Lab 10.4: Visualizing Word Embeddings using t-SNE

2403A52006

MD.Mustafa

Import Required Libraries

```
!pip install gensim
import gensim.downloader as api
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
```

```
Collecting gensim
  Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (8.4 kB)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gensim) (2.0.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.16.3)
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim) (7.5.0
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=1.8.1->gensim)
Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (27.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 47.5 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

Load Pre-trained Word2Vec Model

```
print("Loading Word2Vec model...")
model = api.load("word2vec-google-news-300")

print("\nModel Loaded Successfully!")
print("Vocabulary Size:", len(model.key_to_index))

# Example vector
print("\nExample Vector for 'king' (first 10 dimensions):")
print(model["king"][:10])
```

```
Loading Word2Vec model...
[==================================================] 100.0% 1662.8/1662.8MB downloaded

Model Loaded Successfully!
Vocabulary Size: 3000000

Example Vector for 'king' (first 10 dimensions):
[ 0.12597656  0.02978516  0.00860596  0.13964844 -0.02563477 -0.03613281
  0.11181641 -0.19824219  0.05126953  0.36328125]
```

Select 40–50 Meaningful Words

```
words = [
    # Royalty
    "king", "queen", "prince", "princess", "monarch",

    # Countries / Cities
    "india", "china", "france", "paris", "london", "delhi",

    # Animals
    "dog", "cat", "lion", "tiger", "elephant",

    # Technology
    "computer", "software", "internet", "technology", "data",

    # Professions
    "doctor", "nurse", "teacher", "engineer", "student",

    # Fruits
    "apple", "banana", "mango", "orange", "grape",
    # Sports
    "football", "cricket", "tennis", "basketball", "hockey",

    # Vehicles
    "car", "bus", "train", "airplane", "truck",

    # Education
    "school", "college", "university", "classroom", "exam"
]

# Extract vectors
vectors = np.array([model[word] for word in words])

print("Total words selected:", len(words))
```

```
Total words selected: 46
```

Apply t-SNE (Dimensionality Reduction)

```
tsne = TSNE(n_components=2, random_state=42, perplexity=10)
reduced_vectors = tsne.fit_transform(vectors)

print("t-SNE Reduction Complete!")
print("Reduced Shape:", reduced_vectors.shape)
```

```
t-SNE Reduction Complete!
Reduced Shape: (46, 2)
```

Plot the Visualization

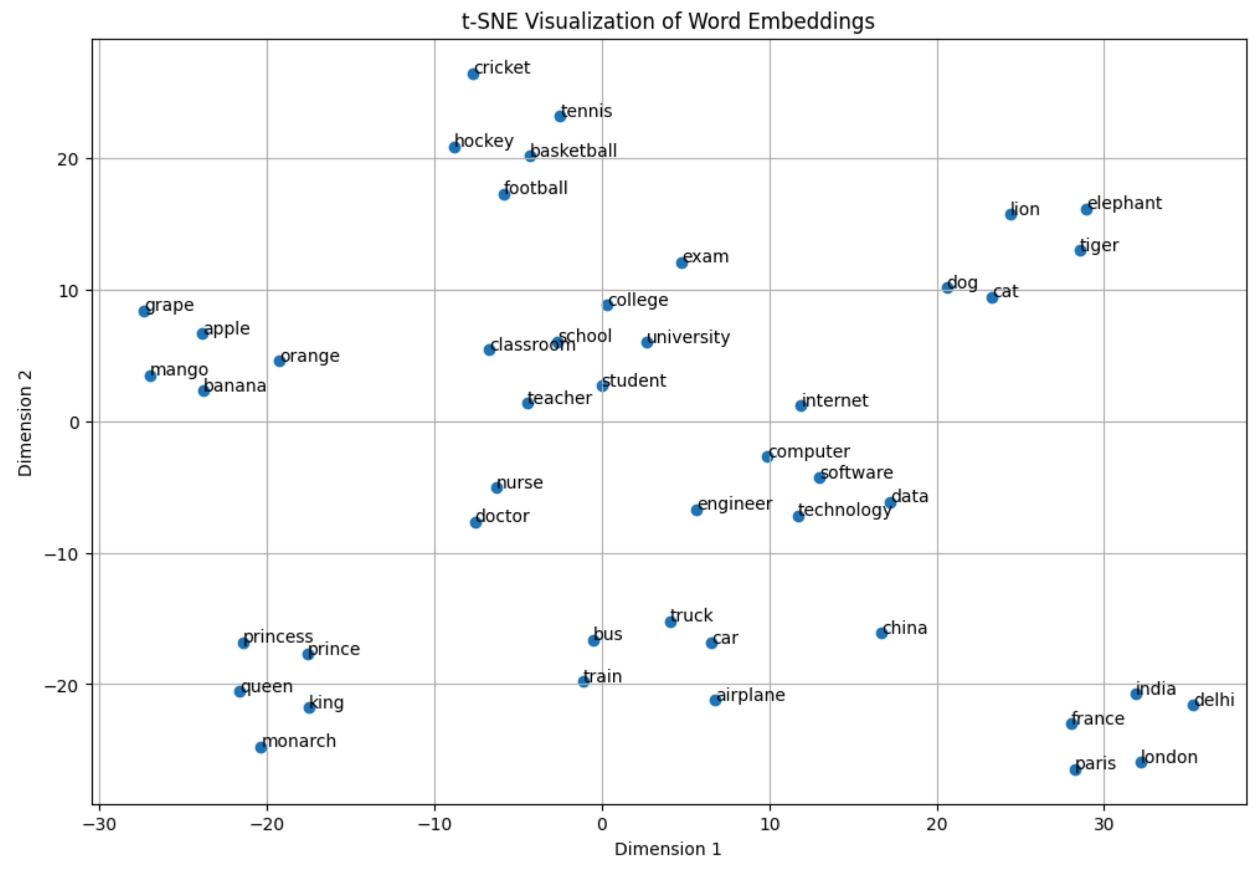
```
plt.figure(figsize=(12, 8))

x = reduced_vectors[:, 0]
y = reduced_vectors[:, 1]

plt.scatter(x, y)

for i, word in enumerate(words):
    plt.annotate(word, (x[i], y[i]))

plt.title("t-SNE Visualization of Word Embeddings")
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")
plt.grid(True)
plt.show()
```



Interpretation

The t-SNE visualization shows clear semantic clusters. Royalty words such as king and queen appear close together. Animal-related words form a separate group. Technology-related words cluster in another region. Countries and cities tend to group based on geographic relations. Sports terms appear close due to contextual similarity. Some words may appear slightly misplaced due to limitations of dimensionality reduction and dataset bias. Overall, embeddings capture