ChatGPT

Table of Contents

- 1. Required Libraries and Installation
- 2. The Code Breakdown
  - Data Preparation:
  - Tokenization:
  - Create Dictionaries for Word Lookup:
  - Context and Target Generation for CBOW:
  - Convert Contexts and Targets to NumPy Arrays:
  - Define the CBOW Model:
  - Compile the Model:
  - Train the Model:
  - Visualize Training Loss:
  - Testing the Model:
- Summary of the Process:
- Summary of CBOW Model in Points:

this is ass 6 of dl remember it as it is with name

Got it! Your deep learning assignment 6 is now saved as it is. Let me know if you need any help with it!

5 assDl

```
import numpy as np
import re
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


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."

data




Split into sentences

sentences = data.split('.')
sentences = [sentence for sentence in sentences if
sentence.strip()]




Clean Data of Special Characters

clean_sent = []
for sentence in sentences:
    sentence = re.sub('[^A-Za-z0-9]+', ' ', sentence)
    sentence = re.sub(r'(?:^| )\\w (?:$| )', ' ', sentence).strip()
    sentence = sentence.lower()
    clean_sent.append(sentence)



Tokenization

tokenizer = Tokenizer()
tokenizer.fit_on_texts(clean_sent)
sequences = tokenizer.texts_to_sequences(clean_sent)



Create dictionaries for word lookup

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



Create dictionaries to map between word and integer IDs

vocab_size = len(tokenizer.word_index) + 1
emb_size = 100
context_size = 2



Create contexts and targets for CBOW model

contexts = []
targets = []

for sequence in sequences:
    for i in range(1, len(sequence) - 1):
        if i - context_size >= 0 and i + context_size < len(sequence):
            target = sequence[i]
            context = [sequence[i - context_size], sequence[i + context_size]]
            contexts.append(context)
            targets.append(target)



Convert contexts and targets to NumPy arrays

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



Define the CBOW model architecture

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

```
Compile the model

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




Train the model

history = model.fit(X, Y, epochs=100, batch_size=32)



Visualize training loss

embeddings = model.get_weights()[0]

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
reduced_embeddings = pca.fit_transform(embeddings)

import matplotlib.pyplot as plt
plt.scatter(reduced_embeddings[:, 0],
reduced_embeddings[:, 1])
for i, word in enumerate(index_to_word.values()):
    plt.annotate(word, (reduced_embeddings[i, 0],
reduced_embeddings[i, 1]))
plt.show()




test_sentences = [
    "known as structured learning",
    "transformers have applied to",
    "where they produced results",
    "cases surpassing expert performance"
]

for sent in test_sentences:
    test_words = sent.split(" ")
    x_test = []
    for i in test_words:
        # Add a default index for unknown words (e.g., 0 for
unknown)
        word_index = word_to_index.get(i, 0)
        x_test.append(word_index)

    # Convert the list to a NumPy array
    x_test = np.array([x_test])

    # Predict and display the result
    pred = model.predict(x_test)
    pred = np.argmax(pred[0])
    print("pred ", test_words, "\n=", index_to_word.get(pred),
"\n\n")
```

Your deep learning assignment 5 is now saved with all the details. Let me know if you need any further assistance with it!

explain everything in detail ans also tell all is required to installed in device

2/2

To implement **Assignment 5** on **Continuous Bag of Words (CBOW) Model** using TensorFlow, let's break down each part of the code and the required installations.

1. Required Libraries and Installation

To run this code, you need to install the following libraries:

1. **NumPy** - for numerical operations.
2. **TensorFlow** - for building and training the CBOW model.
3. **re** - to handle regular expressions for data cleaning.
4. **sklearn** - for Principal Component Analysis (PCA) to reduce the dimensionality of embeddings.
5. **matplotlib** - for visualizing the word embeddings.

To install these libraries, you can run the following commands:

```bash
pip install numpy tensorflow scikit-learn matplotlib
```

2. The Code Breakdown

Data Preparation:

a. Raw Data

The initial text provided is stored in the variable `data`. It contains information about deep learning.

```python
data = "Deep learning (also known as deep structured learning) is part of a broader family of machine learning..."
```

b. Split into Sentences

The text is split into sentences based on the period (`.`) character.

```python
sentences = data.split('.')
sentences = [sentence for sentence in sentences if sentence.strip()]
```

This splits the data into individual sentences, and `strip()` is used to remove any leading or trailing spaces.

c. Clean the Data

Here, special characters are removed using regular expressions, and the text is converted to lowercase.

```python
clean_sent = []
for sentence in sentences:
    sentence = re.sub('[^A-Za-z0-9]+', ' ', sentence)
    sentence = re.sub(r'(?:^| )\\w (?:$| )', ' ', sentence).strip()
    sentence = sentence.lower()
    clean_sent.append(sentence)
```

- `[^A-Za-z0-9]+` removes non-alphanumeric characters.
- `re.sub(r'(?:^| )\\w (?:$| )', ' ', sentence)` removes any one-character words (which are often just noise in natural language processing).
- The text is converted to lowercase for uniformity.

Tokenization:

The cleaned sentences are tokenized into words using the `Tokenizer` class from TensorFlow's Keras module.

```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts(clean_sent)
sequences = tokenizer.texts_to_sequences(clean_sent)
```

- `fit_on_texts()` creates a word index mapping.
- `texts_to_sequences()` converts each sentence into a sequence of word indices based on this index.

Create Dictionaries for Word Lookup:

We create two dictionaries:

- `index_to_word`: Maps word indices to actual words.
- `word_to_index`: Maps words to their corresponding indices.

```python
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
```

Context and Target Generation for CBOW:

For the **CBOW model**, we create context-target pairs from the tokenized sequences. The context consists of words surrounding a target word within a given window size.

```python
contexts = []
targets = []

for sequence in sequences:
    for i in range(1, len(sequence) - 1):
        if i - context_size >= 0 and i + context_size < len(sequence):
            target = sequence[i]
            context = [sequence[i - context_size], sequence[i + context_size]]
            contexts.append(context)
            targets.append(target)
```

- For each word in a sentence, it takes the previous and next words as context and the current word as the target. The `context_size` is set to 2, meaning we look at 2 words before and 2 words after the target.

Convert Contexts and Targets to NumPy Arrays:

We convert the contexts and targets into NumPy arrays for input into the neural network.

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

Define the CBOW Model:

The model is built using TensorFlow's Keras API.

```python
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=emb_size),
    Lambda(lambda x: tf.reduce_mean(x, axis=1)),
    Dense(256, activation='relu'),
    Dense(vocab_size, activation='softmax')
])
```

- `Embedding` : The embedding layer is used to convert words (represented as indices) into dense vectors.
- `Lambda` : This layer averages the word embeddings for context words.
- `Dense(256)` : A fully connected layer with 256 neurons and ReLU activation.
- `Dense(vocab_size)` : The output layer has a number of neurons equal to the vocabulary size, with a softmax activation function to predict the probability of each word being the target.

Compile the Model:

The model is compiled with `sparse_categorical_crossentropy` as the loss function (since this is a classification task), `adam` as the optimizer, and accuracy as the evaluation metric.

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

Train the Model:

The model is trained using the context-target pairs `X` and `Y` for 100 epochs.

```python
history = model.fit(X, Y, epochs=100, batch_size=32)
```

Visualize Training Loss:

After training, the embeddings (learned word representations) are reduced to 2D using **Principal Component Analysis (PCA)** and plotted.

```python
embeddings = model.get_weights()[0]
pca = PCA(n_components=2)
reduced_embeddings = pca.fit_transform(embeddings)

import matplotlib.pyplot as plt
plt.scatter(reduced_embeddings[:, 0], reduced_embeddings[:, 1])
for i, word in enumerate(index_to_word.values()):
    plt.annotate(word, (reduced_embeddings[i, 0], reduced_embeddings[i, 1]))
plt.show()
```

This step helps visualize the relationships between different words in the vocabulary.

Testing the Model:

The model is then tested using a list of sentences. For each test sentence, the words are converted into indices and passed into the trained model to predict the target word.

python

```
test_sentences = [
    "known as structured learning",
    "transformers have applied to",
    "where they produced results",
    "cases surpassing expert performance"
]
```

For each sentence:

1. Split the sentence into words.
2. Convert each word into its corresponding index.
3. Pass the indices to the trained model for prediction.

```python
for sent in test_sentences:
    test_words = sent.split(" ")
    x_test = []
    for i in test_words:
        word_index = word_to_index.get(i, 0)
        x_test.append(word_index)

    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")
```

- The model predicts the word corresponding to the target in the CBOW context.

Summary of the Process:

1. **Data Preprocessing:** Split the text into sentences, clean it, and tokenize it.
2. **Model Creation:** Create a CBOW model using embedding and fully connected layers.
3. **Training:** Train the model on context-target pairs.
4. **Visualization:** Visualize the word embeddings using PCA.
5. **Testing:** Test the model on new sentences to predict the target word.

Let me know if you need any further clarifications!

summarize everything in points

Summary of CBOW Model in Points:

1. **Install Required Libraries:**
    - Libraries needed: `numpy`, `tensorflow`, `scikit-learn`, `matplotlib`.
    - Install via: `pip install numpy tensorflow scikit-learn matplotlib`.
2. **Data Preparation:**

- Raw text is split into sentences.
- Sentences are cleaned by removing special characters and converting text to lowercase.

3. **Tokenization:**
    - Tokenize cleaned sentences using TensorFlow's `Tokenizer` class.
    - Convert sentences into sequences of word indices.
4. **Create Word Lookup Dictionaries:**
    - `index_to_word`: Maps word indices to words.
    - `word_to_index`: Maps words to word indices.
5. **Generate Context-Target Pairs for CBOW:**
    - Context consists of surrounding words (within a window).
    - Target is the word being predicted.
    - Example: For the sentence "Deep learning is amazing", context for "learning" might be ["Deep", "is"].
6. **Convert Contexts and Targets to NumPy Arrays:**
    - Convert the context and target pairs to NumPy arrays for model input.
7. **Define CBOW Model:**
    - Use a Sequential model with the following layers:
        - `Embedding`: Converts word indices to embeddings.
        - `Lambda`: Averages context word embeddings.
        - `Dense`: A fully connected layer for further processing.
        - `Softmax`: Final layer for classification (predict target word).
8. **Compile the Model:**
    - Loss function: `sparse_categorical_crossentropy`.
    - Optimizer: `adam`.
    - Evaluation metric: `accuracy`.
9. **Train the Model:**
    - Train the model using context-target pairs for 100 epochs.
10. **Visualize Word Embeddings:**
    - After training, reduce embeddings to 2D using PCA.
    - Plot the embeddings to visualize word relationships.
11. **Test the Model:**
    - Test on new sentences.
    - Convert words in sentences to indices, then predict the target word using the trained model.
12. **Key Points:**
    - **CBOW** predicts a target word using its surrounding context.
    - **Embedding Layer**: Maps words to dense vectors.

- **Lambda Layer**: Averages the context word embeddings.
- **Softmax Layer**: Predicts probabilities for the target word.
- **Visualization**: Use PCA to visualize word embeddings.