Word Embeddings: Hands On

In previous lecture notebooks you saw all the steps needed to train the CBOW model. This notebook will walk you through how to extract the word embedding vectors from a model.

Let's dive into it!

In [1]:
```
import numpy as np
from utils2 import get_dict
```

Before moving on, you will be provided with some variables needed for further procedures, which should be familiar by now. Also a trained CBOW model will be simulated, the corresponding weights and biases are provided:

In [2]:
```
# Define the tokenized version of the corpus
words = ['i', 'am', 'happy', 'because', 'i', 'am', 'learning']

# Define V. Remember this is the size of the vocabulary
V = 5

# Get 'word2Ind' and 'Ind2word' dictionaries for the tokenized corpus
word2Ind, Ind2word = get_dict(words)


# Define first matrix of weights
W1 = np.array([[ 0.41687358,  0.08854191, -0.23495225,  0.28320538,  0.41800106],
               [ 0.32735501,  0.22795148, -0.23951958,  0.4117634 , -0.23924344],
               [ 0.26637602, -0.23846886, -0.37770863, -0.11399446,  0.34008124]])

# Define second matrix of weights
W2 = np.array([[-0.22182064, -0.43008631,  0.13310965],
               [ 0.08476603,  0.08123194,  0.1772054 ],
               [ 0.1871551 , -0.06107263, -0.1790735 ],
               [ 0.07055222, -0.02015138,  0.36107434],
               [ 0.33480474, -0.39423389, -0.43959196]])

# Define first vector of biases
b1 = np.array([[ 0.09688219],
               [ 0.29239497],
               [-0.27364426]])

# Define second vector of biases
b2 = np.array([[ 0.0352008 ],
               [-0.36393384],
               [-0.12775555],
               [-0.34802326],
               [-0.07017815]])
```

Extracting word embedding vectors

Once you have finished training the neural network, you have three options to get word embedding vectors for the words of your vocabulary, based on the weight matrices $\mathbf{W_1}$ and/or $\mathbf{W_2}$.

Option 1: extract embedding vectors from $\mathbf{W_1}$

The first option is to take the columns of $\mathbf{W_1}$ as the embedding vectors of the words of the vocabulary, using the same order of the words as for the input and output vectors.

> Note: in this practice notebooks the values of the word embedding vectors are meaningless after a single iteration with just one training example, but here's how you would proceed after the training process is complete.

For example $\mathbf{W_1}$ is this matrix:

In [3]:
```
# Print W1
W1
```

Out[3]:
```
array([[ 0.41687358,  0.08854191, -0.23495225,  0.28320538,  0.41800106],
       [ 0.32735501,  0.22795148, -0.23951958,  0.4117634 , -0.23924344],
       [ 0.26637602, -0.23846886, -0.37770863, -0.11399446,  0.34008124]])
```

The first column, which is a 3-element vector, is the embedding vector of the first word of your vocabulary. The second column is the word embedding vector for the second word, and so on.

The first, second, etc. words are ordered as follows.

In [4]:
```
# Print corresponding word for each index within vocabulary's range
for i in range(V):
    print(Ind2word[i])
```

```
am
because
happy
i
learning
```

So the word embedding vectors corresponding to each word are:

In [5]:
```
# Loop through each word of the vocabulary
for word in word2Ind:
    # Extract the column corresponding to the index of the word in the vocabulary
    word_embedding_vector = W1[:, word2Ind[word]]
    # Print word alongside word embedding vector
    print(f'{word}: {word_embedding_vector}')
```

```
am: [0.41687358 0.32735501 0.26637602]
because: [ 0.08854191  0.22795148 -0.23846886]
happy: [-0.23495225 -0.23951958 -0.37770863]
i: [ 0.28320538  0.4117634  -0.11399446]
learning: [ 0.41800106 -0.23924344  0.34008124]
```

Option 2: extract embedding vectors from $\mathbf{W_2}$

The second option is to take $\mathbf{W_2}$ transposed, and take its columns as the word embedding vectors just like you did for $\mathbf{W_1}$.

In [6]:
```
# Print transposed W2
W2.T
```

Out[6]:
```
array([[-0.22182064,  0.08476603,  0.1871551 ,  0.07055222,  0.33480474],
       [-0.43008631,  0.08123194, -0.06107263, -0.02015138, -0.39423389],
       [ 0.13310965,  0.1772054 , -0.1790735 ,  0.36107434, -0.43959196]])
```

In [7]:
```
# Loop through each word of the vocabulary
for word in word2Ind:
    # Extract the column corresponding to the index of the word in the vocabulary
    word_embedding_vector = W2.T[:, word2Ind[word]]
    # Print word alongside word embedding vector
    print(f'{word}: {word_embedding_vector}')
```

```
am: [-0.22182064 -0.43008631  0.13310965]
because: [0.08476603 0.08123194 0.1772054 ]
happy: [ 0.1871551  -0.06107263 -0.1790735 ]
i: [ 0.07055222 -0.02015138  0.36107434]
learning: [ 0.33480474 -0.39423389 -0.43959196]
```

Option 3: extract embedding vectors from $\mathbf{W_1}$ and $\mathbf{W_2}$

The third option, which is the one you will use in this week's assignment, uses the average of $\mathbf{W_1}$ and $\mathbf{W_2^T}$.

Calculate the average of $\mathbf{W_1}$ and $\mathbf{W_2^T}$, and store the result in `W3`.

In [8]:
```
# Compute W3 as the average of W1 and W2 transposed
W3 = (W1+W2.T)/2

# Print W3
W3
```

Out[8]:
```
array([[ 0.09752647,  0.08665397, -0.02389858,  0.1768788 ,  0.3764029 ],
       [-0.05136565,  0.15459171, -0.15029611,  0.19580601, -0.31673866],
       [ 0.19974284, -0.03063173, -0.27839106,  0.12353994, -0.04975536]])
```

Expected output:

```
array([[ 0.09752647,  0.08665397, -0.02389858,  0.1768788 ,  0.3764029 ],
       [-0.05136565,  0.15459171, -0.15029611,  0.19580601, -0.31673866],
       [ 0.19974284, -0.03063173, -0.27839106,  0.12353994, -0.04975536]])
```

Extracting the word embedding vectors works just like the two previous options, by taking the columns of the matrix you've just created.

In [9]:
```
# Loop through each word of the vocabulary
for word in word2Ind:
    # Extract the column corresponding to the index of the word in the vocabulary
    word_embedding_vector = W3[:, word2Ind[word]]
    # Print word alongside word embedding vector
    print(f'{word}: {word_embedding_vector}')
```

```
am: [ 0.09752647 -0.05136565  0.19974284]
because: [ 0.08665397  0.15459171 -0.03063173]
happy: [-0.02389858 -0.15029611 -0.27839106]
i: [0.1768788  0.19580601 0.12353994]
learning: [ 0.3764029  -0.31673866 -0.04975536]
```

Now you know 3 different options to get the word embedding vectors from a model!

How this practice relates to and differs from the upcoming graded assignment

- After extracting the word embedding vectors, you will use principal component analysis (PCA) to visualize the vectors, which will enable you to perform an intrinsic evaluation of the quality of the vectors, as explained in the lecture.

Congratulations on finishing all lecture notebooks for this week!

You're now ready to take on this week's assignment!

Keep it up!

In [ ]: