NAME: ARJUN T ANIL

REG NO: 24PMC117

**CADL4:**

With a text corpus (e.g., a collection of news articles, academic papers, or social media posts):

a.      Preprocess the text data (tokenization, stop word removal, etc.).

b.      Apply LDA to identify topics within the corpus using a Python library (e.g., Gensim).

c.       Visualize the topics and their associated words.

Document the code and the results of their LDA model in a Moodle workshop.

GitHub Code: <https://github.com/arjuntanil/CADL4.git>

Code:

**1. Restart Runtime**

import os, sys

os.kill(os.getpid(), 9)

* This line forces Google Colab (or Jupyter) to restart the Python runtime after installing or fixing dependencies.
* Useful when some libraries (like SpaCy models) require a restart.

**2. Import Libraries & Download NLTK Data**

import nltk

nltk.download('punkt')

nltk.download('punkt\_tab')

nltk.download('stopwords')

nltk.download('omw-1.4')

* **NLTK** provides tools for tokenization, stop words, etc.
* Downloads required datasets:
  + punkt → tokenizer model.
  + stopwords → list of common words (e.g., *the, is, and*).
  + omw-1.4 → WordNet lemmatizer dictionary.

**3. Load SpaCy Model**

import spacy

nlp = spacy.load("en\_core\_web\_sm", disable=['parser','ner'])

* Loads SpaCy’s **English small model**.
* We disable parser + NER since we only need the **tagger + lemmatizer** for text cleaning.

**4. Import Other Libraries**

import gensim

from gensim import corpora

from gensim.models import Phrases, LdaModel

from gensim.models.phrases import Phraser

import pyLDAvis

import pyLDAvis.gensim\_models as gensimvis

import pandas as pd

import matplotlib.pyplot as plt

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

* **Gensim** → used for topic modeling (LDA).
* **Phrases/Phraser** → for creating bigrams/trigrams.
* **pyLDAvis** → interactive visualization of topics.
* **matplotlib, pandas** → general visualization/data handling.

**5. Define Corpus**

corpus = [

"Just tried the new cafe downtown — the coffee was amazing and the ambiance perfect.",

"Feeling blessed to have completed my first 5K run today! #fitness #goals",

...

]

* A small dataset of **10 social media posts** used as our sample corpus.

**6. Preprocessing Function**

stop\_words = set(stopwords.words('english'))

def preprocess\_texts(texts, nlp\_pipeline):

processed\_texts = []

for doc in texts:

tokens = word\_tokenize(doc.lower()) # Tokenization

tokens = [t for t in tokens if t.isalpha() and t not in stop\_words and len(t) > 2] # Remove stopwords & short tokens

spacy\_doc = nlp\_pipeline(" ".join(tokens))

lemmas = [token.lemma\_ for token in spacy\_doc if token.is\_alpha and token.lemma\_ not in stop\_words and len(token.lemma\_) > 2] # Lemmatization

processed\_texts.append(lemmas)

return processed\_texts

🔹 Steps inside function:

1. **Lowercasing** all words.
2. **Tokenization** using NLTK.
3. Remove:
   * Non-alphabetic tokens (e.g., numbers, symbols).
   * Stop words (the, is, etc.).
   * Very short words (like *a, an*).
4. **Lemmatization** using SpaCy → converts words to their root forms (*running → run, studies → study*).

**7. Run Preprocessing**

processed = preprocess\_texts(corpus, nlp)

print("Sample preprocessed docs:", processed[:3])

* Cleans the corpus.
* Example:
* "Just tried the new cafe downtown — the coffee was amazing"
* → ['try', 'new', 'cafe', 'downtown', 'coffee', 'amazing', 'ambiance', 'perfect']

**8. Create Bigrams**

bigram = Phrases(processed, min\_count=2, threshold=5)

bigram\_mod = Phraser(bigram)

data\_bigrams = [bigram\_mod[doc] for doc in processed]

* **Bigrams** = word pairs (e.g., *machine learning*, *data science*).
* Captures phrases that appear together often.
* min\_count=2 → phrase must appear at least twice.

**9. Dictionary + Corpus**

id2word = corpora.Dictionary(data\_bigrams)

id2word.filter\_extremes(no\_below=1, no\_above=0.5)

gensim\_corpus = [id2word.doc2bow(text) for text in data\_bigrams]

* **Dictionary** maps each word to an ID.
* filter\_extremes removes very rare or very common words.
* **Corpus (Bag of Words)** represents each document as a list of (word\_id, frequency) pairs.

**10. Train LDA Model**

num\_topics = 3

lda\_model = LdaModel(corpus=gensim\_corpus,

id2word=id2word,

num\_topics=num\_topics,

random\_state=100,

update\_every=1,

passes=10,

alpha='auto')

* Builds a **Latent Dirichlet Allocation (LDA)** model.
* Parameters:
  + num\_topics=3 → find 3 hidden topics.
  + passes=10 → number of iterations over data.
  + alpha='auto' → adjusts topic-document distribution automatically.

**11. Print Topics**

for idx, topic in lda\_model.print\_topics(num\_topics=num\_topics, num\_words=6):

print(f"Topic {idx}: {topic}")

* Prints each discovered topic and its top words.
* Example output:
* Topic 0: coffee, cafe, downtown, amazing
* Topic 1: run, fitness, goals, today
* Topic 2: climate, change, ice, melt

**12. Visualization**

pyLDAvis.enable\_notebook()

vis = gensimvis.prepare(lda\_model, gensim\_corpus, id2word)

vis

* Produces an **interactive visualization**:
  + Each **circle = topic**.
  + Distance between circles shows how similar topics are.
  + Right panel shows **top words per topic**.

Output :

