**PRACTICAL: 5**

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

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

# a. Data preparation

data = """We are about to study the idea of a computational process. Computational processes are abstract beings that inhabit computers.

As they evolve, processes manipulate other abstract things called data. The evolution of conjure the spirits of the computer with our spells.""" sentences = data.split(".") sentences

['We are about to study the idea of a computational process',

'\nComputational processes are abstract beings that inhabit computers',

'\nAs they evolve, processes manipulate other abstract things called data', '\nThe evolution of a process is directed by a pattern of rules\ncalled a pr

*#Clean Data* clean\_sentences = [] **for** sentence **in** sentences: *# skip empty string* **if** sentence == "":

**continue**;

*# remove special characters*

sentence = re.sub('[^A-Za-z0-9]+', ' ', sentence)

*# remove 1 letter words*

sentence = re.sub(r'(?:^| )\w(?:$| )', ' ', sentence).strip() *# lower all characters*

sentence = sentence.lower() clean\_sentences.append(sentence) clean\_sentences

*# Define the corpus* corpus = clean\_sentences

*# Convert the corpus to a sequence of integers* tokenizer = Tokenizer() tokenizer.fit\_on\_texts(corpus)

sequences = tokenizer.texts\_to\_sequences(corpus) print("After converting our words in the corpus \ into vector of integers:") print(sequences)

After converting our words in the corpus into vector of integers:

[[4, 5, 11, 6, 12, 1, 13, 2, 7, 8], [7, 3, 5, 9, 14, 15, 16, 17], [18, 19, 20

, 3, 21, 22, 9, 23, 10, 24], [1, 25, 2, 8, 26, 27, 28, 29, 2, 30, 10, 31], [3

2, 33, 34, 6, 35, 3], [36, 37, 4, 38, 1, 39, 2, 1, 40, 41, 42, 43]]

# b. Generate training data

*# Define the parameters*

vocab\_size = len(tokenizer.word\_index) + 1 embedding\_size = 10 window\_size = 2

*# Generate the context-target pairs* contexts = [] targets = [] **for** sequence **in** sequences:

**for** i **in** range(window\_size, len(sequence) - window\_size):

context = sequence[i - window\_size:i] + sequence[i + 1:i + window

\_size + 1]

target = sequence[i]

# c. Train model

*# Define the CBOW model* model = Sequential()

model.add(Embedding(input\_dim=vocab\_size, output\_dim=embedding\_size, input\_le ngth=2 \* window\_size))

model.add(Lambda(**lambda** x: tf.reduce\_mean(x, axis=1))) model.add(Dense(256, activation='relu')) model.add(Dense(512, activation='relu'))

model.add(Dense(units=vocab\_size, activation='softmax'))

*# Compile the model*

Epoch 1/200

C:\Users\Samruddhi\AppData\Local\Programs\Python\Python311\Lib\site-packages\ keras\src\layers\core\embedding.py:90: UserWarning: Argument `input\_length` i s deprecated. Just remove it.

warnings.warn(

WARNING:tensorflow:From C:\Users\Samruddhi\AppData\Local\Programs\Python\Pyth on311\Lib\site-packages\keras\src\backend\tensorflow\core.py:204: The name tf .placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 19ms/step - accuracy: 0.0000e+00 - loss: 3.7847 Epoch 2/200

2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.1998 - loss: 3.7771

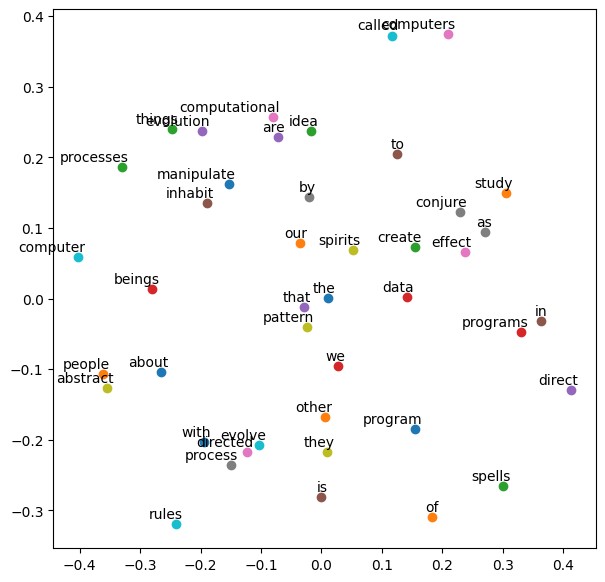
Epoch 3/200

2

Epoch 190/200

# d. Output

*# Visualize the embeddings* plt.figure(figsize=(7, 7)) **for** i, word **in** enumerate(tokenizer.word\_index.keys()): x, y = reduced\_embeddings[i] plt.scatter(y) plt.annotate(word, xy=(x, y), xytext=(5, 2), textcoords='offset points', ha='right', va='bottom') plt.show()



*# test model* test\_sentenses = [ "we are to study",

"create programs direct processes",

"spirits process study program",

"idea study people create"

]

**for** test\_sentense **in** test\_sentenses:

test\_words = test\_sentense.split(" ") print("Words: ", test\_words) x\_test = [] **for** i **in** test\_words: x\_test.append(word\_to\_index\_map.get(i)) x\_test = np.array([x\_test]) print("Indexs: ", x\_test) test\_predictions = model.predict(x\_test) y\_pred = np.argmax(test\_predictions[0])

print("Predictons: ",test\_words, " => ", index\_to\_word\_map.get(y\_pred)) print("\n")

Words: ['we', 'are', 'to', 'study']

Indexs: [[ 4 5 6 12]]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step

Predictons: ['we', 'are', 'to', 'study'] => about

Words: ['create', 'programs', 'direct', 'processes']

Indexs: [[33 34 35 3]]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step

Predictons: ['create', 'programs', 'direct', 'processes'] => to