```python
import numpy as np
import pandas as pd
#from keras.preprocessing.text import Tokenizer  # This import is no
longer valid in Keras 3
from tensorflow.keras.preprocessing.text import Tokenizer # Import the
correct Tokenizer class
from tensorflow.keras.utils import text_dataset_from_directory #
Updated import for Keras 3
from tensorflow.keras.utils import pad_sequences
from tensorflow.keras.utils import to_categorical

# Sample corpus
corpus = [
    "I love programming in Python",
    "I enjoy learning about machine learning",
    "Python is great for data science",
    "I love learning new technologies",
    "Programming is fun and rewarding"
]

# Parameters
window_size = 2  # Number of context words
embedding_dim = 10  # Dimension of the embedding layer

# Tokenize the text
# tokenizer = Tokenizer()  # This initialization is no longer valid in
Keras 3
tokenizer = Tokenizer() # Create an instance of the Tokenizer class
tokenizer.fit_on_texts(corpus)
word_index = tokenizer.word_index
index_word = {i: w for w, i in word_index.items()}
print(f"Word Index: {word_index}")
```

```
Word Index: {'i': 1, 'learning': 2, 'love': 3, 'programming': 4,
'python': 5, 'is': 6, 'in': 7, 'enjoy': 8, 'about': 9, 'machine': 10,
'great': 11, 'for': 12, 'data': 13, 'science': 14, 'new': 15,
'technologies': 16, 'fun': 17, 'and': 18, 'rewarding': 19}
```

```python
# Generate training data
def generate_training_data(corpus, window_size):
    input_data = []
    target_data = []

    for sentence in corpus:
        words = sentence.split()
        for i, word in enumerate(words):
            # Define the start and end indices of the context window
            start_index = max(0, i - window_size)
            end_index = min(len(words), i + window_size + 1)
```

```
            # Lowercase words before appending to the lists
            context = [words[j].lower() for j in range(start_index, 
end_index) if j != i]  # Exclude the target word
            input_data.append(context)
            target_data.append(word.lower()) # Lowercase target word 
as well

    return input_data, target_data

input_data, target_data = generate_training_data(corpus, window_size)
print(f"Input Data: {input_data}")
print(f"Target Data: {target_data}")

# Convert words to sequences
# Lowercase words when looking them up in the word_index
input_sequences = [[word_index[word] for word in context] for context 
in input_data]
target_sequences = [word_index[word] for word in target_data]

# Pad sequences
max_length = max(len(seq) for seq in input_sequences)
input_sequences = pad_sequences(input_sequences, maxlen=max_length)

# One-hot encode target data
target_sequences = to_categorical(target_sequences, 
num_classes=len(word_index) + 1)

print(f"Padded Input Sequences: {input_sequences}")
print(f"One-hot Encoded Target Sequences: {target_sequences}")
```

```
Input Data: [['love', 'programming'], ['i', 'programming', 'in'], 
['i', 'love', 'in', 'python'], ['love', 'programming', 'python'], 
['programming', 'in'], ['enjoy', 'learning'], ['i', 'learning', 
'about'], ['i', 'enjoy', 'about', 'machine'], ['enjoy', 'learning', 
'machine', 'learning'], ['learning', 'about', 'learning'], ['about', 
'machine'], ['is', 'great'], ['python', 'great', 'for'], ['python', 
'is', 'for', 'data'], ['is', 'great', 'data', 'science'], ['great', 
'for', 'science'], ['for', 'data'], ['love', 'learning'], ['i', 
'learning', 'new'], ['i', 'love', 'new', 'technologies'], ['love', 
'learning', 'technologies'], ['learning', 'new'], ['is', 'fun'], 
['programming', 'fun', 'and'], ['programming', 'is', 'and', 
'rewarding'], ['is', 'fun', 'rewarding'], ['fun', 'and']]
Target Data: ['i', 'love', 'programming', 'in', 'python', 'i', 
'enjoy', 'learning', 'about', 'machine', 'learning', 'python', 'is', 
'great', 'for', 'data', 'science', 'i', 'love', 'learning', 'new', 
'technologies', 'programming', 'is', 'fun', 'and', 'rewarding']
Padded Input Sequences: [[ 0  0  3  4]
 [ 0  1  4  7]
 [ 1  3  7  5]
```

```
 [ 0  3  4  5]
 [ 0  0  4  7]
 [ 0  0  8  2]
 [ 0  1  2  9]
 [ 1  8  9 10]
 [ 8  2 10  2]
 [ 0  2  9  2]
 [ 0  0  9 10]
 [ 0  0  6 11]
 [ 0  5 11 12]
 [ 5  6 12 13]
 [ 6 11 13 14]
 [ 0 11 12 14]
 [ 0  0 12 13]
 [ 0  0  3  2]
 [ 0  1  2 15]
 [ 1  3 15 16]
 [ 0  3  2 16]
 [ 0  0  2 15]
 [ 0  0  6 17]
 [ 0  4 17 18]
 [ 4  6 18 19]
 [ 0  6 17 19]
 [ 0  0 17 18]]
One-hot Encoded Target Sequences: [[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

```python
from keras.models import Sequential
from keras.layers import Embedding, Dense, Flatten

# Define the CBOW model
model = Sequential()
model.add(Embedding(input_dim=len(word_index) + 1,
output_dim=embedding_dim, input_length=max_length))
model.add(Flatten())
model.add(Dense(len(word_index) + 1, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model
model.fit(input_sequences, target_sequences, epochs=100, verbose=1)
```

```
Epoch 1/100
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
embedding.py:90: UserWarning: Argument `input_length` is deprecated.
Just remove it.
  warnings.warn(
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.0370 - loss: 3.0064
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.0370 - loss:
3.0023
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.0370 - loss:
2.9981
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.0370 - loss:
2.9940
Epoch 5/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.0370 - loss:
2.9899
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.0370 - loss:
2.9858
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.0741 - loss:
2.9817
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.0741 - loss:
2.9776
```

```
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - accuracy: 0.0741 - loss:
2.9735
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.1111 - loss:
2.9694
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.1111 - loss:
2.9653
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.1481 - loss:
2.9612
Epoch 13/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.1852 - loss:
2.9571
Epoch 14/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.2593 - loss:
2.9530
Epoch 15/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - accuracy: 0.2593 - loss:
2.9488
Epoch 16/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - accuracy: 0.2593 - loss:
2.9447
Epoch 17/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.2593 - loss:
2.9405
Epoch 18/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.3333 - loss:
2.9362
Epoch 19/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.4444 - loss:
2.9320
Epoch 20/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.4444 - loss:
2.9277
Epoch 21/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.4444 - loss:
2.9234
Epoch 22/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4444 - loss:
2.9190
Epoch 23/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.4444 - loss:
2.9146
Epoch 24/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step - accuracy: 0.4444 - loss:
2.9101
Epoch 25/100
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.4444 - loss:
2.9056
Epoch 26/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.4444 - loss:
2.9010
Epoch 27/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.4815 - loss:
2.8964
Epoch 28/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4815 - loss:
2.8917
Epoch 29/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.4815 - loss:
2.8869
Epoch 30/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4815 - loss:
2.8821
Epoch 31/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.4815 - loss:
2.8772
Epoch 32/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.5185 - loss:
2.8722
Epoch 33/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.5556 - loss:
2.8672
Epoch 34/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.5556 - loss:
2.8620
Epoch 35/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.5185 - loss:
2.8568
Epoch 36/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.5185 - loss:
2.8515
Epoch 37/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.5185 - loss:
2.8461
Epoch 38/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.5185 - loss:
2.8406
Epoch 39/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.5185 - loss:
2.8351
Epoch 40/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5185 - loss:
2.8294
Epoch 41/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.5185 - loss:
```

```
2.8236
Epoch 42/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5185 - loss:
2.8178
Epoch 43/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5185 - loss:
2.8118
Epoch 44/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5185 - loss:
2.8058
Epoch 45/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.5556 - loss:
2.7996
Epoch 46/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.5556 - loss:
2.7933
Epoch 47/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5556 - loss:
2.7869
Epoch 48/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.5556 - loss:
2.7804
Epoch 49/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.5556 - loss:
2.7738
Epoch 50/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5556 - loss:
2.7671
Epoch 51/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.5556 - loss:
2.7602
Epoch 52/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.5556 - loss:
2.7533
Epoch 53/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.5556 - loss:
2.7462
Epoch 54/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - accuracy: 0.5556 - loss:
2.7390
Epoch 55/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.5556 - loss:
2.7316
Epoch 56/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.5556 - loss:
2.7242
Epoch 57/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.5556 - loss:
2.7166
```

```
Epoch 58/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.6296 - loss:
2.7089
Epoch 59/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.6296 - loss:
2.7010
Epoch 60/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.6296 - loss:
2.6930
Epoch 61/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6296 - loss:
2.6849
Epoch 62/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6296 - loss:
2.6767
Epoch 63/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6296 - loss:
2.6683
Epoch 64/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6296 - loss:
2.6598
Epoch 65/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.6296 - loss:
2.6511
Epoch 66/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6296 - loss:
2.6424
Epoch 67/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.6296 - loss:
2.6334
Epoch 68/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6667 - loss:
2.6244
Epoch 69/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.6667 - loss:
2.6151
Epoch 70/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.6667 - loss:
2.6058
Epoch 71/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.6667 - loss:
2.5963
Epoch 72/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6667 - loss:
2.5867
Epoch 73/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.6667 - loss:
2.5769
Epoch 74/100
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.6667 - loss:
2.5670
Epoch 75/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.6667 - loss:
2.5569
Epoch 76/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.6667 - loss:
2.5467
Epoch 77/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.6667 - loss:
2.5363
Epoch 78/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - accuracy: 0.6667 - loss:
2.5258
Epoch 79/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.6667 - loss:
2.5152
Epoch 80/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.6667 - loss:
2.5044
Epoch 81/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.6667 - loss:
2.4934
Epoch 82/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.7037 - loss:
2.4823
Epoch 83/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.7037 - loss:
2.4711
Epoch 84/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.7037 - loss:
2.4597
Epoch 85/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.7037 - loss:
2.4482
Epoch 86/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.7037 - loss:
2.4366
Epoch 87/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.7037 - loss:
2.4247
Epoch 88/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.7037 - loss:
2.4128
Epoch 89/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.7407 - loss:
2.4007
Epoch 90/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.7407 - loss:
```

```
2.3884
Epoch 91/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.7407 - loss:
2.3761
Epoch 92/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.7407 - loss:
2.3635
Epoch 93/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - accuracy: 0.7778 - loss:
2.3509
Epoch 94/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.7778 - loss:
2.3381
Epoch 95/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.7778 - loss:
2.3251
Epoch 96/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 0.7778 - loss:
2.3121
Epoch 97/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.7778 - loss:
2.2988
Epoch 98/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.7778 - loss:
2.2855
Epoch 99/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.7778 - loss:
2.2720
Epoch 100/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.7778 - loss:
2.2584

<keras.src.callbacks.history.History at 0x7c9a917554b0>
```

```python
# Output the word embeddings
embeddings = model.layers[0].get_weights()[0]
for word, idx in word_index.items():
    print(f"Word: {word}, Embedding: {embeddings[idx]}")
```

```
Word: i, Embedding: [-0.12723807  0.0722872  -0.2590279  -0.14985624 -
0.06649505  0.03341588
  0.17615412 -0.16333963 -0.01597998  0.22771382]
Word: learning, Embedding: [-0.14798063 -0.14527999  0.19364522
0.11821359  0.12949903 -0.1491143
 -0.127835   -0.11308666 -0.14891794  0.0288964 ]
Word: love, Embedding: [-0.07334268 -0.17823648 -0.05131005
0.19873175  0.07453911  0.15947078
  0.13139799  0.06427691  0.01967752  0.11893987]
Word: programming, Embedding: [-0.00714831 -0.17457998 -0.01627276 -
0.26079333  0.1522902  -0.02547743
```

```
 -0.13122462  0.02962573 -0.16688572  0.10528577]
Word: python, Embedding: [-0.19561563 -0.06649638 -0.13120905 -
0.09030433 -0.08158397  0.1838165
 -0.05753448 -0.02663034  0.10142747  0.12950444]
Word: is, Embedding: [ 0.11794697 -0.15327705 -0.11196724 -0.19286169
0.16639006 -0.11259601
  0.08789127 -0.19433765 -0.06369892 -0.07012028]
Word: in, Embedding: [-0.12434448 -0.172447   -0.00460472 -0.16053607
-0.00789125 -0.04308063
  0.01130818 -0.08486404  0.10873087  0.05173623]
Word: enjoy, Embedding: [-0.10521057 -0.1111218  -0.23301758
0.16199657  0.11546712  0.17092848
  0.18043683  0.15380174  0.06852843  0.16962731]
Word: about, Embedding: [ 0.20476696  0.19606988  0.08025766
0.03718189 -0.11963624 -0.1676353
  0.16416214  0.15770729 -0.22431204 -0.12770456]
Word: machine, Embedding: [ 0.14834522  0.11249292 -0.1596038  -
0.06392263 -0.18525869 -0.05144744
 -0.16210373 -0.10423717 -0.00760821  0.02683414]
Word: great, Embedding: [-0.19710313  0.06647684 -0.15784794 -
0.02798615  0.16606295 -0.16229694
  0.11416657  0.14690469 -0.14053577 -0.16629192]
Word: for, Embedding: [-0.06209755  0.11437545 -0.1584993  -0.13552491
-0.14718238  0.14550593
  0.10034624 -0.09183014  0.16233836  0.17414947]
Word: data, Embedding: [-0.02263114 -0.15314606  0.10385013  0.2162584
-0.12426104  0.16376813
  0.10046268  0.06270266 -0.05159009  0.08287404]
Word: science, Embedding: [ 0.01644773  0.07531504  0.08737639
0.15103568 -0.08860677  0.10496236
 -0.02813444  0.13077521  0.03885876 -0.1874421 ]
Word: new, Embedding: [ 0.0503987   0.09917612  0.14667276 -0.20742428
-0.11677802 -0.10531148
 -0.14967465 -0.05372128 -0.0971799   0.05877364]
Word: technologies, Embedding: [ 0.02865334 -0.13208233 -0.16005892 -
0.14278245 -0.00830435 -0.13470733
  0.04711578  0.05848897  0.10630768 -0.02176987]
Word: fun, Embedding: [-0.07761869  0.04991721 -0.00350431  0.14390838
-0.01755077 -0.0802723
 -0.0481532   0.22755125  0.08754651 -0.01487658]
Word: and, Embedding: [ 0.21613356  0.15283169  0.00670594  0.05565186
-0.09094392  0.09545989
  0.18609893 -0.10900354  0.12888223  0.09090495]
Word: rewarding, Embedding: [ 0.03252169  0.03653452  0.14169812 -
0.03463521  0.1636323   0.15355764
  0.07141843  0.03318454 -0.12099437 -0.00792441]
```