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

```
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 representa
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, 3, 24, 25, 1], [1, 26, 27, 9, 28, 9, 29, 30], [2, 1, 31, 32, 6, 2, 4, 3, 2, 33, 3, 2, 34, 1,
```

```
index_to_word = {}
word_to_index = {}

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


    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', 15: 'is', 16: 'part', 7: 'of', 17: 'a', 18: 'broader', 19: 'family', 8: 'machine', 20: 'method

{'deep': 2, 'learning': 1, 'also': 12, 'known': 13, 'as': 6, 'structured': 14, 'is': 15, 'part': 16, 'of': 7, 'a': 17, 'broader': 18, 'family': 19, 'machine': 8, 'methods':
```

```
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[i + 2]]

        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

[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,
```

```
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
```

```
X = np.array(contexts)
Y = np.array(targets)
```

```
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_size),
    Lambda(lambda x: tf.reduce_mean(x, axis=1)),
    Dense(256, activation='relu'),
    Dense(512, activation='relu'),
    Dense(vocab_size, activation='softmax')
])
```

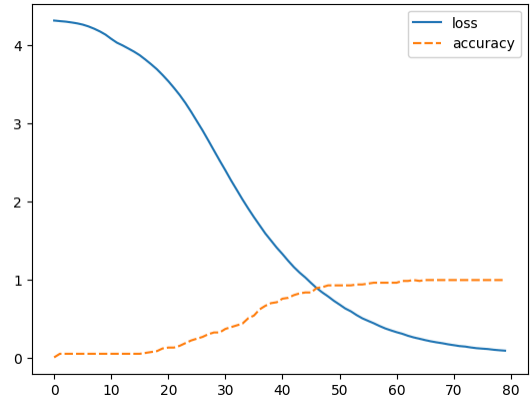
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

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

```
Epoch 52/80
3/3 [==============================] - 0s 10ms/step - loss: 0.6366 - accuracy: 0.9318
Epoch 53/80
3/3 [==============================] - 0s 10ms/step - loss: 0.5992 - accuracy: 0.9318
Epoch 54/80
3/3 [==============================] - 0s 8ms/step - loss: 0.5516 - accuracy: 0.9432
Epoch 55/80
3/3 [==============================] - 0s 8ms/step - loss: 0.5104 - accuracy: 0.9432
Epoch 56/80
3/3 [==============================] - 0s 8ms/step - loss: 0.4783 - accuracy: 0.9545
Epoch 57/80
3/3 [==============================] - 0s 12ms/step - loss: 0.4471 - accuracy: 0.9659
Epoch 58/80
3/3 [==============================] - 0s 8ms/step - loss: 0.4126 - accuracy: 0.9659
Epoch 59/80
3/3 [==============================] - 0s 12ms/step - loss: 0.3814 - accuracy: 0.9659
Epoch 60/80
3/3 [==============================] - 0s 10ms/step - loss: 0.3564 - accuracy: 0.9659
Epoch 61/80
3/3 [==============================] - 0s 8ms/step - loss: 0.3330 - accuracy: 0.9659
Epoch 62/80
3/3 [==============================] - 0s 8ms/step - loss: 0.3126 - accuracy: 0.9886
Epoch 63/80
3/3 [==============================] - 0s 12ms/step - loss: 0.2871 - accuracy: 0.9886
Epoch 64/80
3/3 [==============================] - 0s 12ms/step - loss: 0.2662 - accuracy: 1.0000
Epoch 65/80
3/3 [==============================] - 0s 12ms/step - loss: 0.2495 - accuracy: 0.9886
Epoch 66/80
3/3 [==============================] - 0s 20ms/step - loss: 0.2320 - accuracy: 1.0000
Epoch 67/80
3/3 [==============================] - 0s 10ms/step - loss: 0.2162 - accuracy: 1.0000
Epoch 68/80
3/3 [==============================] - 0s 8ms/step - loss: 0.2032 - accuracy: 1.0000
Epoch 69/80
3/3 [==============================] - 0s 10ms/step - loss: 0.1921 - accuracy: 1.0000
Epoch 70/80
3/3 [==============================] - 0s 10ms/step - loss: 0.1782 - accuracy: 1.0000
Epoch 71/80
3/3 [==============================] - 0s 13ms/step - loss: 0.1672 - accuracy: 1.0000
Epoch 72/80
3/3 [==============================] - 0s 8ms/step - loss: 0.1557 - accuracy: 1.0000
Epoch 73/80
3/3 [==============================] - 0s 16ms/step - loss: 0.1493 - accuracy: 1.0000
Epoch 74/80
3/3 [==============================] - 0s 16ms/step - loss: 0.1378 - accuracy: 1.0000
Epoch 75/80
3/3 [==============================] - 0s 8ms/step - loss: 0.1286 - accuracy: 1.0000
Epoch 76/80
3/3 [==============================] - 0s 8ms/step - loss: 0.1228 - accuracy: 1.0000
Epoch 77/80
3/3 [==============================] - 0s 9ms/step - loss: 0.1176 - accuracy: 1.0000
Epoch 78/80
3/3 [==============================] - 0s 12ms/step - loss: 0.1086 - accuracy: 1.0000
Epoch 79/80
3/3 [==============================] - 0s 12ms/step - loss: 0.1022 - accuracy: 1.0000
Epoch 80/80
3/3 [==============================] - 0s 16ms/step - loss: 0.0965 - accuracy: 1.0000
```

```
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 broader family of machine learning methods based on artificial neural networks with representati
```

```
'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 learn
```

```
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(" ")

    x_test =[]
    for i in test_words:
        x_test.append(word_to_index.get(i))
    x_test = np.array([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 66ms/step
pred  ['known', 'as', 'structured', 'learning']
= deep


1/1 [==============================] - 0s 53ms/step
pred  ['transformers', 'have', 'applied', 'to']
= been


1/1 [==============================] - 0s 52ms/step
pred  ['where', 'they', 'produced', 'results']
= have


1/1 [==============================] - 0s 55ms/step
pred  ['cases', 'surpassing', 'expert', 'performance']
= human
```