NLP INTERNAL-2

SET-1

A)

```
-- !pip install genism

-- !pip install nltk
from gensim.models import Word2Vec

from nltk.tokenize import word_tokenize

from sklearn.metrics.pairwise import cosine_similarity

import nltk

nltk.download('punkt')

corpus = "Alice is sitting on a bench with her sister, and she is really rather bored. Suddenly, a white rabbit passes by, checking his watch and lamenting that he is going to be late."

tokens = [word_tokenize(corpus.lower())]

model = Word2Vec(sentences=tokens, vector_size=50, window=3, min_count=1, sg=1)

alice_vector = model.wv['alice']

rabbit_vector = model.wv['rabbit']

similarity_score = cosine_similarity([alice_vector], [rabbit_vector])[0][0]

print(f"Similarity score between 'Alice' and 'rabbit': {similarity_score:.4f}")
```

B)

```
-- !pip install sentence-transformers

import gensim

import numpy as np

corpus = [

    "Machine learning is a subset of artificial intelligence.",

    "Data privacy and security are important in modern technology.",

    "Mobile apps are becoming more popular among users.",

    "E-commerce platforms make online shopping convenient.",

    "Cyber security measures are crucial to protect against cyber threats."
]
data = [sentence.lower().split() for sentence in corpus]
model = gensim.models.Word2Vec(data, vector_size=50, window=5, min_count=1)
for sentence in corpus:
    vector = np.mean([model.wv[word] for word in sentence.lower().split() if word in model.wv], axis=0)
    print(f"Sentence: {sentence}\nAssignment Vector (first 5 values): {vector[:5]}\n")
```

SET-2

A)

```
-- !pip install genism
from gensim.models import Word2Vec
sentences = [
    ["data", "science", "is", "fun"],
    ["word", "embedding", "is", "useful"],
    ["machine", "learning", "is", "interesting"],
    ["I", "love", "deep", "learning"]
]
model = Word2Vec(sentences, vector_size=50, window=5, min_count=1, workers=2)
word = "learning"
vector = model.wv[word]
print(f"Vector representation for '{word}':\n", vector)
similar_words = model.wv.most_similar(word, topn=3)
print(f"\nWords similar to '{word}':")
for similar_word, score in similar_words:
    print(f"{similar_word}: {score:.4f}")
```

B)

```
-- !pip install genism
import warnings
import gensim
```

```
from gensim.corpora.dictionary import Dictionary
warnings.filterwarnings(action='ignore')
corpus = [
    "Sugar is bad to consume. My sister likes to have sugar, but not my father.",
    "My father spends a lot of time driving my sister around to dance practice.",
    "Doctors suggest that driving may cause increased stress and blood pressure.",
    "Sometimes I feel pressure to perform well at school, but my father never seems to drive my sister to do better.",
    "Health experts say that Sugar is not good for your lifestyle."
]
data = [sentence.lower().split() for sentence in corpus]
dictionary = Dictionary(data)
bow_corpus = [dictionary.doc2bow(text) for text in data]
lda_model = gensim.models.LdaModel(bow_corpus, num_topics=2, id2word=dictionary,passes=10)
for idx, topic in lda_model.print_topics(-1):
    print(f"Topic {idx + 1}: {topic}")
```

SET – 3

A) (This code can be executed only in colab notebook)

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
num_samples = 1000
sequence_length = 10
vocab_size = 10000
X = np.random.randint(vocab_size, size=(num_samples, sequence_length))
y = np.random.randint(2, size=num_samples)
model = Sequential([
    Embedding(vocab_size, 32, input_length=sequence_length),
    SimpleRNN(64),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X, y, epochs=3, batch_size=32, validation_split=0.2)
```

B) (This code can be executed only in colab notebook)

```
from transformers import pipeline
classifier = pipeline("sentiment-analysis")
reviews = [
    "I absolutely loved the movie, it was fantastic!",
    "The plot was dull and uninspiring.",
    "Amazing storyline and great acting!",
    "I didn't enjoy the movie, it was too slow."
]
for review in reviews:
    result = classifier(review)[0]
    print(f"Review: {review}\nSentiment: {result['label']}, Score: {result['score']:.2f}\n")
```

SET – 4

A) (This code can be executed only in colab notebook)

```
from transformers import pipeline
classifier = pipeline("text-classification", model="bert-base-uncased", tokenizer="bert-base-uncased")
texts = [
    "I absolutely loved the movie, it was fantastic!",
    "The plot was dull and uninspiring.",
    "Amazing storyline and great acting!",
    "I didn't enjoy the movie, it was too slow."
]
```

```
for text in texts:
    result = classifier(text)[0]
    print(f"Text: {text}\nLabel: {result['label']}, Score: {result['score']:.2f}\n")
```

B)

```
num_dict = {
    'zero': 0, 'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5, 'six': 6,
    'seven': 7, 'eight': 8, 'nine': 9, 'ten': 10, 'eleven': 11, 'twelve': 12,
    'thirteen': 13, 'fourteen': 14, 'fifteen': 15, 'sixteen': 16, 'seventeen': 17,
    'eighteen': 18, 'nineteen': 19, 'twenty': 20, 'thirty': 30, 'forty': 40,
    'fifty': 50, 'sixty': 60, 'seventy': 70, 'eighty': 80, 'ninety': 90
}
def text_to_number(text):
    words = text.lower().split()
    num = 0
    for word in words:
        if word in num_dict:
            num += num_dict[word]
    return num
print(text_to_number("five"))
print(text_to_number("twenty three"))
```

SET – 5

A) (This code can be executed only in colab notebook)

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import yfinance as yf
data = yf.download('AAPL', start="2022-01-01", end="2023-01-01")
data['Previous Close'] = data['Close'].shift(1)
```

```
data = data.dropna()
x = data[['Previous Close', 'Volume']]
y = data['Open']
model = LinearRegression().fit(x, y)
future_data = [[150.00, 1000000]]
predicted_open = model.predict(future_data)
print(f'Predicted open price: {predicted_open[0]}')
```

B)

```
import numpy
responses = {
"hello": ["Hi!", "Hello!", "Hey there!"],
"how are you": ["I'm just a bot, but I'm doing well!", "I'm good, thanks!"],
"bye": ["Goodbye!", "See you later!", "Take care!"],
}
def get_response(message):
    message = message.lower()
    for key in responses:
        if key in message:
            return random.choice(responses[key])
    return "I'm sorry, I don't understand."
while True:
    user_input = input("You: ")
    if user_input.lower() == "exit":
        print("Chatbot: Goodbye!")
    break
response = get_response(user_input)
print("Chatbot:", response)
```

SET- 6

A) (This code can be executed only in colab notebook)

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis")

def should_watch_movie(review):

    sentiment = classifier(review)[0]

    if sentiment['label'] == 'POSITIVE' and sentiment['score'] > 0.8:

        return "Decision: Go for the movie!"

    else:

        return "Decision: Skip the movie."

review = "The movie was thrilling with an amazing storyline!"

print(f"Review: {review}")

print(should_watch_movie(review))
```
