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

#a. Data preparation

#b. Generate training data

#c. Train model

#d. Output

#Data preparation

import numpy as np

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

# Sample text data (replace with your own text data)

text\_data = [

    "this is an example sentence for cbow modeling",

    "cbow is a word embedding technique",

    "word embeddings capture the meaning of words",

]

# Tokenize the text data

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(text\_data)

# Convert text to sequences of numerical tokens

sequences = tokenizer.texts\_to\_sequences(text\_data)

# Create a vocabulary and reverse vocabulary

word\_index = tokenizer.word\_index

reverse\_word\_index = {v: k for k, v in word\_index.items()}

# Determine the context window size

context\_window = 2

# Generate context-target pairs

context\_target\_pairs = []

for sequence in sequences:

    for target\_position in range(context\_window, len(sequence) - context\_window):

        context = sequence[target\_position - context\_window:target\_position] + \

                  sequence[target\_position + 1:target\_position + 1 + context\_window]

        target = sequence[target\_position]

        context\_target\_pairs.append((context, target))

#Generate the model

# Split the context and target from the pairs

contexts, targets = zip(\*context\_target\_pairs)

# Convert contexts and targets to numpy arrays

contexts = np.array(contexts)

targets = np.array(targets)

#Train the model

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, Dense, GlobalAveragePooling1D

# Define the CBOW model

embedding\_dim = 100  # Adjust as needed

model = Sequential([

    Embedding(input\_dim=len(word\_index) + 1, output\_dim=embedding\_dim, input\_length=2 \* context\_window),

    GlobalAveragePooling1D(),

    Dense(len(word\_index) + 1, activation='softmax')

])

# Compile the model

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

# Train the model

model.fit(contexts, targets, epochs=100, batch\_size=64)

#Output

# Get the word embeddings from the model

word\_embeddings = model.layers[0].get\_weights()[0]

# Access the embeddings for a specific word (e.g., 'example')

word = 'example'

word\_index = word\_index[word]

embedding\_vector = word\_embeddings[word\_index]

print(f'Embedding for "{word}": {embedding\_vector}')