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

a. Data preparation
b. Generate training data
c. Train model
d. Output

```
import numpy as np
import re
```

```
data = """Deep learning (also known as deep structured learning) is part of a b
data
```

```
'Deep learning (also known as deep structured learning) is part of a broader
family of machine learning methods based on artificial neural networks with
representation learning. Learning can be supervised, semi-supervised or
unsupervised. Deep-learning architectures such as deep neural networks, deep
belief networks, deep reinforcement learning, recurrent neural networks,
convolutional neural networks and Transformers have been applied to fields
including computer vision, speech recognition, natural language processing,
machine translation, bioinformatics, drug design, medical image analysis,
climate science, material inspection and board game programs, where they have
produced results comparable to and in some cases surpassing human expert
performance.'
```

```
sentences = data.split('.')
sentences
```

```
['Deep learning (also known as deep structured learning) is part of a broader
family of machine learning methods based on artificial neural networks with
representation learning',
 ' Learning can be supervised, semi-supervised or unsupervised',
 ' Deep-learning architectures such as deep neural networks, deep belief
networks, deep reinforcement learning, recurrent neural networks, convolutional
neural networks and Transformers have been applied to fields including computer
vision, speech recognition, natural language processing, machine translation,
bioinformatics, drug design, medical image analysis, climate science, material
inspection and board game programs, where they have produced results comparable
to and in some cases surpassing human expert performance',
 '']
```

```
clean_sent=[]
for sentence in sentences:
    if sentence=="":
        continue
    sentence = re.sub('[^A-Za-z0-9]+', ' ', (sentence))
    sentence = re.sub(r'(?:^| )\w (?:$| )', ' ', (sentence)).strip()
    sentence = sentence.lower()
    clean_sent.append(sentence)

clean_sent
```

```
['deep learning also known as deep structured learning is part of a broader
family of machine learning methods based on artificial neural networks with
representation learning',
 'learning can be supervised semi supervised or unsupervised',
 'deep learning architectures such as deep neural networks deep belief networks
deep reinforcement learning recurrent neural networks convolutional neural
networks and transformers have been applied to fields including computer vision
speech recognition natural language processing machine translation
bioinformatics drug design medical image analysis climate science material
inspection and board game programs where they have produced results comparable
to and in some cases surpassing human expert performance']
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(clean_sent)
sequences = tokenizer.texts_to_sequences(clean_sent)
print(sequences)
```

```
[[2, 1, 12, 13, 6, 2, 14, 1, 15, 16, 7, 17, 18, 19, 7, 8, 1, 20, 21, 22, 23, 4,
```

```
index_to_word = {}
word_to_index = {}

for i, sequence in enumerate(sequences):
#     print(sequence)
    word_in_sentence = clean_sent[i].split()
#     print(word_in_sentence)

    for j, value in enumerate(sequence):
        index_to_word[value] = word_in_sentence[j]
        word_to_index[word_in_sentence[j]] = value

print(index_to_word, "\n")
print(word_to_index)
```

```
{2: 'deep', 1: 'learning', 12: 'also', 13: 'known', 6: 'as', 14: 'structured', 1

{'deep': 2, 'learning': 1, 'also': 12, 'known': 13, 'as': 6, 'structured': 14, '
```

```
vocab_size = len(tokenizer.word_index) + 1
emb_size = 10
context_size = 2

contexts = []
targets = []

for sequence in sequences:
    for i in range(context_size, len(sequence) - context_size):
        target = sequence[i]
        context = [sequence[i - 2], sequence[i - 1], sequence[i + 1], sequence[
#         print(context)
        contexts.append(context)
        targets.append(target)
print(contexts, "\n")
print(targets)
```

```
[[2, 1, 13, 6], [1, 12, 6, 2], [12, 13, 2, 14], [13, 6, 14, 1], [6, 2, 1, 15], [

[12, 13, 6, 2, 14, 1, 15, 16, 7, 17, 18, 19, 7, 8, 1, 20, 21, 22, 23, 4, 3, 24,
```

```
#printing features with target
for i in range(5):
    words = []
    target = index_to_word.get(targets[i])
    for j in contexts[i]:
        words.append(index_to_word.get(j))
    print(words," -> ", target)
```

```
['deep', 'learning', 'known', 'as']  ->  also
['learning', 'also', 'as', 'deep']  ->  known
['also', 'known', 'deep', 'structured']  ->  as
['known', 'as', 'structured', 'learning']  ->  deep
['as', 'deep', 'learning', 'is']  ->  structured
```

```
# Convert the contexts and targets to numpy arrays
X = np.array(contexts)
Y = np.array(targets)
```

```
# print(X)
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, Lambda
```

```
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=emb_size, input_length=2*context
    Lambda(lambda x: tf.reduce_mean(x, axis=1)),
    Dense(256, activation='relu'),
    Dense(512, activation='relu'),
    Dense(vocab_size, activation='softmax')
])
```

```
C:\Users\Parth\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\s
  warnings.warn(
```

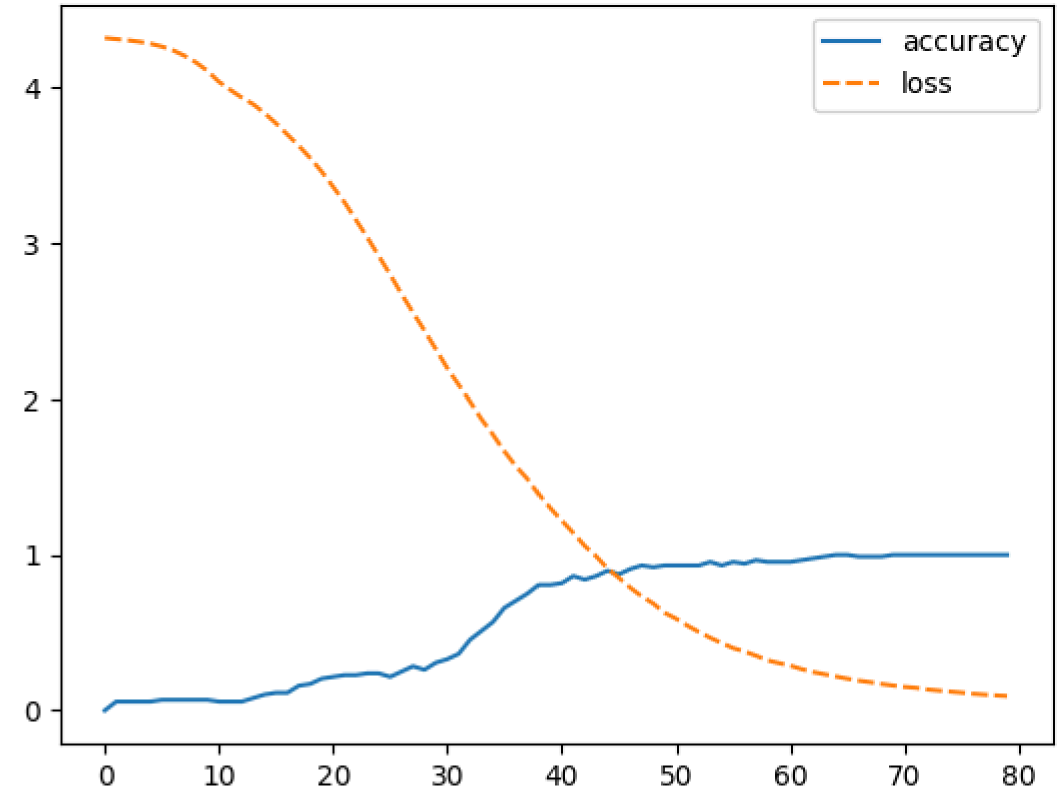
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics
history = model.fit(X, Y, epochs=80)
```

```
Epoch 1/80
WARNING:tensorflow:From C:\Users\Parth\AppData\Local\Programs\Python\Python312

3/3 ━━━━━━━━━━━━━━━━━━━━ 39s 82ms/step - accuracy: 0.0000e+00 - loss: 4.3187
Epoch 2/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.0568 - loss: 4.3114
Epoch 3/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.0568 - loss: 4.3042
Epoch 4/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.0568 - loss: 4.2947
Epoch 5/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.0568 - loss: 4.2823
Epoch 6/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 87ms/step - accuracy: 0.0682 - loss: 4.2634
Epoch 7/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.0682 - loss: 4.2387
Epoch 8/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.0682 - loss: 4.2036
Epoch 9/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.0682 - loss: 4.1612
Epoch 10/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.0682 - loss: 4.1069
Epoch 11/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.0568 - loss: 4.0370
Epoch 12/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - accuracy: 0.0568 - loss: 3.9851
Epoch 13/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step - accuracy: 0.0568 - loss: 3.9354
Epoch 14/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step - accuracy: 0.0795 - loss: 3.8930
Epoch 15/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - accuracy: 0.1023 - loss: 3.8352
Epoch 16/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 178ms/step - accuracy: 0.1136 - loss: 3.7699
Epoch 17/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 93ms/step - accuracy: 0.1136 - loss: 3.6975
Epoch 18/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 68ms/step - accuracy: 0.1591 - loss: 3.6255
Epoch 19/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - accuracy: 0.1705 - loss: 3.5467
Epoch 20/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step - accuracy: 0.2045 - loss: 3.4600
Epoch 21/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.2159 - loss: 3.3625
Epoch 22/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - accuracy: 0.2273 - loss: 3.2625
Epoch 23/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - accuracy: 0.2273 - loss: 3.1498
Epoch 24/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 87ms/step - accuracy: 0.2386 - loss: 3.0366
Epoch 25/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 169ms/step - accuracy: 0.2386 - loss: 2.9175
Epoch 26/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 104ms/step - accuracy: 0.2159 - loss: 2.7982
Epoch 27/80
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step - accuracy: 0.2500 - loss: 2.6726
Epoch 28/80
```

```
import seaborn as sns
sns.lineplot(model.history.history)
```

```
<Axes: >
```



```
from sklearn.decomposition import PCA

embeddings = model.get_weights()[0]

pca = PCA(n_components=2)
reduced_embeddings = pca.fit_transform(embeddings)
```

```
print("'Deep learning (also known as deep structured learning) is part of a brc
```

```
'Deep learning (also known as deep structured learning) is part of a broader fam
```

```
# test model: select some sentences from above paragraph
test_sentenses = [
    "known as structured learning",
    "transformers have applied to",
    "where they produced results",
    "cases surpassing expert performance"
]
```

```
for sent in test_sentenses:
    test_words = sent.split(" ")
#     print(test_words)
    x_test =[]
    for i in test_words:
        x_test.append(word_to_index.get(i))
    x_test = np.array([x_test])
#     print(x_test)

    pred = model.predict(x_test)
    pred = np.argmax(pred[0])
    print("pred ", test_words, "\n=", index_to_word.get(pred),"\n\n")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 377ms/step
pred  ['known', 'as', 'structured', 'learning']
= deep


1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step
pred  ['transformers', 'have', 'applied', 'to']
= been


1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
pred  ['where', 'they', 'produced', 'results']
= have


1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
pred  ['cases', 'surpassing', 'expert', 'performance']
= human
```

Start coding or generate with AI.