PRACTICAL-5

PROBLEM STATEMENT:

5. Implement the Continuous Bag of Words (CBOW) Model. Stages can be:

a. Data preparation

b. Generate training data

c. Train model

d. Output

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CODE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import numpy as np

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, Lambda, Dense

from tensorflow.keras import backend as K

import matplotlib.pyplot as plt

# Example text data

text = "This is a simple example sentence for learning a Continuous Bag of Words model with TensorFlow."

# a. DATA PREPARATION

# Tokenize the text

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts([text])

word2idx = tokenizer.word\_index

idx2word = {v: k for k, v in word2idx.items()}

vocab\_size = len(word2idx) + 1 # Add 1 for padding

# Convert text to sequence of word indices

sequences = tokenizer.texts\_to\_sequences([text])[0]

# b. GENERATE TRAINING DATA

# Generate context-target pairs

window\_size = 2

data = []

for i in range(window\_size, len(sequences) - window\_size):

context = [sequences[i - j] for j in range(1, window\_size + 1)] + \

[sequences[i + j] for j in range(1, window\_size + 1)]

target = sequences[i]

data.append((context, target))

# Convert data into input and output arrays

x\_train = np.array([item[0] for item in data])

y\_train = np.array([item[1] for item in data])

# c. TRAIN THE MODEL

# Model parameters

embedding\_dim = 50

# CBOW model structure

model = Sequential([

Embedding(input\_dim=vocab\_size, output\_dim=embedding\_dim), # Embedding layer without input\_length

Lambda(lambda x: K.mean(x, axis=1)), # Averaging context embeddings

Dense(vocab\_size, activation='softmax') # Predict target word

])

# Compile the model

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy')

model.summary()

# Train the model and capture the history

history = model.fit(x\_train, y\_train, epochs=100, verbose=2)

# d. OUTPUT

# Get the word embeddings from the trained model

embedding\_layer = model.layers[0]

weights = embedding\_layer.get\_weights()[0]

# Print the embedding for a sample word

sample\_word = "learning"

print(f"Embedding for '{sample\_word}':\n", weights[word2idx[sample\_word]])

# Display the learned embeddings for each word

for word, index in word2idx.items():

print(f"Word: {word}, Embedding: {weights[index]}")

# e. PLOT TRAINING LOSS

# Plot training loss

plt.plot(history.history['loss'], label='loss')

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend()

plt.show()