**AIM: Perform topic modeling using LDA (Latent Dirichlet Allocation).**

**DESCRIPTION:**

Topic modeling is an unsupervised machine learning technique used to discover hidden themes or topics in a collection of documents.  
Latent Dirichlet Allocation (LDA) is one of the most popular probabilistic models for topic modeling. It assumes that:

* Each document is a mixture of several topics.
* Each topic is a distribution over words.

LDA helps in automatic organization, summarization, and understanding of large text corpora.

**Problem Definition**

**Given a set of documents:**

Doc1: "Machine learning is a subfield of AI..."

Doc2: "Deep learning uses multiple layers..."

Doc3: "Natural language processing helps computers understand human language..."

The goal is to discover latent topics such as:

Topic 1: machine, learning, data, algorithm

Topic 2: deep, layers, neural, model

Topic 3: language, processing, natural, human

Each document can belong to multiple topics with different probabilities.

**LDA**

LDA is a generative probabilistic model based on Dirichlet distributions:

1. Assume K topics exist in the corpus.
2. For each document:
   * A distribution over topics is drawn from a Dirichlet prior (alpha).
   * Each word is generated by first choosing a topic (based on document’s topic distribution) and then choosing a word from the topic’s word distribution (Dirichlet prior beta).
3. Using the observed words in documents, LDA infers the hidden topic structure:
   * Document-topic distributions (theta)
   * Topic-word distributions (phi)

**Key Components of LDA**

| **Component** | **Description** |
| --- | --- |
| Topic | A cluster of words that frequently occur together |
| Document-topic distribution | Probability of each topic in a document (theta) |
| Topic-word distribution | Probability of each word in a topic (phi) |
| Hyperparameters | alpha (controls sparsity of document-topic distribution), beta (controls sparsity of topic-word distribution) |

**Steps in LDA Topic Modeling**

1. Preprocessing
   * Tokenization (split text into words)
   * Lowercasing
   * Stopword removal
   * Optional: Lemmatization or stemming
2. Bag-of-Words Representation
   * Convert each document into a vector of word counts (or TF-IDF)
3. Choose Number of Topics (K)
   * Usually based on domain knowledge or trial-and-error
4. Train LDA Model
   * Use Gibbs sampling or variational inference to learn:
     + Topic-word distributions
     + Document-topic distributions
5. Evaluate and Interpret Topics
   * Examine top words per topic
   * Optionally calculate topic coherence for validation
6. Use Cases
   * Discover hidden patterns in text
   * Document clustering
   * Recommendation systems
   * Summarization

**Advantages of LDA**

* Unsupervised — no labeled data needed
* Probabilistic — allows documents to belong to multiple topics
* Scalable to large corpora
* Helps reduce dimensionality for text analysis

**Limitations**

* Choice of K (number of topics) is often subjective
* Requires large corpora for meaningful topics
* Sensitive to preprocessing
* May produce overlapping or noisy topics

**Applications**

* News article categorization
* Research paper clustering
* Social media trend analysis
* Customer review analysis
* Search and recommendation engines

| **Concept** | **Description** |
| --- | --- |
| Input | A collection of text documents |
| Output | K latent topics (word clusters) + document-topic distribution |
| Core Idea | Each document is a mixture of topics; each topic is a mixture of words |
| Algorithm | Probabilistic generative model using Dirichlet distributions |

**MODULES TO INSTALL IN RUN COMMAND:**

pip install scikit-learn nltk

**PROGRAM:**

import random

from collections import defaultdict

# ---------------------------------------------------------------

# Step 1: Sample Documents

documents = [

"Machine learning is a field of artificial intelligence that uses statistical techniques to give computer systems the ability to learn from data.",

"Natural language processing is a subfield of AI concerned with the interactions between computers and human language.",

"Deep learning is a class of machine learning algorithms that use multiple layers to progressively extract higher-level features from raw input.",

"Reinforcement learning is a type of machine learning where agents learn how to behave in an environment by performing actions.",

"The study of algorithms that can automatically improve through experience is central to machine learning."

]

# ---------------------------------------------------------------

# Step 2: Preprocessing (lowercase, remove punctuation, split words)

def preprocess(doc):

doc = doc.lower()

for ch in ['.', ',', '!', '?', ';', ':']:

doc = doc.replace(ch, '')

words = doc.split()

# basic stopword list

stopwords = set([

"is", "a", "the", "of", "and", "to", "from", "that", "in", "as", "for", "with", "how", "an", "be", "by", "on", "it"

])

words = [w for w in words if w not in stopwords]

return words

texts = [preprocess(doc) for doc in documents]

# Build vocabulary

vocab = sorted(set(word for doc in texts for word in doc))

vocab\_index = {word: i for i, word in enumerate(vocab)}

print(f"\nNumber of documents: {len(texts)}")

print(f"Vocabulary size: {len(vocab)}")

# ---------------------------------------------------------------

# Step 3: Initialize LDA Parameters

K = 3 # number of topics

alpha = 0.1

beta = 0.01

iterations = 100

# Initialize counts

doc\_topic\_counts = [defaultdict(int) for \_ in texts]

topic\_word\_counts = [defaultdict(int) for \_ in range(K)]

topic\_counts = [0] \* K

document\_lengths = [len(doc) for doc in texts]

# Random initial topic assignment

Z = []

for d, doc in enumerate(texts):

current\_doc\_topics = []

for word in doc:

topic = random.randrange(K)

current\_doc\_topics.append(topic)

doc\_topic\_counts[d][topic] += 1

topic\_word\_counts[topic][word] += 1

topic\_counts[topic] += 1

Z.append(current\_doc\_topics)

# ---------------------------------------------------------------

# Step 4: Gibbs Sampling

for it in range(iterations):

for d, doc in enumerate(texts):

for i, word in enumerate(doc):

topic = Z[d][i]

# Remove current word/topic assignment

doc\_topic\_counts[d][topic] -= 1

topic\_word\_counts[topic][word] -= 1

topic\_counts[topic] -= 1

# Compute probabilities for each topic

p = []

for k in range(K):

p\_topic = (doc\_topic\_counts[d][k] + alpha) / (document\_lengths[d] - 1 + K \* alpha)

p\_word = (topic\_word\_counts[k][word] + beta) / (topic\_counts[k] + len(vocab) \* beta)

p.append(p\_topic \* p\_word)

# Normalize

total = sum(p)

p = [val / total for val in p]

# Sample new topic

r = random.random()

cumulative = 0.0

new\_topic = None

for k, prob in enumerate(p):

cumulative += prob

if r < cumulative:

new\_topic = k

break

# Assign new topic

Z[d][i] = new\_topic

doc\_topic\_counts[d][new\_topic] += 1

topic\_word\_counts[new\_topic][word] += 1

topic\_counts[new\_topic] += 1

if (it + 1) % 20 == 0:

print(f"Iteration {it+1} completed...")

# ---------------------------------------------------------------

# Step 5: Display Top Words per Topic

def top\_words(topic\_word\_counts, num\_words=5):

for k, word\_counts in enumerate(topic\_word\_counts):

sorted\_words = sorted(word\_counts.items(), key=lambda x: x[1], reverse=True)

top = [w for w, c in sorted\_words[:num\_words]]

print(f"Topic {k+1}: {', '.join(top)}")

print("\n--- TOPICS DISCOVERED BY LDA ---")

top\_words(topic\_word\_counts, num\_words=5)

**OUTPUT:**

"C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\.venv\Scripts\python.exe" "C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\TOKENIZATION.py"

Number of documents: 5

Vocabulary size: 51

Iteration 20 completed...

Iteration 40 completed...

Iteration 60 completed...

Iteration 80 completed...

Iteration 100 completed...

--- TOPICS DISCOVERED BY LDA ---

Topic 1: algorithms, intelligence, ability, layers, progressively

Topic 2: language, natural, subfield, ai, processing

Topic 3: learning, machine, learn, class, extract

Process finished with exit code 0

**RESULT:**

Hence, implemented topic modeling using LDA (Latent Dirichlet Allocation) successfully.