NLP HW4

1. Implement the code below and run it as is.

```
# Install Gensim if you haven't already
!pip install gensim
import gensim
from gensim import corpora
from gensim.models import LdaModel
from gensim.models import CoherenceModel
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk import download
import string

# Sample documents
documents = [
    "Topic modeling is an unsupervised machine learning technique used to
discover hidden topics in a collection of documents.",
    "Latent Dirichlet Allocation (LDA) is a popular topic modeling
algorithm.",
    "Gensim is a Python library for topic modeling.",
    "Natural Language Processing (NLP) is a field of study focused on
making sense of human language using computers.",
    "Topic modeling can be used for clustering similar documents or for
text summarization."
]

# Preprocessing
download('stopwords')
download('punkt')
download('wordnet')
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
def preprocess_text(text):
    tokens = word_tokenize(text.lower())
    tokens = [token for token in tokens if token not in stop_words and
token not in string.punctuation]
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return tokens
processed_docs = [preprocess_text(doc) for doc in documents]

# Create dictionary and corpus
dictionary = corpora.Dictionary(processed_docs)
corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
```

```
# Build LDA model
lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=2,
passes=10)

# Print the topics
for topic_id, topic in lda_model.print_topics():
    print(f"Topic {topic_id}: {topic}")
```

```
Topic 0: 0.072*"language" + 0.043*"sense" + 0.043*"processing" +
0.043*"using" + 0.043*"computer" + 0.043*"study" + 0.043*"making" +
0.043*"field" + 0.043*"focused" + 0.043*"natural"
Topic 1: 0.109*"topic" + 0.087*"modeling" + 0.053*"used" +
0.053*"document" + 0.032*"collection" + 0.032*"clustering" +
0.032*"dirichlet" + 0.032*"latent" + 0.032*"learning" + 0.032*"technique"
```

```
# Compute coherence score
coherence_model = CoherenceModel(model=lda_model, texts=processed_docs,
dictionary=dictionary, coherence='c_v')
coherence_score = coherence_model.get_coherence()
print(f"Coherence Score: {coherence_score}")
```

```
Coherence Score: 0.2935471717548586
```

2. In your report, report the output and discuss it from the perspective of the quality of the topics learned.

Two topics have been identified:

- Topic 0 seems to revolve around language processing, studying, and computer use, with keywords such as "language," "sense," "processing," "using," "computer," "study," "making," "focused," "natural," and "field."
- Topic 1 is clearly focused on the topic of topic modeling itself, with words like "topic," "modeling," "used," "document," "collection," "clustering," "dirichlet," "latent," and "learning."

The coherence score, which is a measure of how interpretable the topics are to humans, is reported as approximately 0.2935. In the context of topic modeling, a higher coherence score usually indicates better model performance, as the topics are more distinct and meaningful. A coherence score around 0.3 is relatively low, suggesting that the topics identified might not be very coherent or distinct from one another. This could be due to the small size of the document set (only four documents) and the close relation of the subject matter within these documents, which might not provide enough differentiation for the LDA to distinguish clear, separate topics.

For better results and higher coherence, one could consider the following:

- Increasing the corpus size with more documents to provide the LDA algorithm with more context and variance.
- Tuning hyperparameters of the LDA model such as the number of topics, the number of passes, and others.
- Trying different preprocessing steps, such as more thorough cleaning, to ensure that the most relevant words are being used to discover the topics.

Overall, the LDA has made an attempt to categorize the information in the provided documents, but the quality of the topics learned could likely be improved with a larger and more varied dataset.

3. Replace the documents hardcoded in this code by some documents that are more elaborate of your choice. Show these documents in your report.

```
# Sample documents
documents = [
    "Artificial Intelligence (AI) is a branch of computer science that aims to create systems able to perform tasks that would normally require human intelligence.",
    "Machine learning is a subset of AI that includes algorithms that give computers the ability to learn from and make predictions on data.",
    "Deep learning is a subset of machine learning that uses neural networks with many layers, enabling the modeling of complex patterns in data.",
    "Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns.",
    "Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world.",
    "Natural language processing (NLP) enables computers to understand and process human languages, facilitating user interactions.",
    "Data mining is the process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.",
    "Predictive analytics uses statistical algorithms and machine learning techniques to identify the likelihood of future outcomes based on historical data.",
    "Recurrent Neural Networks (RNNs) are a type of neural network that are well-suited to processing sequences of data for tasks like speech recognition.",
    "Generative Adversarial Networks (GANs) consist of two neural networks contesting with each other in a game, typically used in unsupervised learning tasks.",
```

```
    "Reinforcement learning is concerned with how software agents ought to
take actions in an environment to maximize some notion of cumulative
reward.",
    "Decision trees are a type of supervised learning algorithm that is
used for classification and regression tasks.",
    "Random forests are an ensemble learning method that operates by
constructing a multitude of decision trees at training time to output the
class that is the mode of the classes of the individual trees.",
    "Support vector machines (SVMs) are supervised learning models with
associated learning algorithms that analyze data for classification and
regression.",
    "K-means clustering is a type of unsupervised learning, which is used
when you have unlabeled data and the goal is to find groups in the data.",
    "The Principal Component Analysis (PCA) is a dimensionality-reduction
method that is often used to reduce the dimensionality of large data sets
by transforming a large set of variables into a smaller one that still
contains most of the information in the large set.",
    "Gradient boosting is a machine learning technique for regression and
classification problems that builds a model from a set of weak prediction
models, typically decision trees.",
    "Anomaly detection is the identification of rare items, events or
observations which raise suspicions by differing significantly from the
majority of the data.",
    "Time series analysis comprises methods for analyzing time series data
in order to extract meaningful statistics and other characteristics of the
data.",
    "Text mining, also referred to as text data mining, roughly equivalent
to text analytics, is the process of deriving high-quality information
from text.",
    "Bagging, or Bootstrap Aggregating, is a machine learning ensemble
meta-algorithm designed to improve the stability and accuracy of machine
learning algorithms.",
    "A/B testing, also known as split testing, is a marketing experiment
wherein you split your audience to test a number of variations of a
campaign and determine which performs better.",
    "Association rule learning is a rule-based machine learning method for
discovering interesting relations between variables in large databases.",
    "The field of robotics is closely related to AI. Robotics involves
building robots that can interact with the physical world.",
    "Bioinformatics involves the application of computational methods to
understand biological data, such as genetic sequences.",
    "Quantum computing is an emerging technology that uses the principles
of quantum mechanics to perform computations more efficiently than
traditional computers.",
```

```
    "The Internet of Things (IoT) is a network of physical objects that
are embedded with sensors, software, and other technologies for the
purpose of connecting and exchanging data with other devices and systems
over the internet."
]

coherence_scores = []
num_topics = []

# number of topics
for i in range(2, 10):
  # Build LDA model
  lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=i,
passes=50)
  # Compute coherence score
  coherence_model = CoherenceModel(model=lda_model, texts=processed_docs,
dictionary=dictionary, coherence='c_v')
  coherence_score = coherence_model.get_coherence()
  coherence_scores.append(coherence_score)
  num_topics.append(i)

max_index = np.argmax(coherence_scores)
best_coherence_score = coherence_scores[max_index]
best_num_topics = num_topics[max_index]
print(f"Number of Topics: {best_num_topics}, Coherence Score:
{best_coherence_score}")
```

```
Number of Topics: 4, Coherence Score: 0.48281782896858266
```

```
# Build best LDA model
best_lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=4,
passes=50)

# Print the topics
for topic_id, topic in best_lda_model.print_topics():
    print(f"Topic {topic_id}: {topic}")
```

Topic 0: 0.037*"learning" + 0.032*"data" + 0.032*"machine" + 0.027*"set" + 0.026*"large" + 0.021*"text" + 0.016*"method" + 0.016*"mining" + 0.011*"ai" + 0.011*"model"

Topic 1: 0.020*"human" + 0.020*"network" + 0.020*"algorithm" + 0.020*"internet" + 0.020*"language" + 0.011*"system" + 0.011*"physical" + 0.011*"software" + 0.011*"computer" + 0.011*"designed"

Topic 2: 0.046*"learning" + 0.024*"data" + 0.024*"machine" + 0.024*"us" + 0.017*"tree" + 0.017*"algorithm" + 0.017*"class" + 0.017*"quantum" + 0.009*"decision" + 0.009*"classification"

Topic 3: 0.034*"data" + 0.023*"network" + 0.017*"computer" + 0.017*"neural" + 0.017*"task" + 0.017*"intelligence" + 0.017*"learning" + 0.012*"used" + 0.012*"type" + 0.012*"time"

4. Run the code on your new documents and discuss the quality of the results from the perspective of what topics get learned and what words get associated with them.

Given the coherence score of approximately 0.4828, it is relatively moderate, indicating the topics are reasonably interpretable but still not highly distinct or exclusive. This could be due to overlapping vocabulary across topics or the diverse nature of the document set.

Now, looking at the topics produced by the best model:

- Topic 0 seems to focus on machine learning and data, with terms like "learning," "data," "machine," "set," "large," "text," "method," "mining," "ai," and "model." This topic seems to capture the general machine learning and data science aspect.
- Topic 1 combines a mix of technology and AI concepts, with words such as "human," "network," "algorithm," "internet," "language," "system," "physical," "software," "computer," "designed." This topic might reflect the intersection of human-computer interaction, networking, and algorithmic development.
- Topic 2 focuses heavily on learning and algorithms, with "learning," "data," "machine," "us," "tree," "algorithm," "class," "quantum," "decision," "classification." It seems to be more centered on machine learning algorithms and their applications.
- Topic 3 is about data processing and neural networks, as seen with words like "data," "network," "computer," "neural," "task," "intelligence," "learning," "used," "type," "time." This could represent topics related to artificial intelligence and neural networks specifically.

The quality of these results, based on the topics and their associated words, seems to be quite good considering the coherence score. The LDA model has managed to find words that are thematically related to each other and to the overall topic they represent.

However, the distinctiveness of the topics could potentially be improved. For example, words like "learning" and "data" appear in multiple topics, which could indicate some overlap. This is a common issue in LDA models, especially when the corpus contains documents with similar or related content. Increasing the number of documents, providing a larger variety of topics, or fine-tuning preprocessing steps (like including bigrams or trigrams) could potentially result in clearer, more separated topics.

Moreover, for a more thorough understanding of the model's quality, it would be beneficial to read through the actual documents tagged with each topic to see if they make sense, or

to look at the distribution of topics across documents to ensure that they are not just a collection of commonly used words but represent coherent themes within the documents.

5. Add a section to your report in which you describe in your own words what the various main lines in the code do. Think of it this way. Imagine that you had to explain this code to someone (such as a product manager or executive) who does not understand Python or the various operations this code performs. Your section should contain this explanation. For example, explain what wordnet is and how it is used in this code? What is lemmatization and why is it used in this code?

```
#load 'stopwords' package that contains lists of stop words, which are
common words like "the," "is," "in," etc., that are often removed during
text preprocessing to focus on more meaningful words.
download('stopwords')
#load 'punkt' package that is used for tokenizing text into sentences and
words. It contains models for punctuation that helps in splitting text
into a list of words or sentences.
download('punkt')
#load 'wordnet' package that allows access to the WordNet database, which
is used for lemmatization. WordNet is a large lexical database of English
where nouns, verbs, adjectives, and adverbs are grouped into sets of
cognitive synonyms (synsets).
download('wordnet')
#create a set of English stop words
stop_words = set(stopwords.words('english'))
#lemmatizer is used to reduce words to their base or dictionary form
(lemmas)
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    # create a new tokens list that converts each word into lower case and
split the string into individual words or tokens (including punctuations
and special characters).
    tokens = word_tokenize(text.lower())
    # create a new tokens list that includes only tokens not in
'stop_words' list and not found in 'string.punctuation'
(!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~)
    tokens = [token for token in tokens if token not in stop_words and
token not in string.punctuation]
    # create a new tokens list that reduces a word to its base or root
form (lemma)
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return tokens
```

```
# create a new processed_docs list that converts each string in the
documents using preprocess_text() function
processed_docs = [preprocess_text(doc) for doc in documents]

# it maps each unique word in the processed documents to a unique integer
ID and collects statistics about word freqencies.
dictionary = corpora.Dictionary(processed_docs)
# each document is transformed into a list of tuples, with each tuple
containing a word ID (corresponding to a unique word) and its frequency in
that document.
corpus = [dictionary.doc2bow(doc) for doc in processed_docs]

# create a coherence_scores list that will store coherence score of
different model
coherence_scores = []
# create a num_topics list that will store number of topics parameter
num_topics = []

# loop over different number of topics
for i in range(2, 10):
  # Build LDA model
  lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=i,
passes=50)
  # Evaluate the quality of the topics that a topic model (LDA) has
learned.
  coherence_model = CoherenceModel(model=lda_model, texts=processed_docs,
dictionary=dictionary, coherence='c_v')
  # Compute coherence store of a coherence model
  coherence_score = coherence_model.get_coherence()
  # Append coherence score of each model to the coherence_scores list
  coherence_scores.append(coherence_score)
  # Append number of topics parameter to the num_topics list
  num_topics.append(i)

# identify the index for the highest coherence score
max_index = np.argmax(coherence_scores)
# identify the highest coherence score
best_coherence_score = coherence_scores[max_index]
# identify the number of topics parameter (model) that has the highest
coherence score
best_num_topics = num_topics[max_index]
print(f"Number of Topics: {best_num_topics}, Coherence Score:
{best_coherence_score}")
```

```
# Build the LDA model that has the highest coherence score
best_lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=4,
passes=50)

# Print the topics
for topic_id, topic in best_lda_model.print_topics():
    print(f"Topic {topic_id}: {topic}")
```