CSCI E-89B Introduction to Natural Language Processing

Harvard Extension School

Dmitry Kurochkin

Fall 2025
Lecture 7

Contents

Contents

Introduction to Topic Modeling

- **Topic Modeling:** A type of statistical model used to identify abstract (latent) topics within a collection of documents.
- **Goal:** Discover hidden thematic structures in large textual datasets.
  - **Hidden Themes:** Uncover underlying patterns or themes not explicitly stated in text data.
  - **Automatic Identification:** Use computational methods to detect these themes without manual intervention.
  - **Large Datasets:** Essential for analyzing vast amounts of text, making sense of content that is too large for manual review.
- **Methods:** Common approaches include LDA, NMF, and STM
- **Benefits:** Allows for better document organization, understanding, and retrieval.
- **Multi-topic Representation:** Recognizes that documents often contain multiple overlapping topics.
  - LDA models documents as mixtures of topics, providing a nuanced representation of text data.
  - This approach captures the complexity of real-world documents that typically span various thematic areas.

Advantages Over Traditional Clustering

- **Overlapping Topics:** Unlike clustering, which assigns each document to a single cluster, LDA reflects the real-world scenario where documents can belong to multiple topics.
- **Probabilistic Approach:** LDA uses probabilistic distributions, offering a flexible way to model topic proportions within documents.
- **Rich Semantic Insights:** Offers deeper insights into the thematic distribution of words and documents that clustering methods like k-means might miss.

Contents

Latent Dirichlet Allocation (LDA)

- **Concept:** A generative probabilistic model for collections of discrete data.
- **Mathematical Framework:**
  - Determine the number of words N in a document:
    - ★ $N \sim \text{Poisson}(\xi)$: Document length is sampled from a Poisson distribution.
  - Each document is a mixture of K latent topics:
    - ★ $\theta_m \sim \text{Dirichlet}(\alpha)$: Topic distribution for each document m.
    - ★ For each word $w_n (n = 1, 2, \ldots, N)$ in document m:
      $z_n \sim \text{Multinomial}(\theta_m)$: Select a topic.
      $w_n \sim \text{Multinomial}(\beta_{z_n})$: Generate a word from the topic.
- **Components:**
  - Documents are distributions over topics.
  - Topics are distributions over words.
- **Flexibility:** Captures multi-topic nature of text for nuanced overlaps.

Reference: Blei DM, Ng A, Jordan M (2003). "Latent Dirichlet Allocation." Journal of Machine Learning Research, 3, 993–1022.

Applications of LDA

- **Document Classification:** Topics identified by LDA help categorize documents into predefined classes by analyzing their thematic content.
- **Recommendation Systems:** Utilize topic distributions to suggest related articles, products, or content by finding documents with similar topic profiles.
- **Content Analysis:** Extracts key themes from large volumes of text to summarize and gain insights, useful in media monitoring and trend analysis.
- **Search and Retrieval:** Enhances the accuracy of search engines by indexing documents according to discovered topics rather than relying solely on keyword matching.
- **Social Media Analysis:** Identifies trending topics and public sentiments across social media platforms by examining patterns in user-generated content.

Contents

LDA Implementation in Python

```
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import CountVectorizer

# Sample documents
documents = [
    "Cats purr gently and climb high trees. Chasing a mouse is fun for cats.",
    "Independent creatures, cats enjoy solitude and love their nap time.",
    "Dogs bark loudly at strangers and fetch sticks with enthusiasm.",
    "Loyal dogs accompany humans on hikes and love to chase balls.",
    "When the energetic dog spotted a squirrel, it barked energetically.",
    "A purring cat climbed the bookshelf, watching over the room.",
    "Regularly, dogs enjoy long walks, sniffing and exploring their environment.",
    "Cats meow softly and purr when content, loving to stretch in the sun."
]

# Generate the matrix of token counts
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(documents)

# Train the LDA model
lda = LatentDirichletAllocation(n_components=2, random_state=0)
lda.fit(X)

# Display topics
for index, topic in enumerate(lda.components_):
    print(f"Topic {index + 1}:")
    print([vectorizer.get_feature_names_out()[i] for i in topic.argsort()[:-6:-1]])

Topic 1:
['dogs', 'enjoy', 'long', 'walks', 'exploring']
Topic 2:
['cats', 'purr', 'love', 'trees', 'mouse']
```

Contents

LDA Implementation in R

```r
# Install and load necessary libraries
# install.packages("tm")           # Uncomment to install
# install.packages("topicmodels")  # Uncomment to install

# Load necessary libraries
library(tm)
library(topicmodels)

# Sample documents
documents <- c(
  "Cats purr gently and climb high trees. Chasing a mouse is fun for cats.",
  "Independent creatures, cats enjoy solitude and love their nap time.",
  "Dogs bark loudly at strangers and fetch sticks with enthusiasm.",
  "Loyal dogs accompany humans on hikes and love to chase balls.",
  "When the energetic dog spotted a squirrel, it barked energetically.",
  "A purring cat climbed the bookshelf, watching over the room.",
  "Regularly, dogs enjoy long walks, sniffing and exploring their environment.",
  "Cats meow softly and purr when content, loving to stretch in the sun."
)

# Create a Text Corpus
corpus <- Corpus(VectorSource(documents))

# Pre-process the data
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stripWhitespace)

# Create a Document-Term Matrix
dtm <- DocumentTermMatrix(corpus)
```

LDA Implementation in R (Continued)

```
# Train the LDA model with 2 topics
lda_model <- LDA(dtm, k = 2, control = list(seed = 1234))

# Output the topics
terms_lda <- terms(lda_model, 5)  # Top 5 terms for each topic

# Print each topic
cat("LDA Model Topics:\n\n")
for (topic_idx in 1:ncol(terms_lda)) {
  cat(paste("Topic", topic_idx, ":", paste(terms_lda[, topic_idx], collapse = ", "), "\n"))
}
```

```
Topic 1 : dogs, enjoy, love, bark, enthusiasm
Topic 2 : cats, purr, chasing, climb, fun
```

Contents

Non-Negative Matrix Factorization (NMF)

- **Concept:**
  - NMF decomposes a non-negative matrix V (e.g., document-term matrix) into two non-negative matrices, W and H.
  - This decomposition captures the latent structure in data, associating terms with topics and topics with documents.
- **Mathematical Formulation:**
  - Given $V \approx WH$, where V is the document-term matrix.
  - W approximates the document-topic relationships and H the topic-term relationships.
- **Conditions:**
  - Non-negativity ensures interpretability, as elements represent weights or counts.
  - Useful in scenarios where data is inherently non-negative, like word counts or pixel values.

Contents

Example of NMF from Text Data

- **Text Data:**
  - Doc 1: "cats meow"
  - Doc 2: "dogs bark"
  - Doc 3: "cats purr, dogs growl"
- **Vocabulary:** "cats", "dogs", "meow", "bark", "purr", "growl"
- **Document-Term Matrix V:**

$$V = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

- **NMF Decomposition:** $V \approx WH$

$$W \approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.5 & 0.5 \end{bmatrix}, \quad H \approx \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

- **Interpretation:**
  - W: Illustrates document affinity towards two latent topics.
  - H: Displays term association with these latent topics.

Contents

Applications and Benefits of NMF

- **Use Cases:**
  - **Clustering:** Ideal for grouping similar items in unstructured data, such as text or images, revealing hidden patterns.
  - **Dimensionality Reduction:** Simplifies complex datasets (e.g., image or text collections) without significant loss of important information, aiding in visualization and analysis.
  - **Feature Extraction:** Useful for extracting meaningful features that represent underlying data structures, crucial in bioinformatics and document analysis.
- **Benefits:**
  - **Interpretability:** Provides intuitive, parts-based data representations that are easy to understand and explain.
  - **Versatility:** Enhances tasks such as document classification, recommendation systems, and anomaly detection in unsupervised settings.
  - **Sparsity and Efficiency:** Encourages sparse solutions that are computationally efficient to store and process.

Comparison with LDA and Considerations

- **Comparison with LDA:**
  - **Algorithmic Approach:** NMF uses a deterministic, algebraic approach, making it simpler and often faster to compute compared with the probabilistic, MCMC-based methods in LDA.
  - **Result Interpretation:** Outputs of NMF are more straightforward to interpret due to non-negativity constraints, whereas LDA might require more context to fully understand topic distributions.
  - **Flexibility vs. Depth:** While NMF is highly flexible and can be applied across varied datasets, LDA offers a deeper statistical grounding with its probabilistic assumptions, potentially providing more nuanced insights in certain contexts.
- **Considerations:**
  - Choosing the number of components can significantly influence results; it often requires domain knowledge or empirical testing.
  - NMF is less robust to noise compared to some other methods, so data pre-processing is critical.

Contents

NFM Implementation in Python

```
from sklearn.decomposition import NMF
from sklearn.feature_extraction.text import CountVectorizer

# Sample documents
documents = [
    "Cats purr gently and climb high trees. Chasing a mouse is fun for cats.",
    "Independent creatures, cats enjoy solitude and love their nap time.",
    "Dogs bark loudly at strangers and fetch sticks with enthusiasm.",
    "Loyal dogs accompany humans on hikes and love to chase balls.",
    "When the energetic dog spotted a squirrel, it barked energetically.",
    "A purring cat climbed the bookshelf, watching over the room.",
    "Regularly, dogs enjoy long walks, sniffing and exploring their environment.",
    "Cats meow softly and purr when content, loving to stretch in the sun."
]

# Generate the matrix of token counts
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(documents)

# Train the NMF model
nmf_model = NMF(n_components=2, random_state=0)
W = nmf_model.fit_transform(X)
H = nmf_model.components_

# Display topics
feature_names = vectorizer.get_feature_names_out()
for topic_idx, topic in enumerate(H):
    print(f"Topic {topic_idx + 1}:")
    print(" ".join([feature_names[i] for i in topic.argsort()[:-6:-1]]))

Topic 1: cats purr high trees climb
Topic 2: dogs love enjoy humans loyal
```