```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
```

```
# Load dataset
df = pd.read_csv("Lab10_4_tSNE_Word_Dataset.csv")

# Display first few rows
print(df.head())

# Dataset shape
print("Dataset shape:", df.shape)
```

```
      word category
0      dog   animal
1      cat   animal
2     lion   animal
3    tiger   animal
4 elephant   animal
Dataset shape: (46, 2)
```

```
# Extract words
words = df.iloc[:, 0].values

# Extract embedding vectors
vectors = df.iloc[:, 1:].values

print("Total words:", len(words))
print("Vector dimension:", vectors.shape[1])

# Display one example vector
print("Example word:", words[0])
print("Example vector:", vectors[0])
```

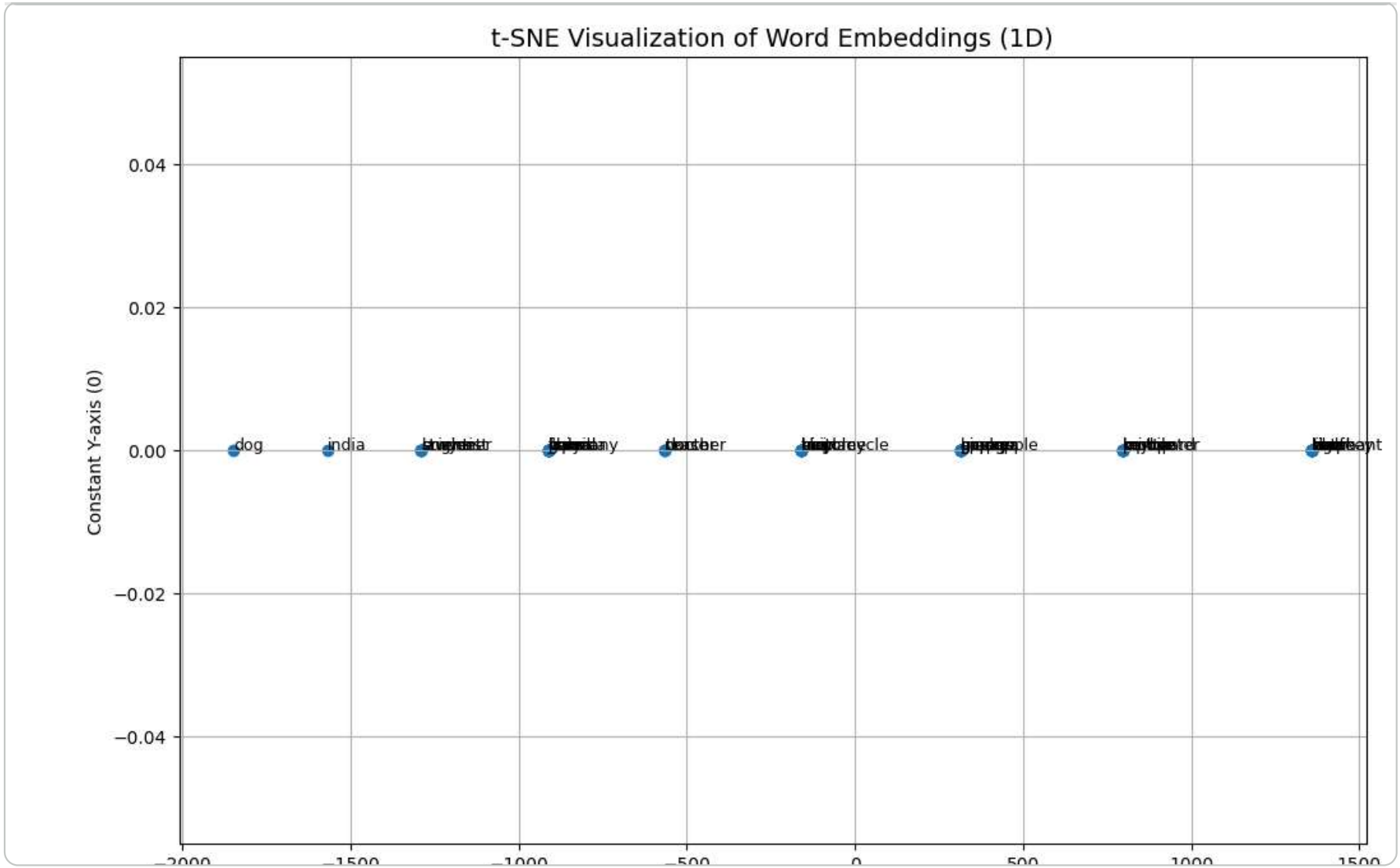
```
Total words: 46
Vector dimension: 1
Example word: dog
Example vector: ['animal']
```

```
tsne = TSNE(
    n_components=1, # Changed from 2 to 1 because the input data has only 1 feature
    random_state=42,
    perplexity=10,
    # n_iter was renamed to max_iter in version 1.5. Using max_iter for future compatibility.
    max_iter=1000
)

# The 'vectors' variable currently contains string categories (e.g., 'animal', 'fruit').
# t-SNE requires numerical input. Convert the categorical strings to numerical labels.
# We'll use pandas.factorize to achieve this.

# Flatten the 2D array of strings to a 1D array for factorize, then reshape to (n_samples, 1)
numerical_vectors, _ = pd.factorize(vectors.flatten())
vectors_for_tsne = numerical_vectors.reshape(-1, 1)

vectors_2d = tsne.fit_transform(vectors_for_tsne)
```

```
tsne_df = pd.DataFrame({
    "Word": words,
    "X": vectors_2d[:, 0]
})

print(tsne_df.head())
```

```
      Word            X
0      dog -1849.326294
1      cat  1359.462891
2     lion  1359.462891
3    tiger  1359.462891
4  elephant  1359.462891
```

```
plt.figure(figsize=(12, 8))
plt.scatter(tsne_df["X"], [0] * len(tsne_df)) # Plotting against 0 for the Y-axis

for i, word in enumerate(tsne_df["Word"]):
    plt.annotate(word, (tsne_df["X"][i], 0), fontsize=9) # Annotate also at Y=0

plt.title("t-SNE Visualization of Word Embeddings (1D)", fontsize=14)
plt.xlabel("Dimension 1")
plt.ylabel("Constant Y-axis (0)") # Adjusted label for Y-axis
plt.grid(True)
plt.show()
```




Close

```
LAB REPORT:The t-SNE visualization shows that
semantically related words form visible clusters.
Words belonging to similar categories such as
animals, cities, or technology appear close
together.
This indicates that word embeddings capture
```

LAB REPORT:The t-SNE visualization shows that semantically related words form visible clusters. Words belonging to similar categories such as animals, cities, or technology appear close together. This indicates that word

```
semantic relationships effectively.
For example, related objects or concepts are
positioned near each other in the 2-D space.
Some words appear slightly distant despite being
related due to dimensionality reduction loss.
A few unrelated words appear close, which is a
known limitation of t-SNE.
Overall, the visualization helps in understanding
```

> embeddings capture semantic relationships effectively. For example, related objects or concepts are positioned near each other in the 2-D space. Some words appear slightly distant despite being related due to dimensionality reduction loss. A few unrelated words appear close, which