In [4]:
```
import numpy as np
import re
```

In [5]:
```
data = """Deep learning (also known as deep structured learning) is par
data
```

Out[5]:
```
'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, se
mi-supervised or unsupervised. Deep-learning architectures such as de
ep neural networks, deep belief networks, deep reinforcement learnin
g, recurrent neural networks, convolutional neural networks and Trans
formers have been applied to fields including computer vision, speech
recognition, natural language processing, machine translation, bioinf
ormatics, drug design, medical image analysis, climate science, mater
ial inspection and board game programs, where they have produced resu
lts comparable to and in some cases surpassing human expert performan
ce.'
```

In [6]:
```
sentences = data.split('.')
sentences
```

Out[6]:
```
['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 bel
ief networks, deep reinforcement learning, recurrent neural networks,
convolutional neural networks and Transformers have been applied to f
ields including computer vision, speech recognition, natural language
processing, machine translation, bioinformatics, drug design, medical
image analysis, climate science, material inspection and board game p
rograms, where they have produced results comparable to and in some c
ases surpassing human expert performance',
 '']
```

In [7]:

```
#a data preparation
#clean data
clean_sent=[]
for sentence in sentences:
    if sentence=="": #skip empty string
        continue
    sentence = re.sub('[^A-Za-z0-9]+', ' ', (sentence)) #remove spl cha
    sentence = re.sub(r'(?:^| )\w (?:$| )', ' ', (sentence)).strip() #r
    sentence = sentence.lower() #lower all characters
    clean_sent.append(sentence)

clean_sent
```

Out[7]:

```
['deep learning also known as deep structured learning is part of a b
roader 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 belie
f networks deep reinforcement learning recurrent neural networks conv
olutional neural networks and transformers have been applied to field
s including computer vision speech recognition natural language proce
ssing machine translation bioinformatics drug design medical image an
alysis climate science material inspection and board game programs wh
ere they have produced results comparable to and in some cases surpas
sing human expert performance']
```

In [15]:

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

In [17]:

```
#convert the clean sentences to a sequence of integers
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, 3, 24, 25, 1], [1, 26, 27, 9, 28, 9, 29, 30], [2, 1, 31, 3
2, 6, 2, 4, 3, 2, 33, 3, 2, 34, 1, 35, 4, 3, 36, 4, 3, 5, 37, 10, 38,
39, 11, 40, 41, 42, 43, 44, 45, 46, 47, 48, 8, 49, 50, 51, 52, 53, 5
4, 55, 56, 57, 58, 59, 5, 60, 61, 62, 63, 64, 10, 65, 66, 67, 11, 5,
68, 69, 70, 71, 72, 73, 74]]
```

In [18]:

```
#createing dictionary for word to index and index to word
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: 'str
uctured', 15: 'is', 16: 'part', 7: 'of', 17: 'a', 18: 'broader', 19:
'family', 8: 'machine', 20: 'methods', 21: 'based', 22: 'on', 23: 'ar
tificial', 4: 'neural', 3: 'networks', 24: 'with', 25: 'representatio
n', 26: 'can', 27: 'be', 9: 'supervised', 28: 'semi', 29: 'or', 30:
'unsupervised', 31: 'architectures', 32: 'such', 33: 'belief', 34: 'r
einforcement', 35: 'recurrent', 36: 'convolutional', 5: 'and', 37: 't
ransformers', 10: 'have', 38: 'been', 39: 'applied', 11: 'to', 40: 'f
ields', 41: 'including', 42: 'computer', 43: 'vision', 44: 'speech',
45: 'recognition', 46: 'natural', 47: 'language', 48: 'processing', 4
9: 'translation', 50: 'bioinformatics', 51: 'drug', 52: 'design', 53:
'medical', 54: 'image', 55: 'analysis', 56: 'climate', 57: 'science',
58: 'material', 59: 'inspection', 60: 'board', 61: 'game', 62: 'progr
ams', 63: 'where', 64: 'they', 65: 'produced', 66: 'results', 67: 'co
mparable', 68: 'in', 69: 'some', 70: 'cases', 71: 'surpassing', 72:
'human', 73: 'expert', 74: 'performance'}

{'deep': 2, 'learning': 1, 'also': 12, 'known': 13, 'as': 6, 'structu
red': 14, 'is': 15, 'part': 16, 'of': 7, 'a': 17, 'broader': 18, 'fam
ily': 19, 'machine': 8, 'methods': 20, 'based': 21, 'on': 22, 'artifi
cial': 23, 'neural': 4, 'networks': 3, 'with': 24, 'representation':
25, 'can': 26, 'be': 27, 'supervised': 9, 'semi': 28, 'or': 29, 'unsu
pervised': 30, 'architectures': 31, 'such': 32, 'belief': 33, 'reinfo
rcement': 34, 'recurrent': 35, 'convolutional': 36, 'and': 5, 'transf
ormers': 37, 'have': 10, 'been': 38, 'applied': 39, 'to': 11, 'field
s': 40, 'including': 41, 'computer': 42, 'vision': 43, 'speech': 44,
'recognition': 45, 'natural': 46, 'language': 47, 'processing': 48,
'translation': 49, 'bioinformatics': 50, 'drug': 51, 'design': 52, 'm
edical': 53, 'image': 54, 'analysis': 55, 'climate': 56, 'science': 5
7, 'material': 58, 'inspection': 59, 'board': 60, 'game': 61, 'progra
ms': 62, 'where': 63, 'they': 64, 'produced': 65, 'results': 66, 'com
parable': 67, 'in': 68, 'some': 69, 'cases': 70, 'surpassing': 71, 'h
uman': 72, 'expert': 73, 'performance': 74}
```

In [19]:

```
#b. generating training data

# Define the parameters
vocab_size = len(tokenizer.word_index) + 1
emb_size = 10
context_size = 2

#generate the context target pairs
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], s
#         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], [2, 14, 15, 16], [14, 1, 16, 7], [1, 15, 7, 17], [15, 16,
17, 18], [16, 7, 18, 19], [7, 17, 19, 7], [17, 18, 7, 8], [18, 19, 8,
1], [19, 7, 1, 20], [7, 8, 20, 21], [8, 1, 21, 22], [1, 20, 22, 23],
[20, 21, 23, 4], [21, 22, 4, 3], [22, 23, 3, 24], [23, 4, 24, 25],
[4, 3, 25, 1], [1, 26, 9, 28], [26, 27, 28, 9], [27, 9, 9, 29], [9, 2
8, 29, 30], [2, 1, 32, 6], [1, 31, 6, 2], [31, 32, 2, 4], [32, 6, 4,
3], [6, 2, 3, 2], [2, 4, 2, 33], [4, 3, 33, 3], [3, 2, 3, 2], [2, 33,
2, 34], [33, 3, 34, 1], [3, 2, 1, 35], [2, 34, 35, 4], [34, 1, 4, 3],
[1, 35, 3, 36], [35, 4, 36, 4], [4, 3, 4, 3], [3, 36, 3, 5], [36, 4,
5, 37], [4, 3, 37, 10], [3, 5, 10, 38], [5, 37, 38, 39], [37, 10, 39,
11], [10, 38, 11, 40], [38, 39, 40, 41], [39, 11, 41, 42], [11, 40, 4
2, 43], [40, 41, 43, 44], [41, 42, 44, 45], [42, 43, 45, 46], [43, 4
4, 46, 47], [44, 45, 47, 48], [45, 46, 48, 8], [46, 47, 8, 49], [47,
48, 49, 50], [48, 8, 50, 51], [8, 49, 51, 52], [49, 50, 52, 53], [50,
51, 53, 54], [51, 52, 54, 55], [52, 53, 55, 56], [53, 54, 56, 57], [5
4, 55, 57, 58], [55, 56, 58, 59], [56, 57, 59, 5], [57, 58, 5, 60],
[58, 59, 60, 61], [59, 5, 61, 62], [5, 60, 62, 63], [60, 61, 63, 64],
[61, 62, 64, 10], [62, 63, 10, 65], [63, 64, 65, 66], [64, 10, 66, 6
7], [10, 65, 67, 11], [65, 66, 11, 5], [66, 67, 5, 68], [67, 11, 68,
69], [11, 5, 69, 70], [5, 68, 70, 71], [68, 69, 71, 72], [69, 70, 72,
73], [70, 71, 73, 74]]

[12, 13, 6, 2, 14, 1, 15, 16, 7, 17, 18, 19, 7, 8, 1, 20, 21, 22, 23,
4, 3, 24, 27, 9, 28, 9, 31, 32, 6, 2, 4, 3, 2, 33, 3, 2, 34, 1, 35,
4, 3, 36, 4, 3, 5, 37, 10, 38, 39, 11, 40, 41, 42, 43, 44, 45, 46, 4
7, 48, 8, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 5, 60, 61, 62,
63, 64, 10, 65, 66, 67, 11, 5, 68, 69, 70, 71, 72]
```

In [20]:

```
#printing features with target with sample of training data
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
```

In [21]:

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

In [22]:

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

In [23]:

```
#c. train model

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

```
WARNING:tensorflow:From C:\Users\shrey\AppData\Roaming\Python\Python3
11\site-packages\keras\src\backend.py:873: The name tf.get_default_gr
aph is deprecated. Please use tf.compat.v1.get_default_graph instead.
```

In [27]:

```
# Compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
```

In [28]:

```
# Train the model
history = model.fit(X, Y, epochs=80)
```

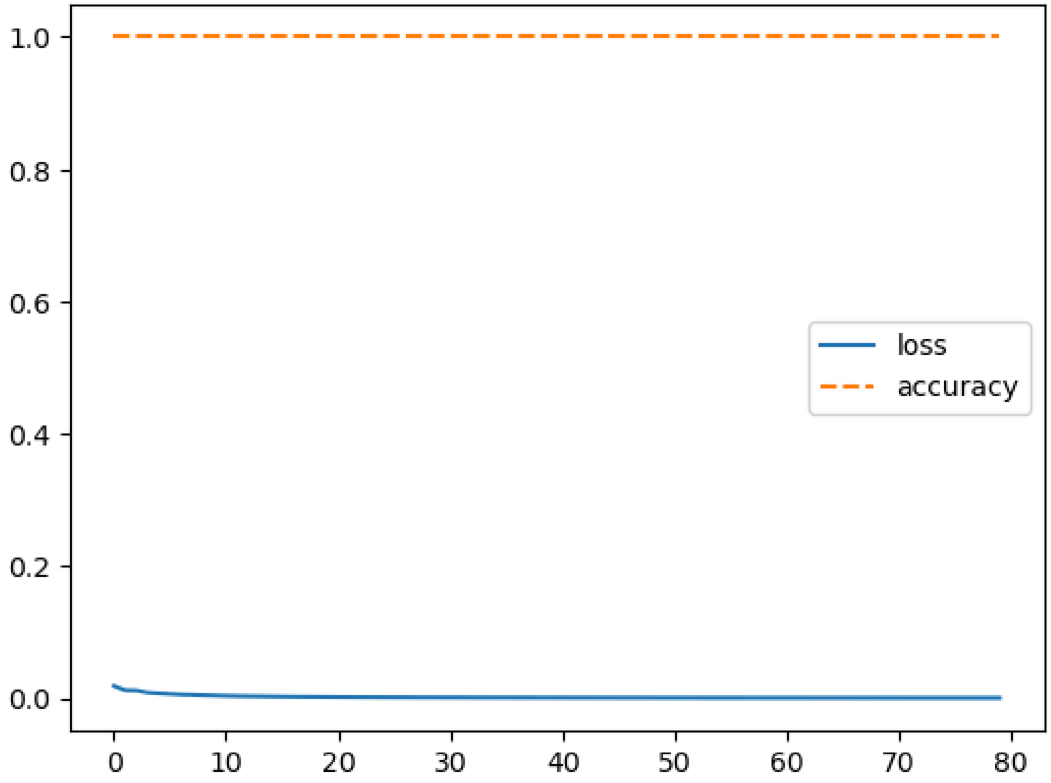
```
Epoch 1/80
3/3 [==============================] - 1s 7ms/step - loss: 0.0189
- accuracy: 1.0000
Epoch 2/80
3/3 [==============================] - 0s 11ms/step - loss: 0.0123
- accuracy: 1.0000
Epoch 3/80
3/3 [==============================] - 0s 9ms/step - loss: 0.0118
- accuracy: 1.0000
Epoch 4/80
3/3 [==============================] - 0s 7ms/step - loss: 0.0086
- accuracy: 1.0000
Epoch 5/80
3/3 [==============================] - 0s 7ms/step - loss: 0.0075
- accuracy: 1.0000
Epoch 6/80
3/3 [==============================] - 0s 7ms/step - loss: 0.0065
- accuracy: 1.0000
Epoch 7/80
```

In [29]:

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

Out[29]:

```
<Axes: >
```



In [30]:

```
from sklearn.decomposition import PCA
# Get the word embeddings
embeddings = model.get_weights()[0]

# Perform PCA to reduce the dimensionality of the embeddings
pca = PCA(n_components=2)
reduced_embeddings = pca.fit_transform(embeddings)
```

In [31]:

```
#d. output

# 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"
]
```

In [32]:

```
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 182ms/step
pred  ['known', 'as', 'structured', 'learning']
= deep


1/1 [==============================] - 0s 40ms/step
pred  ['transformers', 'have', 'applied', 'to']
= been


1/1 [==============================] - 0s 37ms/step
pred  ['where', 'they', 'produced', 'results']
= have


1/1 [==============================] - 0s 28ms/step
pred  ['cases', 'surpassing', 'expert', 'performance']
= human
```

In [ ]:

```
```