Lab Assignment: Learning Word Embeddings — CBOW and BERT

Objectives

1. Train static word embeddings using the Continuous Bag of Words (CBOW) model. 2. Understand the forward and backward pass in CBOW.
3. Extract contextual embeddings using BERT.
4. Compare CBOW and BERT embeddings through experiments.

Dataset

Use the Tiny Shakespeare corpus or any small English text dataset. Tokenize the text, lowercase it, and build a vocabulary.

Part 1 — Training CBOW Embeddings

1. Preprocessing Steps

• Lowercase the corpus.
• Tokenize using whitespace.
• Build a vocabulary.
• Convert each word into a one-hot vector.
• Use a context window size of 2.

2. CBOW Architecture

Input: Average of context word one-hot vectors (size V)
Hidden layer: Weight matrix W1 (V × D)
Output layer: Weight matrix W2 (D × V)
Activation: Softmax
Loss: Cross-entropy

3. Training Procedure

Students must implement:
• Context-target pair creation
• Forward pass: average one-hot → hidden vector → output probabilities • Backpropagation: compute gradients for W1 and W2
• Weight updates using gradient descent

Code Skeleton

```
import numpy as np
from collections import defaultdict

with open('tiny.txt') as f:
 text = f.read().lower().split()

vocab = list(set(text))
word2idx = {w:i for i, w in enumerate(vocab)}
```

```
idx2word = {i:w for w,i in word2idx.items()}
V = len(vocab)
D = 50
onehot = np.eye(V)

window = 2
training_data = []
for i in range(window, len(text)-window):
 target = word2idx[text[i]]
 context = [
 word2idx[text[i-2]], word2idx[text[i-1]],
 word2idx[text[i+1]], word2idx[text[i+2]]
 ]
 training_data.append((context, target))

W1 = np.random.randn(V, D) * 0.01
W2 = np.random.randn(D, V) * 0.01

def softmax(x):
 e = np.exp(x - np.max(x))
 return e / e.sum()

lr = 0.05
for context, target in training_data:
 x = np.mean(onehot[context], axis=0)
 h = x @ W1
 y = softmax(h @ W2)

 y_true = onehot[target]
 e = y - y_true

 dW2 = np.outer(h, e)
 dW1 = np.outer(x, (W2 @ e))

 W2 -= lr * dW2
 W1 -= lr * dW1
```

4. Extracting Embeddings

Word embeddings are taken as the rows of W1. Save them for analysis.

5. Experiments

- Compute cosine similarity between pairs of words.
- Use PCA or t-SNE to visualize embeddings.

• Analyze clusters such as animals, professions, numbers, etc.

Part 2 — Contextual Embeddings with BERT

Install the required libraries using: pip install transformers torch

Extracting BERT Embeddings

```
from transformers import BertTokenizer, BertModel
import torch

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

sentence = 'The bank will not issue the loan today.'
inputs = tokenizer(sentence, return_tensors='pt')
outputs = model(**inputs)
emb = outputs.last_hidden_state
```

Part 3 — Comparison: CBOW vs BERT

Students must compare embeddings of polysemous words such as 'bank' in different sentences. CBOW produces identical embeddings regardless of context, while BERT produces different embeddings depending on sentence usage.

Downstream Task

Build a small classifier using CBOW and BERT embeddings to compare performance. Example categories: finance-related sentences and nature-related sentences.

Submission Requirements

1. CBOW implementation code.
2. Visualization of CBOW embeddings.
3. Extraction of BERT embeddings.
4. Comparison report: CBOW vs BERT.
5. Explanation of findings.