Week 5 – PRACTICAL

Word2Vec and Word Embeddings

Introduction

This practical introduces word embeddings. This is the numerical representation of words that capture meaning and similarity. You will train and explore a small Word2Vec model using Gensim, understand how word relationships are formed, and observe how bias can appear in embeddings. This session builds upon the previous topics on feature extraction (BoW and TF-IDF) by moving beyond frequency counts into semantic learning.

Tools Overview

- **Gensim** – for Word2Vec model training and similarity checks
- **NLTK** – for text loading and tokenisation
- **matplotlib / sklearn.decomposition (PCA)** – for visualising embeddings
- **Dataset:** NLTK's `Reuters` corpus (subset of English news text)

Part A – Core Tasks (Complete in Class)

Q1). Load and Inspect the Dataset

```
import nltk
nltk.download('reuters')
nltk.download('punkt')
from nltk.corpus import reuters

# Load a manageable subset (first 500 docs)
sentences = [nltk.word_tokenize(reuters.raw(fid).lower())
             for fid in reuters.fileids()[:500]]

print("Number of documents:", len(sentences))
print("Example tokens:", sentences[0][:15])
```

Reflection:

- How large is the dataset you loaded?
- Do you notice anything about the kind of language (topics, style)?

Q2). Train a Word2Vec Model

```
from gensim.models import Word2Vec

model = Word2Vec(
    sentences,
    vector_size=100,
    window=5,
    min_count=1,    # keep all words for small classroom corpora
    sg=1,           # 1 = Skip-gram, 0 = CBOW
    epochs=10
)
```

- What do `vector_size`, `window`, and `sg` represent?
- Which setting (CBOW or Skip-gram) might work better for small datasets?

Q3). Inspect Vocabulary

```
print("Vocabulary size:", len(model.wv))
print("Sample vocab terms:", list(model.wv.key_to_index.keys())[:20])
```

Reflection:

- Which kinds of words dominate the vocabulary?
- Are they mainly nouns, verbs, or function words?

Q4). Explore Word Similarities

```
def in_vocab(m, w):
    return w in m.wv.key_to_index

def safe_most_similar(m, word, topn=10):
    return m.wv.most_similar(word, topn=topn) if in_vocab(m, word) else f"'{word}' not in vocabulary."

def safe_similarity(m, w1, w2):
    miss = [w for w in (w1, w2) if not in_vocab(m, w)]
    return f"Missing tokens: {miss}" if miss else m.wv.similarity(w1, w2)

print(safe_most_similar(model, "bank"))
print(safe_most_similar(model, "oil"))
print(safe_similarity(model, "money", "currency"))
```

Reflection:

- Do the similar words make sense to you?
- Which relationships seem purely topical (e.g., “oil” → “price”)
- and which seem semantic (e.g., “money” → “currency”)?

Q5). Analogy Tasks

```
def safe_analogy(m, positive, negative, topn=10):
    words = positive + negative
    miss = [w for w in words if w not in m.wv]
    return f"Missing tokens: {miss}" if miss else m.wv.most_similar(positive=positive, negative=negative, topn=topn)

# Country ↔ currency patterns common in Reuters (1980s era labels)
print(safe_analogy(model, positive=["japan", "dollar"],   negative=["usa"]))     # ≈ 'yen'
print(safe_analogy(model, positive=["britain", "dollar"], negative=["usa"]))     # ≈ 'sterling'
print(safe_analogy(model, positive=["france", "dollar"],  negative=["usa"]))     # ≈ 'franc'
```

Reflection:

- Do analogy results look meaningful?
- What kind of relationships does the model seem to capture?

Q6). Visualise Embeddings#

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

words = ["bank","money","currency","oil","trade","market","yen","sterling","franc","mark"]
vectors = [model.wv[w] for w in words if w in model.wv]
coords = PCA(n_components=2).fit_transform(vectors)

plt.figure(figsize=(8,6))
kept_words = [w for w in words if w in model.wv]
for i, word in enumerate(kept_words):
    plt.scatter(coords[i,0], coords[i,1])
    plt.annotate(word, (coords[i,0]+0.02, coords[i,1]+0.02))
plt.title("Word Embedding Visualisation")
plt.show()
```

Reflection:

- Which words appear close in your 2D map?
- Does their proximity match your expectation?

🟦 Part B – Extended Tasks

Q7). Compare CBOW and Skip-gram

Train a second model using CBOW (sg=0) and compare results.

```
model_cbow = Word2Vec(sentences, vector_size=100, window=5, min_count=1, sg=0, epochs=10)
print("Skip-gram → ", safe_most_similar(model, "bank"))
print("CBOW      → ", safe_most_similar(model_cbow, "bank"))
```

Question:

- How do the top results differ?

- Which model appears to capture semantic relations better?

Q8). Use Pretrained GloVe Embeddings

```
import gensim.downloader as api
glove = api.load("glove-wiki-gigaword-50")
print(glove.most_similar("car"))
print(glove.similarity("man","woman"))
```

Reflection:

- Compare your trained model to the pretrained GloVe results.
- Are pretrained embeddings more accurate? Why might that be?

Q9). Detect Bias in Embeddings

Try gender analogy tests:

```
def safe_bias_test(m, positive, negative, topn=5):
    missing = [w for w in positive + negative if w not in m.wv]
    if missing:
        return f"Skipped: Missing {missing}. Tip: use a pretrained model for general bias tests."
    return m.wv.most_similar(positive=positive, negative=negative, topn=topn)

print(safe_bias_test(model, ["man","doctor"], ["woman"]))
print(safe_bias_test(model, ["woman","nurse"], ["man"]))
```

Q10). If keywords are missing, run with pretrained GloVe for a true gender-bias demo

```
import gensim.downloader as api
glove = api.load("glove-wiki-gigaword-50")
print(glove.most_similar(positive=["man","doctor"], negative=["woman"]))
print(glove.most_similar(positive=["woman","nurse"], negative=["man"]))
```

Reflection:

- Do you observe any gender or occupational bias?
- What causes such bias in real-world models?

Final Reflection Questions

1) What makes Word2Vec different from frequency-based representations like TF-IDF?

2) How does Skip-gram actually learn meaning from prediction?
3) Why is cosine similarity used instead of Euclidean distance?
4) How could embedding bias affect real-world applications?
5) What are the limitations of Word2Vec for languages with low resources?
6) What insight did your 2D visualisation provide?