

Data Science Lab

Student PRN:123M1H041

Student Name:DARSHAN SHASHIKANT PATHAK

Submission Date:07/10/2024

Advanced Data Science Lab - Practical Assignment: 6

Assignment Based on Text Mining Algorithms

a) Document Classification Using Bag of Words and Naive Bayes:

Step 1: Problem Definition and Overview

Implementing a **Document Classification** system using the **Bag-of-Words (BoW)** model and a **Naive Bayes Classifier**. Our goal is to classify documents into predefined categories (in this case, "Sports" or "Politics").

What is Document Classification?

Document classification is a process where we assign a document to one or more categories based on its content. For instance, a news article about football would be categorized under "Sports," while an article about government policies would be categorized under "Politics."

```
In [1]: # Importing necessary Libraries
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
# Sample dataset (Text and Category)
```

```
df = pd.read_excel("documents.xlsx", index_col="Document ID")
df.head(10)
```

Out[1]:


	Text	Category
Document ID		
1	The team won the championship game last night.	Sports
2	The government passed a new law regarding taxes.	Politics
3	The player scored the winning goal in the finals.	Sports
4	The election results were announced yesterday.	Politics
5	He broke the record for most points in a season.	Sports
6	The president gave a speech about healthcare r...	Politics
7	The coach praised the team's performance after...	Sports
8	New policies were introduced to improve public...	Politics
9	She won a gold medal in the Olympic Games.	Sports
10	The senator discussed climate change in her la...	Politics

Step 2: Understanding the Bag-of-Words Model

What is Bag-of-Words (BoW)?

The Bag-of-Words model is a method for representing text data where each document is represented as a "bag" of its words, ignoring grammar and word order but keeping multiplicity. Essentially, it's a collection of words along with their frequencies.

- Each document is converted into a vector.
- Each position in the vector represents a word from the entire vocabulary (all unique words in all documents).
- The value at each position represents how many times that word appeared in the document.

 No description has been provided for this image

Step 3: Naive Bayes Classifier

What is the Naive Bayes Classifier?

The Naive Bayes classifier is a probabilistic machine learning model based on **Bayes' Theorem**. It's called "naive" because it assumes that all features (in this case, words in the documents) are independent of each other, which is a simplifying assumption that often works well in practice.

Bayes' Theorem is defined as:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

In this equation, using Bayes theorem, we can find the probability of A, given that B occurred. A is the hypothesis, and B is the evidence.

$P(B|A)$ is the probability of B given that A is True.

$P(A)$ and $P(B)$ are the independent probabilities of A and B.

Step 4: Preprocessing the Text

Before we can classify the documents, we need to preprocess the text. Text preprocessing includes:

1. **Tokenization:** Splitting sentences into individual words.
2. **Lowercasing:** Converting all words to lowercase.
3. **Removing Stop Words:** Words like "the", "is", "in" which do not provide meaningful information.
4. **Vectorization:** Converting text into numerical data.

Using the `CountVectorizer` from **Scikit-learn**, to convert text into a Bag-of-Words model.

```
In [2]: # Initializing the CountVectorizer with stop words removal
vectorizer = CountVectorizer(lowercase=True, stop_words='english')

# Converting the documents to a matrix of token counts (Bag-of-Words)
X = vectorizer.fit_transform(df["Text"])

# Printing the vocabulary and feature names
print("Vocabulary: ", vectorizer.vocabulary_)
print("Bag of Words Representation:\n", X.toarray())
```

Vocabulary: {'team': 311, 'won': 346, 'championship': 43, 'game': 123, 'night': 201, 'government': 129, 'passed': 215, 'new': 200, 'law': 168, 'regarding': 256, 'taxes': 310, 'player': 223, 'scored': 268, 'winning': 344, 'goal': 126, 'finals': 114, 'election': 95, 'results': 264, 'announced': 17, 'yesterday': 351, 'broke': 35, 'record': 251, 'points': 226, 'season': 269, 'president': 235, 'gave': 125, 'speech': 292, 'healthcare': 138, 'reform': 254, 'coach': 53, 'praised': 231, 'performance': 218, 'match': 180, 'policies': 227, 'introduced': 163, 'improve': 147, 'public': 243, 'transportation': 328, 'gold': 128, 'medal': 183, 'olympic': 203, 'games': 124, 'senator': 273, 'discussed': 83, 'climate': 51, 'change': 45, 'latest': 166, 'address': 5, 'marathon': 179, 'attracted': 24, 'thousands': 315, 'participants': 212, 'year': 350, 'voter': 338, 'turnout': 331, 'higher': 140, 'expected': 107, 'recent': 248, 'elections': 96, 'set': 278, 'world': 348, '100': 0, 'meters': 187, 'sprint': 296, 'city': 49, 'council': 69, 'approved': 19, 'budget': 37, 'education': 91, 'series': 275, 'begin': 28, 'week': 341, 'debate': 77, 'took': 319, 'place': 220, 'training': 325, 'upcoming': 335, 'triathlon': 329, 'competition': 58, 'prime': 236, 'minister': 188, 'measures': 182, 'combat': 54, 'inflation': 157, 'tennis': 314, 'tournament': 320, 'thrilling': 316, 'final': 113, 'voting': 340, 'regulations': 259, 'implemented': 145, 'ensure': 103, 'fairness': 109, 'athlete': 21, 'event': 105, 'aimed': 13, 'reducing': 253, 'carbon': 40, 'emissions': 98, 'congress': 61, 'local': 177, 'football': 118, 'celebrated': 42, 'victory': 337, 'fans': 110, 'opinion': 205, 'polls': 230, 'increasing': 152, 'support': 306, 'environmental': 104, 'completed': 59, 'races': 246, 'working': 347, 'plan': 221, 'job': 165, 'opportunities': 207, 'young': 352, 'people': 217, 'received': 247, 'award': 25, 'outstanding': 208, 'basketball': 27, 'legislation': 175, 'cybersecurity': 72, 'unanimously': 332, 'held': 139, 'stadium': 297, 'committee': 56, 'formed': 119, 'issues': 164, 'related': 260, 'immigration': 144, 'hat': 135, 'trick': 330, 'soccer': 289, 'mayor': 181, 'plans': 222, 'renovate': 262, 'parks': 211, 'considered': 62, 'greatest': 130, 'athletes': 22, 'time': 317, 'initiative': 159, 'aims': 15, 'increase': 150, 'registration': 258, 'adults': 7, 'parade': 210, 'downtown': 87, 'indicate': 155, 'shift': 279, 'access': 1, 'national': 195, 'gymnastics': 133, 'month': 191, 'trade': 323, 'agreements': 10, 'negotiated': 199, 'boost': 33, 'economic': 90, 'growth': 131, 'sports': 294, 'analysts': 16, 'worldwide': 349, 'proposed': 241, 'enhance': 101, 'data': 75, 'privacy': 237, 'protections': 242, 'consumers': 63, 'achieved': 3, 'personal': 219, 'best': 29, 'swimming': 308, 'weekend': 342, 'community': 57, 'leaders': 171, 'advocating': 9, 'better': 30, 'mental': 186, 'health': 137, 'services': 276, 'baseball': 26, 'playoffs': 225, 'bipartisan': 32, 'effort': 93, 'underway': 333, 'infrastructure': 158, 'needs': 198, 'country': 70, 'recognition': 250, 'contributions': 66, 'women': 345, 'advocacy': 8, 'initiatives': 160, 'launched': 167, 'promote': 239, 'civic': 50, 'engagement': 100, 'citizens': 48, 'named': 194, 'mvp': 193, 'leading': 172, 'improving': 148, 'signed': 285, 'participated': 213, 'international': 162, 'summer': 305, 'policy': 228, 'renewable': 261, 'energy': 99, 'sources': 290, 'lawmakers': 169, 'race': 245, 'officials': 202, 'meeting': 184, 'discuss': 82, 'security': 271, 'today': 318, 'league': 173, 'expansion': 106, 'panel': 209, 'experts': 108, 'conference': 60, 'secured': 270, 'impressive': 146, 'campaign': 38, 'financing': 115, 'debated': 78, 'led': 174, 'stunning': 303, 'id': 143, 'laws': 170, 'sparked': 291, 'controversy': 68, 'states': 299, 'recently': 249, 'trains': 326, 'daily': 74, 'prepare': 233, 'big': 31, 'study': 302, 'shows': 284, 'increased': 151, 'universal': 334, 'incredible': 154, 'skills': 288, 'field': 111, 'earned': 89, 'scholarship': 267, 'political': 229, 'predict': 232, 'significant': 286, 'changes': 46, 'demographics': 80, 'win': 343, 'town': 321, 'selected': 272, 'debates': 79, 'gun': 132, 'control': 67, 'continue': 65, 'divide': 86, 'goals': 127, 'tackle': 309, 'homelessness': 142, 'major': 178, 'cities': 47, 'regional': 257, 'nationals': 196, 'legislators': 176, 'pushing': 244, 'reforms': 255, 'funding': 121, 'elite': 97, 'forums': 120, 'transparency': 327, 'championships': 44, 'aim': 12, 'resources': 263, 'nationwide': 197, 'preparing': 234, 'intense': 161, 'playoff': 224, 'ahead': 11, 'addressing': 6, 'income': 149, 'inequality': 156, 'session': 277, 'olympics': 204, 'brought': 36, 'home': 141, 'silver': 287, 'de

```
velopment': 81, 'multiple': 192, 'records': 252, 'career': 41, 'shifting': 281, 'opinions': 206, 'voters': 339, 'teamwork': 313, 'straight': 300, 'increasingly': 153, 'contentious': 64, 'cycle': 73, 'aiming': 14, 'discussing': 84, 'participation': 214, 'passion': 216, 'drives': 88, 'day': 76, 'sentiment': 274, 'progressive': 238, 'clinched': 52, 'spot': 295, 'comeback': 55, 'campaigns': 39, 'shifted': 280, 'focus': 116, 'youth': 353, 'strategies': 301, 'achievements': 4, 'standards': 298, 'sport': 293, 'surrounding': 307, 'approaches': 18, 'trained': 324, 'hard': 134, 'teams': 312, 'shifts': 282, 'athleticism': 23, 'shone': 283, 'proposals': 240, 'enhancing': 102, 'educational': 92, 'headlines': 136, 'breaking': 34, 'following': 117, 'successful': 304, 'filled': 112, 'memorable': 185, 'moments': 190, 'discussions': 85, 'rights': 265, 'gain': 122, 'traction': 322, 'crucial': 71, 'accomplishments': 2, 'role': 266, 'model': 189, 'efforts': 94, 'urban': 336, 'areas': 20}
```

Bag of Words Representation:

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Explanation:

- **lowercase=True** : Ensures that all words are converted to lowercase before processing (default behavior).
- **stop_words='english'** : Removes common English stop words like "the", "is", etc.
- **fit_transform()** : This method both fits the vectorizer to the data and transforms the text into the Bag-of-Words model.

Creating WordCloud

```
In [3]: # Importing necessary Libraries
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Creating a word cloud from the text
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(' ')

# Displaying the word cloud using matplotlib
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off') # Hide the axes
plt.show()
```


Step 6: Training the Naive Bayes Model

Now that we have preprocessed the text and split the data, we can train the Naive Bayes model using **Multinomial Naive Bayes**, which is well-suited for text classification.

```
In [5]: # Importing the Multinomial Naive Bayes classifier
        from sklearn.naive_bayes import MultinomialNB

        # Initializing the classifier
        nb_classifier = MultinomialNB()

        # Train the classifier on the training data
        nb_classifier.fit(X_train, y_train)
```

```
Out[5]: ▼ MultinomialNB
        MultinomialNB()
```

Explanation:

- `MultinomialNB()` is used for classification with discrete features (like word counts from the BoW model).
- `fit()` trains the classifier using the training data.

Step 7: Making Predictions

Once the model is trained, we can use it to predict the categories of the test data.

```
In [6]: # Predicting the categories of the test set
        y_pred = nb_classifier.predict(X_test)

        # Printing predictions
        print("Predictions:", y_pred)
        print("Actual labels:", y_test)
```

```

Predictions: ['Politics ' 'Politics' 'Sports' 'Politics' 'Sports' 'Politics ' 'Sport
s'
'Sports ' 'Sports' 'Sports' 'Sports ' 'Sports ' 'Politics' 'Politics'
'Sports ' 'Sports' 'Sports' 'Politics' 'Sports' 'Politics']
Actual labels: Document ID
84      Politics
54      Politics
71      Sports
46      Politics
45      Sports
40      Politics
23      Sports
81      Sports
11      Sports
1       Sports
19      Sports
31      Sports
74      Politics
34      Politics
91      Sports
5       Sports
77      Sports
78      Politics
13      Sports
32      Politics
Name: Category, dtype: object

```

Step 8: Evaluating the Model

To measure how well our model performs, we'll calculate the accuracy. Accuracy is the percentage of correct predictions made by the model.

```

In [7]: # Importing accuracy_score to calculate the accuracy
        from sklearn.metrics import accuracy_score

        # Calculating the accuracy
        accuracy = accuracy_score(y_test, y_pred)

        print(f"Model Accuracy: {accuracy * 100:.2f}%")

```

Model Accuracy: 45.00%

Explanation:

- `accuracy_score()` compares the predicted labels with the actual labels and computes the accuracy of the model.

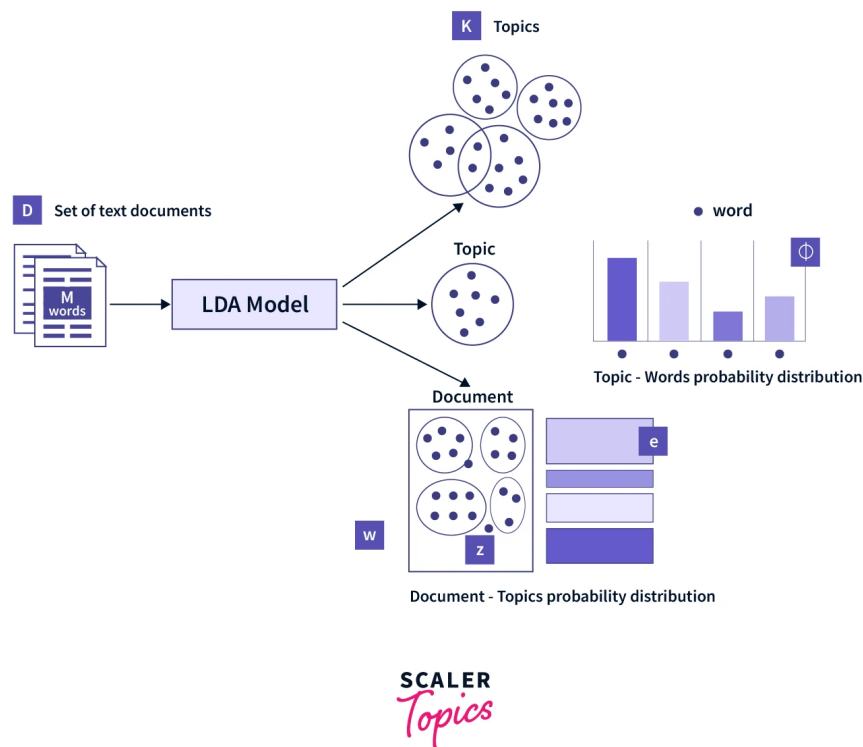
b) Implementing Topic Modeling using Latent Dirichlet Allocation (LDA):

What is Topic Modeling?

Topic modeling is a technique used to automatically discover the hidden topics that occur in a collection of documents. Each document can be seen as a mixture of various topics, and each topic is a distribution over words. Topic modeling algorithms try to reverse this process by recovering the topics from the words in the documents.

Step 1: Understanding Latent Dirichlet Allocation (LDA)

What is Latent Dirichlet Allocation?



LDA is a type of **unsupervised machine learning** algorithm that aims to uncover latent topics in a collection of text documents. It's based on a probabilistic model that assumes each document is a mixture of multiple topics, and each topic is a mixture of multiple words.

LDA uses two distributions:

1. **Document-Topic Distribution:** A probability distribution over topics for each document.
2. **Topic-Word Distribution:** A probability distribution over words for each topic.

Mathematically:

- $P(w|t)$ is the probability of word (w) given topic (t).
- $P(t|d)$ is the probability of topic (t) given document (d).

LDA assumes the following:

- Each document is a mixture of topics.
- Each word in a document is attributable to one of the document's topics.

Key Terminology:

- **Topic:** A distribution over a fixed vocabulary.
- **Dirichlet Distribution:** A distribution used as a prior for the per-document topic distributions.
- **Latent Variable:** In this context, a topic that is "hidden" and must be inferred from the data.

Step 2: Dataset Preparation

```
In [8]: df = pd.read_excel("lda_dataset.xlsx")  
  
df.head(10)
```

```
Out[8]:
```

	Document	Category
0	The game was thrilling and the team gave their...	Sports
1	A new tax bill has been passed in the parliament.	Politics
2	The government announced new healthcare reforms.	Politics
3	The player scored a hat-trick in the final match.	Sports
4	There was a discussion in the senate about the...	Politics
5	The football match was exhilarating, with both...	Sports
6	The president signed a bill for economic growth.	Politics
7	Many supporters celebrated the team's victory.	Sports
8	The debate on environmental law continues in t...	Politics
9	A new basketball court was inaugurated in the ...	Sports

Step 3: Preprocessing the Text Data

Before applying LDA, we must preprocess the text data. This includes:

1. **Tokenization:** Breaking documents into individual words.
2. **Lowercasing:** Converting words to lowercase.
3. **Removing stop words:** Words like "the", "is", "and", which do not contribute to topic identification.
4. **Stemming/Lemmatization:** Reducing words to their root form (optional but often useful).

```
In [9]: # Importing necessary Libraries
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import pandas as pd

# Step 1: Preprocessing the text (convert to lowercase, remove stop words)
vectorizer = CountVectorizer(stop_words='english')

# Step 2: Convert documents into a matrix of token counts
X = vectorizer.fit_transform(df["Document"])

# Display the vocabulary and Bag of Words representation
print("Vocabulary:", vectorizer.get_feature_names_out())
print("Bag of Words Representation:\n", X.toarray())
```

Vocabulary: ['according' 'addressed' 'advanced' 'agreement' 'announced' 'athlete' 'athletics' 'await' 'awarded' 'basketball' 'best' 'boost' 'businesses' 'career' 'celebrated' 'championship' 'change' 'city' 'climate' 'coach' 'committee' 'congress' 'considering' 'continues' 'corporate' 'court' 'cricket' 'cup' 'cuts' 'debate' 'debated' 'debates' 'decide' 'dedication' 'defense' 'discussed' 'discussion' 'dominated' 'eagerly' 'economic' 'education' 'efforts' 'end' 'enthusiasts' 'environmental' 'event' 'excellent' 'excited' 'exhilarating' 'fans' 'fantastic' 'fighting' 'final' 'finals' 'finance' 'finish' 'focus' 'focused' 'football' 'forum' 'forward' 'funding' 'game' 'gave' 'giving' 'government' 'grand' 'growing' 'growth' 'hard' 'hat' 'healthcare' 'held' 'highlights' 'hour' 'house' 'implemented' 'importance' 'improve' 'inaugurated' 'increasing' 'industries' 'infrastructure' 'initiative' 'injury' 'intense' 'international' 'interrupted' 'introduced' 'introducing' 'launch' 'law' 'lawmakers' 'league' 'led' 'left' 'local' 'looking' 'major' 'match' 'mayor' 'medals' 'meeting' 'minister' 'mixed' 'mvp' 'nation' 'national' 'new' 'officials' 'ongoing' 'outstanding' 'overhaul' 'parade' 'parliament' 'passed' 'performance' 'performances' 'plans' 'played' 'player' 'players' 'policies' 'policy' 'political' 'popularity' 'praised' 'president' 'prime' 'public' 'rain' 'reactions' 'received' 'recent' 'referendum' 'reform' 'reforms' 'regarding' 'region' 'regulations' 'remarkable' 'report' 'resumed' 'retirement' 'returned' 'road' 'saw' 'scheme' 'scored' 'season' 'senate' 'set' 'signed' 'small' 'speech' 'star' 'strategy' 'stricter' 'success' 'successful' 'support' 'supporters' 'tax' 'team' 'teams' 'teamwork' 'teamâ' 'tennis' 'thrilling' 'till' 'title' 'tournament' 'trade' 'trained' 'training' 'trick' 'upcoming' 'victory' 'vote' 'welfare' 'win' 'won' 'working' 'works' 'world' 'year']

Bag of Words Representation:

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Explanation:

- `CountVectorizer(stop_words='english')` : This removes common English stop words.
- `fit_transform(documents)` : This creates the Bag-of-Words (BoW) representation, converting the text into a matrix of word counts.

Step 4: Applying Latent Dirichlet Allocation (LDA)

Once the documents are vectorized, we can apply LDA to extract the hidden topics. LDA will attempt to discover a specified number of topics within the documents.

Specifying the number of topics to extract. For this example, let's assume we want to discover **2 topics**.

```
In [10]: # Number of topics
n_topics = 2

# Step 3: Apply LDA for Topic Modeling
lda = LatentDirichletAllocation(n_components=n_topics, random_state=42)
lda.fit(X)

# Step 4: Display the topics
topics = lda.components_

# Show top words per topic
n_top_words = 5
feature_names = vectorizer.get_feature_names_out()
```

Explanation:

- `LatentDirichletAllocation(n_components=n_topics)` : Initializes the LDA model where `n_components` is the number of topics we want to discover.
- `lda.fit(X)` : Fits the LDA model to the Bag-of-Words matrix.
- `components_` : This returns the topic-word distribution matrix. We print the top words that contribute the most to each topic.

```
In [11]: for topic_idx, topic in enumerate(topics):
print(f"\nTopic {topic_idx + 1}:")
print(" ".join([feature_names[i] for i in topic.argsort()[: -n_top_words - 1: -1]]))
```

Topic 1:
new tax reforms government economic

Topic 2:
team star game basketball player

Interpretation:

- **Topic 1** seems to be related to **politics or government affairs** (words like "law", "tax", "reform").
- **Topic 2** seems to be related to **sports** (words like "team", "game", "player").

Step 5: Inference - Document-Topic Distribution

LDA can also give us the **document-topic distribution**, which tells us the probability of each document belonging to each of the topics.

```
In [12]: # Step 5: Get document-topic distribution
doc_topic_dist = lda.transform(X)
```

Explanation:

- `lda.transform(X)` : This returns the probability distribution over topics for each document.
- We then create a pandas DataFrame to neatly display the topic probabilities for each document.

```
In [13]: # Creating a DataFrame for topic probabilities
topic_columns = [f"Topic {i + 1}" for i in range(n_topics)]
doc_topic_df = pd.DataFrame(doc_topic_dist, columns=topic_columns)

# Add document and category columns from the original dataset to the DataFrame
doc_topic_df['Document'] = df['Document']
doc_topic_df['Category'] = df['Category']

# Reorder the columns to show Document, Category, and then Topic probabilities
doc_topic_df = doc_topic_df[['Document', 'Category'] + topic_columns]

# Display the DataFrame
doc_topic_df
```

Out[13]:

	Document	Category	Topic 1	Topic 2
0	The game was thrilling and the team gave their...	Sports	0.074181	0.925819
1	A new tax bill has been passed in the parliament.	Politics	0.898465	0.101535
2	The government announced new healthcare reforms.	Politics	0.886708	0.113292
3	The player scored a hat-trick in the final match.	Sports	0.074699	0.925301
4	There was a discussion in the senate about the...	Politics	0.888280	0.111720
5	The football match was exhilarating, with both...	Sports	0.918945	0.081055
6	The president signed a bill for economic growth.	Politics	0.897226	0.102774
7	Many supporters celebrated the team's victory.	Sports	0.102750	0.897250
8	The debate on environmental law continues in t...	Politics	0.913526	0.086474
9	A new basketball court was inaugurated in the ...	Sports	0.243896	0.756104
10	A referendum was held to decide on the new tra...	Politics	0.922343	0.077657
11	The tennis championship saw some remarkable pe...	Sports	0.087241	0.912759
12	A major tax reform is being discussed by the f...	Politics	0.921852	0.078148
13	The athletics team won several medals at the e...	Sports	0.087783	0.912217
14	The government is introducing stricter regulat...	Politics	0.913603	0.086397
15	Cricket is growing in popularity in the region.	Sports	0.894934	0.105066
16	The senate passed a new education reform bill.	Politics	0.915127	0.084873
17	The team played a fantastic game and advanced ...	Sports	0.074656	0.925344
18	A new climate change initiative was debated in...	Politics	0.925471	0.074529
19	Football fans eagerly await the upcoming world...	Sports	0.929496	0.070504
20	Healthcare reforms have been the focus of the ...	Politics	0.922873	0.077127
21	Basketball players trained for the upcoming se...	Sports	0.096606	0.903394
22	The prime minister addressed the nation about ...	Politics	0.935660	0.064340
23	The game was interrupted due to rain, but resu...	Sports	0.085929	0.914071
24	A government report highlights the success of ...	Politics	0.926453	0.073547
25	The player was awarded the MVP title for his o...	Sports	0.073529	0.926471
26	A new international trade policy was discussed...	Politics	0.620600	0.379400
27	The team dominated the tournament with excelle...	Sports	0.887623	0.112377
28	Lawmakers are working on a bill to improve roa...	Politics	0.086431	0.913569
29	The tennis star announced her retirement after...	Sports	0.074539	0.925461

	Document	Category	Topic 1	Topic 2
30	The mayor discussed the importance of economic...	Politics	0.935354	0.064646
31	A new strategy for the national team has been ...	Sports	0.856612	0.143388
32	The president's speech focused on healthcare, ...	Politics	0.933596	0.066404
33	The coach praised the team for their efforts a...	Sports	0.087783	0.912217
34	The parliament is considering a new tax bill t...	Politics	0.935355	0.064645
35	The final match was intense, with both teams g...	Sports	0.092804	0.907196
36	The government is set to launch a new public w...	Politics	0.935098	0.064902
37	Basketball enthusiasts are looking forward to ...	Sports	0.073036	0.926964
38	New environmental regulations were implemented...	Politics	0.914028	0.085972
39	The star player returned from injury and led t...	Sports	0.064833	0.935167
40	A major healthcare overhaul is in the works, a...	Politics	0.084519	0.915481
41	The team celebrated their championship win wit...	Sports	0.075665	0.924335
42	The new policy on international trade has rece...	Politics	0.205123	0.794877
43	A thrilling finish to the basketball season le...	Sports	0.058830	0.941170
44	The senate passed a law regarding corporate ta...	Politics	0.935701	0.064299
45	The football team has been training hard for t...	Sports	0.905474	0.094526
46	There is ongoing discussion about increasing h...	Politics	0.101921	0.898079
47	The star athlete announced his plans for the n...	Sports	0.086949	0.913051
48	The prime minister introduced new policies to ...	Politics	0.942921	0.057079

Interpretation:

- Document 1, 3, and 5 are primarily related to **Topic 1** (sports).
- Document 2, 4, and 6 are primarily related to **Topic 2** (politics/government).

Step 6: Evaluating the Model

LDA models are often evaluated qualitatively by inspecting the top words per topic and judging whether they make sense for the discovered topics. More advanced metrics, such as **perplexity** and **coherence**, can also be used to assess how well the model fits the data.

```
In [14]: # Step 6: Evaluate the model using Perplexity
perplexity = lda.perplexity(X)
print(f"Model Perplexity: {perplexity:.2f}")
```

Model Perplexity: 257.33

Explanation:

- **Perplexity** is a measure of how well a probability model predicts a sample. Lower perplexity means a better model.
