chain:

1. Company reports earnings miss
2. Stock price drops
3. Analysts downgrade ratings
4. CEO makes reassuring statements
5. Market stabilizes

Mathematical Model: Event sequence probability:

$$P(e_1, ..., e_n) = P(e_1) \prod_{i=2}^{n} P(e_i | e_1, ..., e_{i-1}) \tag{34}$$

Learning Objective:

$$\max_{\theta} \sum_{chains} \log P_\theta(\text{chain}) \tag{35}$$

Applications: Predict likely next events, identify narrative completion, detect anomalous event sequences.

Part IV: Transformer Architecture

Learning Goals: Master the mathematical foundations of Transformers, understand attention mechanisms, and learn how these architectures enable sophisticated narrative understanding.

Why Transformers Revolutionized NLP

The RNN Bottleneck:

Key Concept: Sequential Processing Limitation

RNNs must process sequences step-by-step, preventing parallelization.

Problems with RNNs:

1. **Sequential Dependency:** h_t depends on h_{t-1}
2. **Long-Range Dependencies:** Information loss over long sequences
3. **Training Speed:** Cannot parallelize across time steps
4. **Gradient Issues:** Vanishing/exploding gradients

Transformer Innovation:

- **Attention:** Direct connections between all positions
- **Parallelization:** Process entire sequence simultaneously
- **Long-Range:** No information bottleneck
- **Scalability:** Efficient on modern hardware

Intuition

Instead of passing information sequentially, attention allows direct communication between any two positions in the sequence.

Attention Mechanism: Core Intuition

What Is Attention?

Key Concept: Information Retrieval Analogy

Given a query, find relevant information from a database and compute weighted combination.

Three Components:

1. **Query (Q):** What information are we looking for?
2. **Key (K):** What information is available?
3. **Value (V):** The actual information content

Example: Translating "The cat sat on the mat"

- When translating "cat", query = representation of "cat"
- Keys = representations of all input words
- Values = contextual information from all words
- Attention weights = how much each input word is relevant to "cat"

Mathematical Formulation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{36}$$

Scaled Dot-Product Attention: Step by Step

Breaking Down the Attention Formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{37}$$

Step 1: Compute Similarities QK^T: Dot products between queries and keys

- Large values = high similarity
- Result: $n \times n$ similarity matrix

Step 2: Scale Divide by $\sqrt{d_k}$ to prevent vanishing gradients

- Without scaling: softmax becomes too peaked
- With scaling: maintains gradient flow

Step 3: Normalize Softmax ensures weights sum to 1:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{38}$$

Step 4: Weighted Combination Multiply attention weights by values to get final output.

Multi-Head Attention: Why Multiple Heads?

Single Head Limitation:
One attention mechanism can only capture one type of relationship.

Key Concept: Multiple Representation Subspaces

Different heads can focus on different linguistic phenomena.

Examples of Head Specialization:

- **Head 1:** Subject-verb relationships
- **Head 2:** Adjective-noun modifications
- **Head 3:** Long-range dependencies
- **Head 4:** Local syntactic patterns

Mathematical Formulation:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{39}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{40}$$

Key Insight: Each head learns different linear projections of Q, K, V, allowing specialization.
Computational Cost: h heads with $d_{head} = d_{model}/h$ has same cost as single head.

Self-Attention vs Cross-Attention

Two Types of Attention:

1. **Self-Attention:** Q, K, V all come from same sequence
   - Used in: BERT, GPT encoders
   - Purpose: Understand relationships within sequence
   - Example: "The animal didn't cross the street because it was too tired"
   - "it" attends strongly to "animal"
2. **Cross-Attention:** Q from target, K,V from source
   - Used in: Encoder-decoder models
   - Purpose: Align target with source information
   - Example: Translation - target language queries attend to source language

For Narrative Extraction:

- Self-attention captures relationships between events in same document
- Cross-attention can link events across different documents
- Enables building coherent narrative threads

Part V: Large Language Models for Generation

Learning Goals: Understand how LLMs generate text, control their outputs, and apply them to narrative generation tasks.

Language Modeling: The Foundation

What Is Language Modeling?

Key Concept: Probability Distribution Over Text

A language model assigns probabilities to sequences of words/tokens.

Applications:

- **Text Generation:** Sample from the distribution
- **Text Completion:** Find most likely continuation
- **Quality Assessment:** High probability = fluent text
- **Compression:** Use as probabilistic code

Chain Rule Decomposition:

$$P(x_1, ..., x_T) = \prod_{t=1}^{T} P(x_t | x_1, ..., x_{t-1}) \tag{41}$$

Autoregressive Principle: Model each word conditioned on all previous words.

Intuition

If we can accurately predict the next word given context, we understand language structure.

Training Language Models

Maximum Likelihood Objective:
Given corpus $\mathcal{D} = \{s_1, s_2, ..., s_N\}$, maximize likelihood:

$$\mathcal{L}_{MLE} = \frac{1}{|\mathcal{D}|} \sum_{s \in \mathcal{D}} \log P_\theta(s) \tag{42}$$

Expanding for Autoregressive Model:

$$\mathcal{L}_{MLE} = \frac{1}{|\mathcal{D}|} \sum_{s \in \mathcal{D}} \sum_{t=1}^{|s|} \log P_\theta(x_t | x_{<t}) \tag{43}$$

Cross-Entropy Loss: Equivalent negative log-likelihood:

$$\mathcal{L}_{CE} = -\frac{1}{T} \sum_{t=1}^{T} \log P_\theta(x_t | x_{<t}) \tag{44}$$

Teacher Forcing: During training, use ground truth context (not model predictions).
Why This Works: Gradient descent finds parameters that assign high probability to observed text.

The Exposure Bias Problem

Training vs Inference Mismatch:

Key Concept: Distribution Shift

Model trained on ground truth context but used with its own predictions.

Training: $P(x_t|x_1^{truth}, ..., x_{t-1}^{truth})$
Inference: $P(x_t|x_1^{model}, ..., x_{t-1}^{model})$
Consequences:

- Errors compound over time
- Model hasn't seen its own mistakes during training
- Quality degrades for longer sequences

Solutions:
Scheduled Sampling:

$$x_t^{input} = \begin{cases} x_t^{truth} & \text{with prob } 1 - \epsilon_t \\ \arg\max P(x|x_{<t}) & \text{with prob } \epsilon_t \end{cases} \tag{45}$$

Start with teacher forcing, gradually use model predictions.

Decoding: From Probabilities to Text

The Generation Challenge:
Given trained model $P_\theta(x_t|x_{<t})$, how do we generate text?

Key Concept: Search vs Sampling

Two paradigms: find most likely sequence (search) or sample from distribution.

Greedy Decoding: Always pick most likely next word

$$x_t = \arg\max_x P_\theta(x|x_{<t}) \tag{46}$$

Problems:

- No backtracking - locally optimal choices
- Repetitive, boring output
- Misses creative alternatives

Beam Search: Keep top k sequences

$$\text{score}(x_1, ..., x_t) = \sum_{i=1}^{t} \log P(x_i|x_{<i}) \tag{47}$$

Sampling Methods: Introduce controlled randomness for diversity.

Part VI: Advanced NLP for Narratives

Learning Goals: Master advanced NLP techniques including coreference resolution, relation extraction, temporal reasoning, and causal analysis for narrative understanding.

Coreference Resolution: Linking Mentions

The Coreference Problem:

Key Concept: Entity Tracking

Identify when different phrases refer to the same real-world entity.

Example: "Apple Inc. announced strong earnings. The company's revenue exceeded expectations. Its stock price surged after the announcement."

Coreference Chains:

- "Apple Inc." $\rightarrow$ "The company" $\rightarrow$ "Its"
- All refer to the same entity

Why It Matters for Narratives:

- Track main characters/entities across documents
- Build coherent storylines
- Understand entity-centric narratives

Mathematical Approach: Score all mention pairs, then cluster.

$$s(m_i, m_j) = w^T \phi(m_i, m_j, \text{context}) \tag{48}$$

Temporal Reasoning in Financial Narratives

Time Matters in Financial Stories:

Key Concept: Temporal Dependencies

Financial narratives have complex temporal structures: cause-effect relationships, simultaneous events, overlapping periods.

Allen's Interval Algebra: 13 basic temporal relations

- Before: "Fed meeting" before "market reaction"
- During: "Earnings season" during "Q3"
- Overlaps: "Trade war" overlaps "election period"

Temporal Graph Construction:

- Nodes: Events/time periods
- Edges: Temporal relations
- Constraints: Ensure consistency

Constraint Propagation:

$$r_{AC} = r_{AB} \circ r_{BC} \tag{49}$$

If A before B and B before C, then A before C.

Part VII: Mathematical Optimization

Learning Goals: Understand the optimization challenges in training large language models and the mathematical techniques that make it possible.

The Optimization Challenge in NLP

Why Is Training LLMs Hard?

Key Concept: Scale and Complexity

Modern LLMs have billions of parameters trained on trillions of tokens.

Unique Challenges:

1. **High Dimensionality:** Parameter spaces with billions of dimensions
2. **Non-Convexity:** Multiple local minima, saddle points
3. **Discrete Outputs:** Text generation is fundamentally discrete
4. **Long Sequences:** Gradient flow over long dependencies
5. **Memory Constraints:** GPU memory limitations

Loss Landscape Properties:

- Neural networks have complex, high-dimensional loss surfaces
- Good local minima often have similar quality
- Wide minima generalize better than sharp minima

Intuition

Modern optimizers must navigate these complex landscapes efficiently while avoiding poor local minima.

Beyond Cross-Entropy: Advanced Loss Functions

Cross-Entropy Limitations:
Standard loss: $\mathcal{L}_{CE} = -\sum_i y_i \log p_i$
Problems:

- Treats all errors equally
- Doesn't handle class imbalance
- No robustness to label noise

Focal Loss for Imbalanced Data:

$$\mathcal{L}_{focal} = -\alpha_t (1 - p_t)^\gamma \log(p_t) \tag{50}$$

Intuition: Down-weight easy examples, focus on hard examples.

- $\gamma = 0$: Standard cross-entropy
- $\gamma > 0$: Reduces loss for confident predictions
- α_t: Class-dependent weighting

Label Smoothing:

$$y_i' = (1 - \epsilon) y_i + \frac{\epsilon}{K} \tag{51}$$

Prevents overconfident predictions by mixing true labels with uniform distribution.

Adam Optimizer: The Workhorse of Deep Learning

Why Adam Works So Well:

Key Concept: Adaptive Learning Rates

Different parameters need different learning rates. Adam adapts automatically.

Algorithm:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad \text{(momentum)} \tag{52}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad \text{(variance)} \tag{53}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \text{(bias correction)} \tag{54}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad \text{(bias correction)} \tag{55}$$

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{56}$$

Intuitive Explanation:

- m_t: Exponential moving average of gradients (momentum)
- v_t: Exponential moving average of squared gradients (variance)
- Bias correction: Accounts for initialization at zero
- Final update: Normalized by standard deviation

Learning Rate Scheduling for Transformers

The Warmup Strategy:

Key Concept: Gradual Learning Rate Increase

Start with small learning rate, increase to maximum, then decay.

Mathematical Formula:

$$\eta_t = d_{model}^{-0.5} \cdot \min(t^{-0.5}, t \cdot \text{warmup}^{-1.5}) \tag{57}$$

Why Warmup Works:

- Random initialization creates large gradients initially
- Large learning rates can destabilize training
- Warmup allows model to "settle" before aggressive learning
- After warmup, decrease learning rate for fine-tuning

Practical Guidelines:

- Warmup steps: Typically 4000-10000 steps
- Peak learning rate: Often 1e-4 to 5e-4
- Decay schedule: Linear, cosine, or exponential

Course Conclusion

Journey Completed:
We've traveled from basic word representations to sophisticated narrative generation systems, covering:

- Mathematical foundations of text representation
- Probabilistic and neural topic modeling
- Multi-document information aggregation
- Transformer architecture and attention mechanisms
- Large-scale language model training and generation
- Advanced NLP techniques for narrative understanding
- Optimization methods for training deep networks

Key Achievement: You now understand the complete mathematical pipeline from raw headlines to extracted narratives using state-of-the-art LLM methods.
Applications: Financial narrative analysis, social media monitoring, news summarization, content recommendation, and much more.

Thank You

Questions and Discussion

Contact:
Prof. Dr. Joerg Osterrieder

Course Materials:
Slides, code, and exercises available

Further Reading:
Comprehensive bibliography and resources provided