BICOL UNIVERSITY COLLEGE OF SCIENCE
CS Elective – Artificial Intelligence
Coding Exercises - 5

```
# "Lastname_Firstname_Topic_Modeling.ipynb"
# ================================================
# INSTALL LIBRARIES
# ================================================
!pip install gensim pandas matplotlib pyLDAvis wordcloud --quiet
```

gensim → for LDA topic modeling.
pandas → to handle the dataset.
matplotlib → for plotting word clouds.
pyLDAvis → for interactive topic visualization.
wordcloud → to generate word clouds per topic.

```
# ================================================
# IMPORTS
# ================================================
import pandas as pd
from gensim import corpora
from gensim.models.ldamodel import LdaModel
from gensim.models import CoherenceModel
import re
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis

# ================================================
# SAMPLE DATASET (100 REVIEWS)
# ================================================
positive_reviews = [
    "The product is excellent, I love it!",
    "Very good quality, worth every peso.",
    "Fast delivery and great packaging.",
    "Item arrived in perfect condition.",
    "Super satisfied with this purchase.",
    "Amazing product! Works perfectly.",
    "Highly recommended seller!",
    "Great value for the price.",
    "The material feels premium.",
    "Exactly as described. Very happy!"
]
```

```
negative_reviews = [
    "Terrible quality, not worth the money.",
    "Very disappointed. Stopped working after one day.",
    "Item arrived damaged and unusable.",
    "The product feels very cheap.",
    "Not what I expected at all.",
    "Shipping took too long.",
    "The item is defective.",
    "Completely useless, waste of money.",
    "Wrong item was delivered.",
    "Poor quality materials used."
]
```

Created 100 sample reviews: 50 positive, 50 negative.
Duplicated short positive/negative reviews to make the dataset bigger.
Shuffled the dataset for randomness.

```
df = pd.DataFrame({
    "review": positive_reviews * 5 + negative_reviews * 5
})
df = df.sample(frac=1, random_state=42).reset_index(drop=True)
texts = df["review"].tolist()

# ================================================
# SIMPLE TEXT PREPROCESSING
# ================================================
stop_words = set([
    "the", "is", "and", "a", "an", "for", "to", "it", "this", "very", "with"
])

def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = text.split()
    tokens = [w for w in tokens if w not in stop_words]
    return tokens

processed_docs = [preprocess(doc) for doc in texts]
```

Text Preprocessing

- Lowercase all words.
- Remove punctuation and numbers with regex.
- Split text by spaces.
- Remove common stopwords manually (simple approach avoids NLTK punkt_tab errors).

```
# ================================================
# DICTIONARY AND CORPUS
# ================================================
dictionary = corpora.Dictionary(processed_docs)
corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
```

Dictionary → maps each word to a unique ID.
Corpus → converts each document into a bag-of-words (word ID + frequency).

```
# ================================================
# LDA MODEL (5 TOPICS)
# ================================================
lda_model = LdaModel(
    corpus=corpus,
    id2word=dictionary,
    num_topics=5,
    random_state=42,
    passes=10
)
```

Train with num_topics=5 and passes=10 for reasonable results.
random_state=42 ensures reproducibility.
You may change these parameters

```
# ================================================
# DISPLAY TOPICS (TOP 10 WORDS)
# ================================================
print("===== LDA TOPICS (5 topics × 10 words) =====")
for t in range(5):
    print(f"\nTopic {t}:")
    print(lda_model.print_topic(t, topn=10))

# ================================================
# COHERENCE SCORE
# ================================================
coherence_model_lda = CoherenceModel(
    model=lda_model,
    texts=processed_docs,
    dictionary=dictionary,
    coherence='c_v'
)
lda_coherence = coherence_model_lda.get_coherence()
print("\nLDA Topic Coherence (c_v):", lda_coherence)
```

Measures **topic quality** (c_v): closer to 1 = better.

```
# ================================================
# WORD CLOUD FOR EACH TOPIC
# ================================================
for t in range(5):
    plt.figure(figsize=(6,4))
    topic_words = dict(lda_model.show_topic(t, topn=20))
    wc = WordCloud(width=600, height=400, background_color='white')
    wc.generate_from_frequencies(topic_words)
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.title(f"Topic {t} Word Cloud")
    plt.show()
```

Visualizes the importance of each word in the topic.
Bigger words → higher weight.

```
# ================================================
# INTERACTIVE PYLDAVIS VISUALIZATION
# ================================================
pyLDAvis.enable_notebook()
vis = gensimvis.prepare(lda_model, corpus, dictionary)
vis
```

Shows topic distribution, importance, and distance between topics.

- Circle size → topic importance
- Distance → how similar topics are
- Hover → see top words with weights