CISB5123 Text Analytics
Lab 9
Topic Modeling

Topic modeling is a method to identify topics or themes within a collection of documents. It involves automatically clustering words that tend to co-occur frequently across multiple documents, with the aim of identifying groups of words that represent distinct topics. The ultimate goal is to identify the underlying themes or topics that run through a large corpus of text data.

In this lab, you will learn how to perform topic modeling using Gensim, a popular Python library for topic modeling, and the Latent Dirichlet Allocation (LDA) algorithm to discover abstract topics within a collection of documents.

The general process of topic modeling includes:

- Import the necessary libraries (i.e. NLTK, Gensim, Scikit-learn)
- Load data
- Preprocess the data
- Create a document-term matrix
- Run topic modeling algorithm
- Interpret the results
- Refine the model

EXAMPLE 1 – BASIC LDA WITH SHORT DOCUMENTS

We are going to perform Topic Modeling to the following documents:

```
documents = [
    "Rafael Nadal Joins Roger Federer in Missing U.S. Open",
    "Rafael Nadal Is Out of the Australian Open",
    "Biden Announces Virus Measures",
    "Biden's Virus Plans Meet Reality",
    "Where Biden's Virus Plan Stands"
]
```

1. First we need to import the libraries:

```
# For text preprocessing
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# For topic modeling
from gensim import corpora
from gensim.models import LdaModel

# Download NLTK Resources
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

2. Load the Data:

```
documents = [
    "Rafael Nadal Joins Roger Federer in Missing U.S. Open",
    "Rafael Nadal Is Out of the Australian Open",
    "Biden Announces Virus Measures",
    "Biden's Virus Plans Meet Reality",
    "Where Biden's Virus Plan Stands"
]
```

3. Preprocess the Data

In [7]:
```
stop_words = set(stopwords.words('english'))  # Create a set of English stopwords
lemmatizer = WordNetLemmatizer()  # Initialize a WordNet lemmatizer

def preprocess_text(text):
    tokens = word_tokenize(text.lower())  # Tokenize the text into words and convert to lowercase
    tokens = [token for token in tokens if token.isalnum()]  # Filter out non-alphanumeric tokens
    tokens = [token for token in tokens if token not in stop_words]  # Remove stopwords from the tokens
    tokens = [lemmatizer.lemmatize(token) for token in tokens]  # Lemmatize each token
    return tokens  # Return the preprocessed tokens

preprocessed_documents = [preprocess_text(doc) for doc in documents]  # Preprocess each document in the list
preprocessed_documents
```

Out[7]:
```
[['rafael', 'nadal', 'join', 'roger', 'federer', 'missing', 'open'],
 ['rafael', 'nadal', 'australian', 'open'],
 ['biden', 'announces', 'virus', 'measure'],
 ['biden', 'virus', 'plan', 'meet', 'reality'],
 ['biden', 'virus', 'plan', 'stand']]
```

4. Create a document-term matrix

```
# Create a Gensim Dictionary object from the preprocessed documents
dictionary = corpora.Dictionary(preprocessed_documents)
# Convert each preprocessed document into a bag-of-words representation using the dictionary
corpus = [dictionary.doc2bow(doc) for doc in preprocessed_documents]
```

5. Run LDA

```
#corpus: bag-of-words representation of the documents
#num_topics: number of topics to be extracted by the model
#id2word=dictionary: dictionary mapping from word IDs to words
#passes: number of passes through the corpus during training
 # Train an LDA model on the corpus with 4 topics using Gensim's LdaModel class
lda_model = LdaModel(corpus, num_topics=2, id2word=dictionary, passes=15)
```

6. Interpret Results

```python
# empty list to store dominant topic labels for each document
article_labels = []

#iterate over each processed document
for i, doc in enumerate(preprocessed_documents):
    # for each document, convert to box representation
    bow = dictionary.doc2bow(doc)
    # get list of topic probabilities
    topics = lda_model.get_document_topics(bow)
    # determine topic with highest probability
    dominant_topic = max(topics, key=lambda x: x[1])[0]
    # appenf to the list
    article_labels.append(dominant_topic)
```

```python
# Create DataFrame
df = pd.DataFrame({"Article": documents, "Topic": article_labels})

# Print the DataFrame
print("Table with Articles and Topic:")
print(df)
print()
```

```
Table with Articles and Topic:
                                              Article  Topic
0  Rafael Nadal Joins Roger Federer in Missing U....      1
1         Rafael Nadal Is Out of the Australian Open      1
2                     Biden Announces Virus Measures      0
3                   Biden's Virus Plans Meet Reality      0
4                    Where Biden's Virus Plan Stands      0
```

```python
# Print the top terms for each topic
print("Top Terms for Each Topic:")
for idx, topic in lda_model.print_topics():
    print(f"Topic {idx}:")
    terms = [term.strip() for term in topic.split("+")]
    for term in terms:
        weight, word = term.split("*")
        print(f"- {word.strip()} (weight: {weight.strip()})")
    print()

```

```
Top Terms for Each Topic:
Topic 0:
- "biden" (weight: 0.166)
- "virus" (weight: 0.166)
- "plan" (weight: 0.118)
- "meet" (weight: 0.071)
- "reality" (weight: 0.071)
- "stand" (weight: 0.071)
- "measure" (weight: 0.071)
- "announces" (weight: 0.071)
- "roger" (weight: 0.025)
- "federer" (weight: 0.025)

Topic 1:
- "nadal" (weight: 0.131)
- "rafael" (weight: 0.131)
- "open" (weight: 0.131)
- "australian" (weight: 0.079)
- "join" (weight: 0.078)
- "missing" (weight: 0.078)
- "federer" (weight: 0.078)
- "roger" (weight: 0.078)
- "announces" (weight: 0.027)
- "measure" (weight: 0.027)
```

Topic 0 seems to be related around politics and virus, where the weight of terms like "biden" and "virus" are particularly high, indicating their significance in this topic.

Topic 1 seems to be related to tennis, where the weight of terms like "nadal" and "rafael" are relatively high, suggesting a strong association with this topic.

EXAMPLE 2

We will be using articles ('npr.csv') from NPR (National Public Radio), obtained from their website www.npr.org

Import Libraries

```
# For text preprocessing
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# For topic modeling
from gensim import corpora
from gensim.models import LdaModel
import pandas as pd

# Download NLTK Resources
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

Load the Data

```
df = pd.read_csv('npr.csv')
documents = df['Article'].tolist()
```

Preprocess the Data

```
stop_words = set(stopwords.words('english'))  # Create a set of English stopwords
lemmatizer = WordNetLemmatizer()  # Initialize a WordNet lemmatizer

def preprocess_text(text):
    tokens = word_tokenize(text.lower())  # Tokenize the text into words and convert to lowercase
    tokens = [token for token in tokens if token.isalnum()]  # Filter out non-alphanumeric tokens
    tokens = [token for token in tokens if token not in stop_words]  # Remove stopwords from the tokens
    tokens = [lemmatizer.lemmatize(token) for token in tokens]  # Lemmatize each token
    return tokens  # Return the preprocessed tokens

preprocessed_documents = [preprocess_text(doc) for doc in documents]  # Preprocess each document
in the list

print(preprocessed_documents[0])
```

Create document-term matrix

```
# Create a Gensim Dictionary object from the preprocessed documents
dictionary = corpora.Dictionary(preprocessed_documents)

# Filter out tokens that appear in less than 15 documents or more than 50% of the documents
dictionary.filter_extremes(no_below=15, no_above=0.5)

# Convert each preprocessed document into a bag-of-words representation using the dictionary
corpus = [dictionary.doc2bow(doc) for doc in preprocessed_documents]
```

Run LDA

```
# Run LDA
lda_model = LdaModel(corpus, num_topics=5, id2word=dictionary, passes=15)  # Train an LDA model on the corpus with 2 topics using Gensim's LdaModel class
```

Interpret Results

```
# empty list to store dominant topic labels for each document
article_labels = []

# iterate over each processed document
for i, doc in enumerate(preprocessed_documents):
    # for each document, convert to bag-of-words representation
    bow = dictionary.doc2bow(doc)
    # get list of topic probabilities
    topics = lda_model.get_document_topics(bow)
    # determine topic with highest probability
    dominant_topic = max(topics, key=lambda x: x[1])[0]
    # append to the list
    article_labels.append(dominant_topic)

# Create DataFrame
df_result = pd.DataFrame({"Article": documents, "Topic": article_labels})

# Print the DataFrame
print("Table with Articles and Topic:")
print(df_result)
print()
```

Output:

```
Table with Articles and Topic:
                                                 Article  Topic
0      In the Washington of 2016, even when the polic...      4
1        Donald Trump has used Twitter  —   his prefe...      4
2        Donald Trump is unabashedly praising Russian...      4
3      Updated at 2:50 p. m. ET, Russian President Vl...      4
4      From photography, illustration and video, to d...      0
...                                                  ...    ...
11987  The number of law enforcement officers shot an...      2
11988    Trump is busy these days with victory tours,...      4
11989  It’s always interesting for the Goats and Soda...      1
11990  The election of Donald Trump was a surprise to...      4
11991  Voters in the English city of Sunderland did s...      2

[11992 rows x 2 columns]
```

```
# Print top terms for each topic
for topic_id in range(lda_model.num_topics):
    print(f"Top terms for Topic #{topic_id}:")
    top_terms = lda_model.show_topic(topic_id, topn=10)
    print([term[0] for term in top_terms])
    print()
```

Output:

```
Top terms for Topic #0:
['food', 'school', 'student', 'company', 'water', 'percent', 'world', 'job', 'university', 'much']

Top terms for Topic #1:
['know', 'think', 'thing', 'life', 'really', 'woman', 'story', 'show', 'book', 'something']

Top terms for Topic #2:
['police', 'country', 'report', 'city', 'government', 'attack', 'told', 'war', 'two', 'state']

Top terms for Topic #3:
['health', 'state', 'care', 'law', 'case', 'study', 'child', 'drug', 'patient', 'percent']

Top terms for Topic #4:
['trump', 'clinton', 'president', 'state', 'republican', 'campaign', 'election', 'obama', 'vote', 'house']
```

```
# Print the top terms for each topic with weight
print("Top Terms for Each Topic:")
for idx, topic in lda_model.print_topics():
    print(f"Topic {idx}:")
    terms = [term.strip() for term in topic.split("+")]
    for term in terms:
        weight, word = term.split("*")
        print(f"- {word.strip()} (weight: {weight.strip()})")
    print()
```

Output:

```
Top Terms for Each Topic:
Topic 0:
- "food" (weight: 0.006)
- "school" (weight: 0.006)
- "student" (weight: 0.005)
- "company" (weight: 0.005)
- "water" (weight: 0.004)
- "percent" (weight: 0.004)
- "world" (weight: 0.003)
- "job" (weight: 0.003)
- "university" (weight: 0.003)
- "much" (weight: 0.003)


Topic 1:
- "know" (weight: 0.005)
- "think" (weight: 0.005)
- "thing" (weight: 0.005)
- "life" (weight: 0.005)
- "really" (weight: 0.004)
- "woman" (weight: 0.004)
- "story" (weight: 0.004)
- "show" (weight: 0.003)
- "book" (weight: 0.003)
- "something" (weight: 0.003)

Topic 2:
- "police" (weight: 0.007)
- "country" (weight: 0.006)
- "report" (weight: 0.005)
- "city" (weight: 0.005)
- "government" (weight: 0.004)
- "attack" (weight: 0.004)
- "told" (weight: 0.004)
- "war" (weight: 0.004)
- "two" (weight: 0.004)
- "state" (weight: 0.004)
```

```
Topic 3:
- "health" (weight: 0.011)
- "state" (weight: 0.007)
- "care" (weight: 0.006)
- "law" (weight: 0.006)
- "case" (weight: 0.006)
- "study" (weight: 0.005)
- "child" (weight: 0.005)
- "drug" (weight: 0.005)
- "patient" (weight: 0.005)
- "percent" (weight: 0.005)

Topic 4:
- "trump" (weight: 0.028)
- "clinton" (weight: 0.011)
- "president" (weight: 0.010)
- "state" (weight: 0.008)
- "republican" (weight: 0.008)
- "campaign" (weight: 0.007)
- "election" (weight: 0.006)
- "obama" (weight: 0.006)
- "vote" (weight: 0.005)
- "house" (weight: 0.005)
```

Topic 0: Education
Topic 1: Personal
Topic 2: World Politics
Topic 3: Health
Topic 5: US Election