In [3]:

```
# Assignment 5: Continuous Bag of Words (CBOW) Model

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Embedding, Flatten
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.text import Tokenizer
from sklearn.metrics.pairwise import cosine_similarity

# =====================
# 1. Sample Text
# =====================
corpus = [
    "Deep learning is a subfield of machine learning",
    "Word embeddings capture semantic meaning of words",
    "CBOW model predicts target word using context words"
]

# =====================
# 2. Tokenization
# =====================
tokenizer = Tokenizer()
tokenizer.fit_on_texts(corpus)
word2id = tokenizer.word_index
id2word = {v: k for k, v in word2id.items()}
vocab_size = len(word2id) + 1

print("Vocabulary:", word2id)

# Convert text to sequences
sequences = tokenizer.texts_to_sequences(corpus)

# =====================
# 3. Generate CBOW Training Data
# =====================
window_size = 2
X, y = [], []

for seq in sequences:
    for idx, word in enumerate(seq):
        context = []
        for neighbor in range(-window_size, window_size + 1):
            if neighbor == 0:
                continue
            pos = idx + neighbor
            if pos >= 0 and pos < len(seq):
                context.append(seq[pos])
        if len(context) > 0:
            X.append(np.mean(context))  # simplified CBOW context
            y.append(word)

X = np.array(X).reshape(-1, 1).astype("int32")
y = to_categorical(y, vocab_size)

# =====================
# 4. Define CBOW Model
# =====================
embedding_dim = 50
```

```python
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=1),
    Flatten(),
    Dense(vocab_size, activation="softmax")
])

model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accur
model.summary()

# =====================
# 5. Train Model
# =====================
model.fit(X, y, epochs=100, verbose=0)

# =====================
# 6. Extract Word Embeddings
# =====================
weights = model.get_weights()[0]
print("Embedding matrix shape:", weights.shape)

# =====================
# 7. Test Similarity
# =====================
def most_similar(word, top_n=3):
    if word not in word2id:
        return []
    idx = word2id[word]
    vec = weights[idx].reshape(1, -1)
    sims = cosine_similarity(vec, weights)[0]
    similar_ids = sims.argsort()[-top_n-1:][::-1]
    return [id2word[i] for i in similar_ids if i in id2word and i != idx]

print("Most similar to 'learning':", most_similar("learning"))
print("Most similar to 'model':", most_similar("model"))
```

```
Vocabulary: {'learning': 1, 'of': 2, 'word': 3, 'words': 4, 'deep': 5, 'is': 6,
'a': 7, 'subfield': 8, 'machine': 9, 'embeddings': 10, 'capture': 11, 'semantic':
12, 'meaning': 13, 'cbow': 14, 'model': 15, 'predicts': 16, 'target': 17, 'usin
g': 18, 'context': 19}
```

```
C:\Users\aryan\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\sr
c\layers\core\embedding.py:97: UserWarning: Argument `input_length` is deprecate
d. Just remove it.
  warnings.warn(
```

Model: "sequential"

Layer (type)	Output Shape	
embedding (Embedding)	?	
flatten (Flatten)	?	
dense (Dense)	?	

Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

Non-trainable params: 0 (0.00 B)

```
Embedding matrix shape: (20, 50)
Most similar to 'learning': ['words', 'is']
Most similar to 'model': ['of', 'predicts', 'a']
```

In [ ]: