Lab 5: Sparse Vectors and Embedding Techniques

Introduction

This lab explores vector space models for document representation, focusing on tf-idf computation, normalization, cosine similarity, and Pointwise Mutual Information (PMI). It includes both manual calculations and implementation using Python and Streamlit.

Question 1: TF-IDF and Normalization

Manual TF-IDF Computation

Given a term-document frequency table and IDF values, TF-IDF scores were calculated for each document. These scores reflect the relative importance of each term in a document with respect to the entire corpus.

Sample TF-IDF Table:

Term	D1	D2	D3
car	0.36	0.48	0.36
insurance	0.48	0.36	0.36
auto	0.36	0.00	0.36
best	0.00	0.48	0.00

Query Scoring

Two example queries were evaluated using the TF-IDF scores:
- Query: 'car insurance' → Highest relevance: Document 1
- Query: 'best car' → Highest relevance: Document 2

Euclidean Normalization (Optional)

Each TF-IDF vector was normalized using L2 norm (Euclidean norm) to remove document length bias. The normalized vectors were then used for query scoring.

Question 2: Cosine Similarity and Word Neighbors

TF-IDF vectors were computed for each term in a small sample corpus. Cosine similarity was then used to find the nearest neighbors for a selected term. PCA visualization was applied to plot word vectors.

Example result: Nearest neighbors of 'apple' are 'fruit', 'banana', and 'mango'.

Word analogy was also attempted using TF-IDF vectors (e.g., king - man + woman = queen).

Question 3: Pointwise Mutual Information (PMI)

PMI scores were calculated to quantify the association between terms and documents. PMI is defined as: PMI(x, y) = log2(P(x, y) / (P(x) * P(y))).

Positive PMI (PPMI) values were computed to avoid negative values.

Example: PPMI scores for the query 'machine learning' highlight documents that contain statistically significant co-occurrences.

Streamlit Implementation Summary

The Streamlit app includes:

- Interactive TF-IDF computation and scoring

- Cosine similarity and word analogy visualization using PCA

- PPMI computation with query scoring

Code Snippets (Python)

TF-IDF Computation:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
print(X.toarray())
```

Cosine Similarity:

```
from sklearn.metrics.pairwise import cosine_similarity
cos_sim = cosine_similarity(X)
print(cos_sim)
```

PPMI Computation:

```
import numpy as np
```

```
def compute_ppmi(co_matrix):
    total_count = np.sum(co_matrix)
    word_sum = np.sum(co_matrix, axis=1)
    context_sum = np.sum(co_matrix, axis=0)
    ppmi = np.maximum(0, np.log2((co_matrix * total_count) / (word_sum[:, None] *
context_sum)))
    return ppmi
```