In [1]:
```
from keras.preprocessing import text
from keras.preprocessing import sequence
from keras.utils import pad_sequences
from keras.utils import to_categorical
import numpy as np
import pandas as pd
```

In [2]:
```
data = """Deep learning (also known as deep structured learning) is part of
↪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 netwo
↪deep reinforcement learning, recurrent neural networks, convolutional neu
↪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.
"""
dl_data = data.split()
```

In [3]:
```
tokenizer = text.Tokenizer()
tokenizer.fit_on_texts(dl_data)
word2id = tokenizer.word_index
word2id['PAD'] = 0
id2word = {v:k for k, v in word2id.items()}
wids = [[word2id[w] for w in text.text_to_word_sequence(doc)] for doc in dl
vocab_size = len(word2id)
embed_size = 100
window_size = 2
print('Vocabulary Size:', vocab_size)
print('Vocabulary Sample:', list(word2id.items())[:10])
```

```
Vocabulary Size: 81
Vocabulary Sample: [('learning', 1), ('deep', 2), ('networks', 3), ('an
d', 4), ('as', 5), ('of', 6), ('neural␣', 7), ('↪networks', 8), ('supervi
sed', 9), ('␣', 10)]
```

In [5]:
```
def generate_context_word_pairs(corpus, window_size, vocab_size):
    context_length = window_size*2
    for words in corpus:
        sentence_length = len(words)
        for index, word in enumerate(words):
            context_words = []
            label_word = []
            start = index - window_size
            end = index + window_size + 1
            context_words.append([words[i]
                        for i in range(start, end)
                        if 0 <= i < sentence_length
                        and i != index])
            label_word.append(word)
            x = pad_sequences(context_words, maxlen=context_length)
            y = to_categorical(label_word, vocab_size)
            yield (x, y)
i = 0
for x, y in generate_context_word_pairs(corpus=wids, window_size=window_siz
    if 0 not in x[0]:
# print('Context (X):', [id2word[w] for w in x[0]], '-> Target (Y):',id2wor
        if i == 10:
            break
        i += 1
```

In [6]:
```
import keras.backend as K
from keras.models import Sequential
from keras.layers import Dense, Embedding, Lambda
cbow = Sequential()
cbow.add(Embedding(input_dim=vocab_size, output_dim=embed_size,input_length
cbow.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(embed_size,)))
cbow.add(Dense(vocab_size, activation='softmax'))
cbow.compile(loss='categorical_crossentropy', optimizer='rmsprop')
print(cbow.summary())
# from IPython.display import SVG
# from keras.utils.vis_utils import model_to_dot
# SVG(model_to_dot(cbow, show_shapes=True, show_layer_names=False,rankdir=
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 4, 100)            8100

 lambda (Lambda)             (None, 100)               0

 dense (Dense)               (None, 81)                8181

=================================================================
Total params: 16281 (63.60 KB)
Trainable params: 16281 (63.60 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

In [10]:
```
for epoch in range(1, 6):
    loss = 0.
    i = 0
    for x, y in generate_context_word_pairs(corpus=wids,window_size=window_
        i += 1
        loss += cbow.train_on_batch(x, y)
        if i % 100000 == 0:
            print('Processed {} (context, word) pairs'.format(i))
    print('Epoch:', epoch, '\tLoss:', loss)
    print()
```

```
Epoch: 1        Loss: 451.10684871673584

Epoch: 2        Loss: 447.54382729530334

Epoch: 3        Loss: 445.5001447200775

Epoch: 4        Loss: 443.5994474887848

Epoch: 5        Loss: 441.92649722099304
```

In [11]:
```
weights = cbow.get_weights()[0]
weights = weights[1:]
print(weights.shape)
pd.DataFrame(weights, index=list(id2word.values())[1:]).head()
```

```
(80, 100)
```

Out[11]:

	0	1	2	3	4	5	6	
deep	0.052731	0.031762	0.032208	0.005768	0.053792	0.011835	-0.031391	0.0449
networks	0.038828	-0.025172	-0.046274	-0.015622	-0.009437	-0.014444	-0.031176	-0.0058
and	-0.001400	0.003003	0.041346	0.005582	-0.020057	-0.040284	-0.014791	-0.0142
as	0.023122	0.041794	-0.039984	0.024638	0.037334	0.041932	-0.038312	-0.0002
of	-0.023991	0.034275	0.003980	0.014363	-0.029694	0.042253	0.024232	-0.0135

5 rows × 100 columns

In [13]:
```
from sklearn.metrics.pairwise import euclidean_distances
distance_matrix = euclidean_distances(weights)
print(distance_matrix.shape)
similar_words = {search_term: [id2word[idx] for idx in␣distance_matrix[word
for search_term in ['deep']}
    similar_words
```

```
  Cell In[13], line 4
    similar_words = {search_term: [id2word[idx] for idx in␣distance_matrix
[word2id[search_term]-1].argsort()[1:6]+1]
                                                               ^
SyntaxError: invalid character '␣' (U+2423)
```

In [ ]:

```
```