Lecture Information

Course: CSCI E-89B: Natural Language Processing
Lecture: Lecture 8
Topic: Structural Topic Modeling (STM)
Date: Fall 2024

Contents

1 Quiz Review: LDA and Topic Modeling

Overview

This lecture extends LDA to Structural Topic Modeling (STM), which incorporates document-level metadata (covariates) into topic modeling. We begin with a review of LDA concepts.

1.1 LDA Quiz Questions

What LDA Does

Question: Which statement best describes what LDA does?
Correct Answer: LDA assumes each document is a **mixture of topics**.
Why other options are wrong:

- "Assigns a single topic to each document" — No, LDA assigns **probability distributions** over topics
- "Supervised algorithm requiring labels" — No, LDA is **unsupervised**
- "Uses K-means" — No, LDA uses probabilistic inference, not K-means

1.2 Determining Optimal Number of Topics

Key Challenge in LDA

Determining the optimal number of topics is a significant challenge:

- **Too many topics**: Different topics become similar (redundancy)
- **Too few topics**: Unrelated concepts get combined
- Number of topics is a **hyperparameter** chosen before training

Methods for choosing K:

1. Maximize coherence and exclusivity (balance both)
2. Maximize held-out likelihood (test set likelihood)
3. Domain knowledge and interpretability

1.3 Role of the Dirichlet Distribution

Dirichlet Distribution in LDA

The Dirichlet distribution generates **topic proportions** (prevalence) for each document:

$$\theta_d \sim \text{Dirichlet}(\alpha)$$

θ_d is a vector of probabilities summing to 1, representing how much each topic contributes to document d.

1.4 LDA vs NMF

Important

Key Difference:

- **LDA**: Probabilistic model using maximum likelihood estimation
- **NMF**: Deterministic matrix algebra ($V \approx W \cdot H$)

NMF does **not** "break down documents into additive parts"—it decomposes a matrix into a **product** (multiplication) of two non-negative matrices.

2 Challenges with Sequence Autoencoders

Overview

Before diving into STM, we address a common challenge students face: building autoencoders for text sequences. This is significantly harder than image autoencoders.

2.1 Why Sequence Autoencoders Are Difficult

The Bottleneck Problem

When building a sequence autoencoder:

- You compress an entire sequence (many vectors) into a **single vector**
- This bottleneck loses the **time component**
- Reconstruction becomes extremely difficult without sufficient data

Comparison: Images vs Text

Image Autoencoders:

- Easier because spatial relationships are preserved through convolutions
- Even with compression, local structure remains

Text Autoencoders:

- Entire sentence compressed to single vector
- All word order and sequence information must be encoded
- Requires **enormous** amounts of training data

2.2 Practical Solutions

Strategies for Better Results

1. **Increase bottleneck dimension**: If reconstruction fails, try larger latent representations
2. **Data augmentation**: Create artificial training samples
   - Replace words with synonyms
   - Drop or shuffle words
   - Vary sentence structure
3. **Alternative architecture**: Skip the bottleneck entirely
   - Use sequence-to-sequence without compression
   - Train embeddings without the "encoding" constraint

Historical Note

Early machine translation systems tried this bottleneck approach—compress source sentence to a vector, then decode to target language. This was state-of-the-art briefly, but was abandoned because of the exact difficulties described above. Modern systems (Transformers) avoid hard bottlenecks.

3 Maximum Likelihood Estimation Revisited

Overview

Understanding MLE is crucial for topic modeling. We use an intuitive analogy before applying it to STM.

3.1 The Wet Cat Analogy

Intuitive MLE Example

Your cat comes home wet. What happened?
Possible explanations:

- It's raining outside $\Rightarrow P(\text{cat wet}|\text{rain}) \approx 1$
- Someone deliberately sprayed the cat $\Rightarrow P(\text{cat wet}|\text{sprayed}) < 1$

MLE conclusion: Most likely it was raining, because that explanation maximizes the probability of observing a wet cat.
Caveat: MLE finds the most likely explanation given the model, but isn't always "correct"—the model could be wrong!

3.2 Non-uniqueness in Topic Models

LDA Results Vary Between Runs

LDA/STM may produce different results each time because:

1. **Label switching**: Topic 1 and Topic 2 could swap
2. **Different local optima**: Multiple valid topic configurations exist
3. **Random initialization**: Starting point affects final solution

Multiple Valid Topic Configurations

Given documents:

- Doc 1: "excellent but difficult"
- Doc 2: "interesting but fast"
- Doc 3: "difficult but interesting"

Option A: Topic 1 = {excellent, difficult}, Topic 2 = {interesting, fast}

- Doc 1: 100% Topic 1
- Doc 2: 100% Topic 2
- Doc 3: 50% each

Option B: Topic 1 = {interesting, fast}, Topic 2 = {difficult, interesting}

- Doc 1: 100% Topic 2
- Doc 2: 100% Topic 1
- Doc 3: 50% each (different composition!)

Both are valid MLE solutions! That's why we run multiple times and select the best.

4 Limitations of LDA

Overview

LDA is powerful but has limitations that motivate Structural Topic Modeling (STM).

4.1 The Metadata Problem

Metadata (Covariates)

Metadata is information **about** documents, not the text itself:

- Author name, gender, age
- Publication date
- Source (New York Times, Financial Times, etc.)

- Department or category
- Any other document-level attributes

Metadata Structure

Document	Gender	Author	Year
"Cats sat..."	Female	Amanda Smith	2024
"Dog ran away..."	Male	Douglas Parker	2025

This metadata could help predict topic distributions but LDA ignores it.

Why Ignoring Metadata is Wasteful

If you know:

- An author's typical writing topics
- A publication's editorial focus
- Time periods when certain topics were trending

Ignoring this information makes topic assignment less accurate!

5 Structural Topic Modeling (STM)

Overview

STM extends LDA by incorporating document-level metadata (covariates) directly into the model, improving topic assignment accuracy and enabling hypothesis testing about how covariates affect topic prevalence.

5.1 The STM Model

STM vs LDA

- **LDA**: $\theta_d \sim \text{Dirichlet}(\alpha)$ (same for all documents)
- **STM**: $\theta_d \sim \text{Logistic-Normal}(\mu_d, \Sigma)$ where μ_d depends on covariates

5.2 The Logistic Normal Distribution

Logistic Normal for Topic Proportions

In STM, topic proportions are generated as:

$$\theta_d | X_d \sim \text{Logistic-Normal}(X_d\gamma, \Sigma)$$

where:

- X_d: Covariate vector for document d (metadata)
- γ: Coefficient matrix (to be estimated)

- Σ: Covariance matrix (to be estimated)

How it works:

1. Compute $\mu_d = X_d\gamma = \gamma_0 + \gamma_1 x_{d,1} + \gamma_2 x_{d,2} + \dots$
2. Draw $\eta_d \sim \mathcal{N}(\mu_d, \Sigma)$ (multivariate normal)
3. Apply softmax: $\theta_d = \text{softmax}(\eta_d)$

Concrete Example

If metadata is `Gender` (0 = Male, 1 = Female):

$$\mu_d = \gamma_0 + \gamma_1 \cdot \text{Gender}_d$$

For a female author ($\text{Gender}_d = 1$):

$$\mu_d = \gamma_0 + \gamma_1$$

The coefficient γ_1 captures how gender affects expected topic proportions!

5.3 Categorical Variables as Covariates

One-Hot Encoding for Covariates

When covariates are categorical (like author name), they become multiple coefficients:

$$X_d\gamma = \gamma_0 + \underbrace{\gamma_1 \cdot \mathbf{1}[\text{Amanda}] + \gamma_2 \cdot \mathbf{1}[\text{Douglas}] + \dots}_{\text{One-hot encoded author}}$$

Each category gets its own coefficient in γ.

5.4 The Covariance Matrix

Topic Correlations

The covariance matrix Σ captures correlations between topics:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots \\ \sigma_{12} & \sigma_2^2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

- Diagonal: Variance of each topic's prevalence
- Off-diagonal: Correlations between topics

These are **estimated from data**, not hyperparameters.

6 STM Estimation: The EM Algorithm

Overview

STM uses the Expectation-Maximization (EM) algorithm because topic assignments are latent (unobserved) variables.

6.1 Why EM is Needed

Important

We observe documents but **not**:

- Which topic generated each word (z_n)
- True topic proportions (θ_d)
- Topic-word distributions (β_k)

Since z_n is unobserved, we can't directly compute the likelihood.

6.2 EM Algorithm Steps

EM Algorithm

E-step (Expectation):

- Given current parameters, compute expected values of latent variables
- Calculate $E[\log P(\text{data}, z|\theta)]$

M-step (Maximization):

- Maximize expected log-likelihood with respect to parameters
- Update γ, Σ, β

Iterate until convergence.

Non-uniqueness and Multiple Runs

EM may converge to different local optima. **Best practice**:

1. Run STM multiple times with different initializations
2. Compare coherence and exclusivity for each run
3. Select the best model

The STM package does this automatically by default.

7 STM Implementation in R

Overview

STM is primarily implemented in R. The `stm` package is highly reliable and widely used in social science research.

7.1 Basic Workflow

```r
# Install and load
install.packages("stm")
library(stm)

# Create documents and metadata
documents <- c(
  "cats are wonderful pets",
  "cats enjoy climbing trees",
  # ... more documents by author 1
  "dogs love running outside",
  "dogs are loyal companions"
  # ... more documents by author 2
)

meta <- data.frame(
  author = c(rep("author1", 8), rep("author2", 8))
)

# Preprocess text
processed <- textProcessor(
  documents,
  metadata = meta,
  lowercase = TRUE,
  removestopwords = TRUE,
  removenumbers = TRUE,
  removepunctuation = TRUE,
  stem = TRUE
)

# Prepare documents
out <- prepDocuments(
  processed$documents,
  processed$vocab,
  processed$meta
)

# Fit STM
stm_model <- stm(
  documents = out$documents,
  vocab = out$vocab,
  K = 5,                          # Number of topics
  prevalence = ~ author,          # Covariates
  data = out$meta,
  max.em.its = 100,
  init.type = "Spectral"
)
```

Listing 1: STM Workflow in R

7.2 The Formula Interface

R Formula Notation

The `prevalence` formula specifies covariates:

- ∼ `author`: Intercept + author effect
- ∼ `author + date`: Multiple covariates
- ∼ `author * date`: Interaction effects

The tilde (∼) notation is standard R regression syntax.

7.3 Preprocessing Details

Document Removal During Preprocessing

`textProcessor` may remove:

- Stop words
- Numbers
- Punctuation
- Infrequent terms

If a document becomes **empty** after preprocessing, it's removed along with its metadata row. That's why we use `out$meta` instead of the original metadata!

8 Analyzing STM Results

8.1 Viewing Topic Summaries

```
# Summary of topics
summary(stm_model)

# Plot expected topic proportions
plot(stm_model, type = "summary", n = 5)
```

Listing 2: Summarize and plot topics

8.2 Interpreting Topics

Important

Best Practice: Don't rely solely on top words to interpret topics. Instead, examine **documents with highest topic prevalence**.
Top words may be:

- Common across many topics (e.g., "Australia" in Australian news)
- Stemmed and hard to interpret
- Ambiguous out of context

```r
# Find documents most associated with each topic
findThoughts(stm_model, texts = documents, n = 3, topics = 1:5)
```

Listing 3: Find representative documents

8.3 Estimating Covariate Effects

estimate Effect Function

To understand how covariates affect topic prevalence, use `estimateEffect`:

1. Sample from posterior distribution of θ
2. Regress sampled prevalences on covariates
3. Compute confidence intervals

```r
# Estimate effects for all topics
effects <- estimateEffect(
  1:5 ~ author,                    # Topics ~ covariates
  stmobj = stm_model,
  metadata = out$meta,
  uncertainty = "Global"
)

# Plot difference between authors
plot(effects,
  covariate = "author",
  topics = 1:5,
  model = stm_model,
  method = "difference",
  cov.value1 = "author2",          # On the right
  cov.value2 = "author1",          # Subtracted
  main = "Effect of Author on Topic Prevalence"
)
```

Listing 4: Estimate and plot effects

8.4 Time Series of Topic Prevalence

```r
# If date is a covariate
effects_time <- estimateEffect(
  1:5 ~ date,
  stmobj = stm_model,
  metadata = out$meta
)

plot(effects_time,
  covariate = "date",
  topics = 1:5,
  model = stm_model,
  method = "continuous",
  xlab = "Date",
  main = "Topic Prevalence Over Time"
)
```

Listing 5: Topic prevalence over time

Interpreting Time Plots

The plot shows:

- Expected topic prevalence (line)
- Confidence intervals (shaded region)
- Upward trend: topic becoming more prevalent
- Downward trend: topic becoming less prevalent

This assumes a **linear** trend. For non-linear patterns, use date as categorical or add polynomial terms.

9 Selecting the Number of Topics

Overview

Choosing K (number of topics) requires running STM multiple times and comparing performance metrics.

9.1 Using searchK

```r
# Search across different numbers of topics
k_search <- searchK(
  documents = out$documents,
  vocab = out$vocab,
  K = 2:10,                           # Range of K to try
  prevalence = ~ author,
  data = out$meta,
  init.type = "Spectral"
)

# Plot diagnostics
plot(k_search)
```

Listing 6: Search for optimal K

9.2 Coherence vs Exclusivity Trade-off

Model Selection Criteria

Semantic Coherence: Are top words within a topic co-occurring in documents?
Exclusivity: Are top words unique to each topic?
These often trade off:

- More topics ⇒ Higher exclusivity, lower coherence
- Fewer topics ⇒ Higher coherence, lower exclusivity

```r
# Extract metrics
coherence <- k_search$results$semcoh
exclusivity <- k_search$results$exclus

# Create comparison data frame
```

```r
metrics <- data.frame(
  K = 2:10,
  coherence = coherence,
  exclusivity = exclusivity
)

# Plot
library(ggplot2)
ggplot(metrics, aes(x = exclusivity, y = coherence, label = K)) +
  geom_point() +
  geom_text(nudge_x = 0.01) +
  labs(x = "Exclusivity", y = "Semantic Coherence",
       title = "Coherence vs Exclusivity by Number of Topics") +
  theme_minimal()
```

Listing 7: Extract and plot coherence/exclusivity

Important

Selection Strategy:

1. Rescale both metrics to $[0, 1]$
2. Compute average of rescaled metrics
3. Choose K that maximizes the average
4. Alternatively: visual inspection—pick the point closest to the upper-right corner

10 Real-World Application: Student Evaluations

Overview

A published study analyzed 11 years of student evaluations at Harvard using STM, discovering how topic prevalence varies with instructor gender and department.

10.1 Study Design

Student Evaluation Study

Data: 1 million student evaluations
Covariates:

- Instructor gender
- Instructor age
- Academic division
- Course type

Number of topics: 11 (selected via coherence/exclusivity)

10.2 Key Findings

Important

Gender Differences in Topic Prevalence:
When students discuss **female** instructors, they more often mention:

- "Caring, enthusiastic instructor"
- "Facilitates effective discussion"
- "Nice feedback"

When students discuss **male** instructors, they more often mention:

- "Lectures are interesting and relevant"
- "Uses humor effectively"

These patterns persisted even after controlling for department and course type—suggesting potential **student bias**.

10.3 Division-Level Patterns

Topic Variation by Academic Division

Sciences: "Explains complex concepts effectively" (high prevalence)
Humanities: "Facilitates effective discussions" (high prevalence)
Freshman Seminars: "Positive timely feedback" (high prevalence)
These differences reflect genuine pedagogical differences across disciplines.

10.4 Practical Implications

Why This Matters

If student evaluations show systematic biases:

- Promotion decisions may be affected
- Tenure reviews could be biased
- Adjustments might be needed when interpreting evaluations

STM allows researchers to **quantify** these effects and test their significance.

11 Advanced STM Features

11.1 Topic Correlations

```
# Plot topic correlations
topicCorr(stm_model, method = "simple")
```

Listing 8: Visualize topic correlations

This shows which topics tend to co-occur within documents.

11.2 Selecting Among Multiple Runs

```r
# searchK already runs multiple times internally
# To manually select the best model:
best_model <- selectModel(
  documents = out$documents,
  vocab = out$vocab,
  K = 5,
  prevalence = ~ author,
  data = out$meta,
  runs = 20                      # Number of runs
)

# Select based on exclusivity/coherence
plotModels(best_model)
```

Listing 9: Select best model from multiple runs

12 One-Page Summary

Summary

Structural Topic Modeling (STM) extends LDA by incorporating document meta-data.

Key Differences from LDA:

- LDA: $\theta_d \sim \text{Dirichlet}(\alpha)$
- STM: $\theta_d \sim \text{Logistic-Normal}(X_d\gamma, \Sigma)$

STM Generation Process:

1. Compute $\mu_d = X_d\gamma$ (linear function of covariates)
2. Draw $\eta_d \sim \mathcal{N}(\mu_d, \Sigma)$
3. Apply softmax: $\theta_d = \text{softmax}(\eta_d)$
4. Generate words as in LDA

Why Use STM?:

- Incorporates metadata (author, date, source)
- More accurate topic assignments
- Test hypotheses about covariate effects
- Track topic prevalence over time

R Implementation:

1. `textProcessor()`: Preprocess documents
2. `prepDocuments()`: Prepare for modeling
3. `stm()`: Fit the model
4. `estimateEffect()`: Analyze covariate effects

5. `searchK()`: Find optimal number of topics

Model Selection:

- Balance **coherence** (words co-occur) and **exclusivity** (topics distinct)
- Run multiple times, select best via metrics
- Rescale metrics to $[0, 1]$, maximize average

Best Practices:

- Interpret topics via representative documents, not just top words
- Use `out$meta` after preprocessing (some rows removed)
- Multiple runs are essential due to non-uniqueness

13 Glossary

Key Terms

- **STM**: Structural Topic Modeling—LDA extension with covariates
- **Metadata/Covariates**: Document-level information (author, date, source)
- **Logistic Normal**: Distribution for generating topic proportions in STM
- **Prevalence**: Expected proportion of a topic in documents
- **EM Algorithm**: Expectation-Maximization for latent variable models
- **E-step**: Compute expected log-likelihood given current parameters
- **M-step**: Maximize expected log-likelihood to update parameters
- **Posterior distribution**: Distribution of parameters after observing data
- **estimateEffect**: STM function to analyze covariate effects
- **searchK**: STM function to find optimal number of topics
- **selectModel**: STM function to choose best run
- **findThoughts**: STM function to find representative documents
- **Coherence**: Metric for within-topic word co-occurrence
- **Exclusivity**: Metric for between-topic word distinctiveness
- **textProcessor**: STM preprocessing function
- **prepDocuments**: STM document preparation function