```
Start coding or generate with AI.
```

## Import **Libraries**

```
import sys
!pip install gensim

print("gensim installed successfully.")
```

```
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) (2.1.1)
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 49.5 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
gensim installed successfully.
```

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

print("Libraries imported successfully.")
```

```
Libraries imported successfully.
```

## **Pre-trained Word Embedding Model**

```
import gensim.downloader as api

print("Downloading and loading 'glove-wiki-gigaword-100' model... This may take a few minutes.")
word_vectors = api.load('glove-wiki-gigaword-100')
print("Model 'glove-wiki-gigaword-100' loaded successfully.")
```

```
Downloading and loading 'glove-wiki-gigaword-100' model... This may take a few minutes.
Model 'glove-wiki-gigaword-100' loaded successfully.
```

## WORD **LIST**

```
nature_words = [
    'river', 'mountain', 'forest', 'tree', 'flower', 'ocean', 'wind', 'sun', 'sky', 'earth',
    'animal', 'plant', 'rock', 'cloud', 'rain', 'snow', 'storm', 'desert', 'valley', 'field'
]

technology_words = [
    'computer', 'software', 'hardware', 'internet', 'network', 'algorithm', 'data', 'robot', 'code', 'artificial',
    'intelligence', 'machine', 'learning', 'cyber', 'virtual', 'reality', 'digital', 'device', 'app', 'system'
]

emotion_words = [
    'joy', 'sadness', 'anger', 'fear', 'love', 'happiness', 'grief', 'calm', 'stress', 'excitement',
    'anxiety', 'hope', 'disgust', 'surprise', 'trust', 'loneliness', 'peace', 'frustration', 'contentment', 'envy'
]

word_vectors_list = []
word_labels = []
oov_words = []

def process_words(word_list, label):
    for word in word_list:
        try:
            vector = word_vectors[word]
            word_vectors_list.append(vector)
            word_labels.append(label)
        except KeyError:
            oov_words.append(word)

process_words(nature_words, 'Nature')
process_words(technology_words, 'Technology')
process_words(emotion_words, 'Emotion')

word_vectors_array = np.array(word_vectors_list)

print(f"Total words processed: {len(word_vectors_list) + len(oov_words)}")
print(f"Words with embeddings: {len(word_vectors_list)}")
print(f"Out-of-vocabulary words: {len(oov_words)}")
print(f"Example word vectors shape: {word_vectors_array.shape}")
print(f"Example word labels count: {len(word_labels)}")

if oov_words:
    print(f"OOV words: {', '.join(oov_words)}")
else:
    print("No out-of-vocabulary words found.")
```

```
Total words processed: 60
Words with embeddings: 60
Out-of-vocabulary words: 0
Example word vectors shape: (60, 100)
Example word labels count: 60
No out-of-vocabulary words found.
```

## **t-SNE** REDUCTION

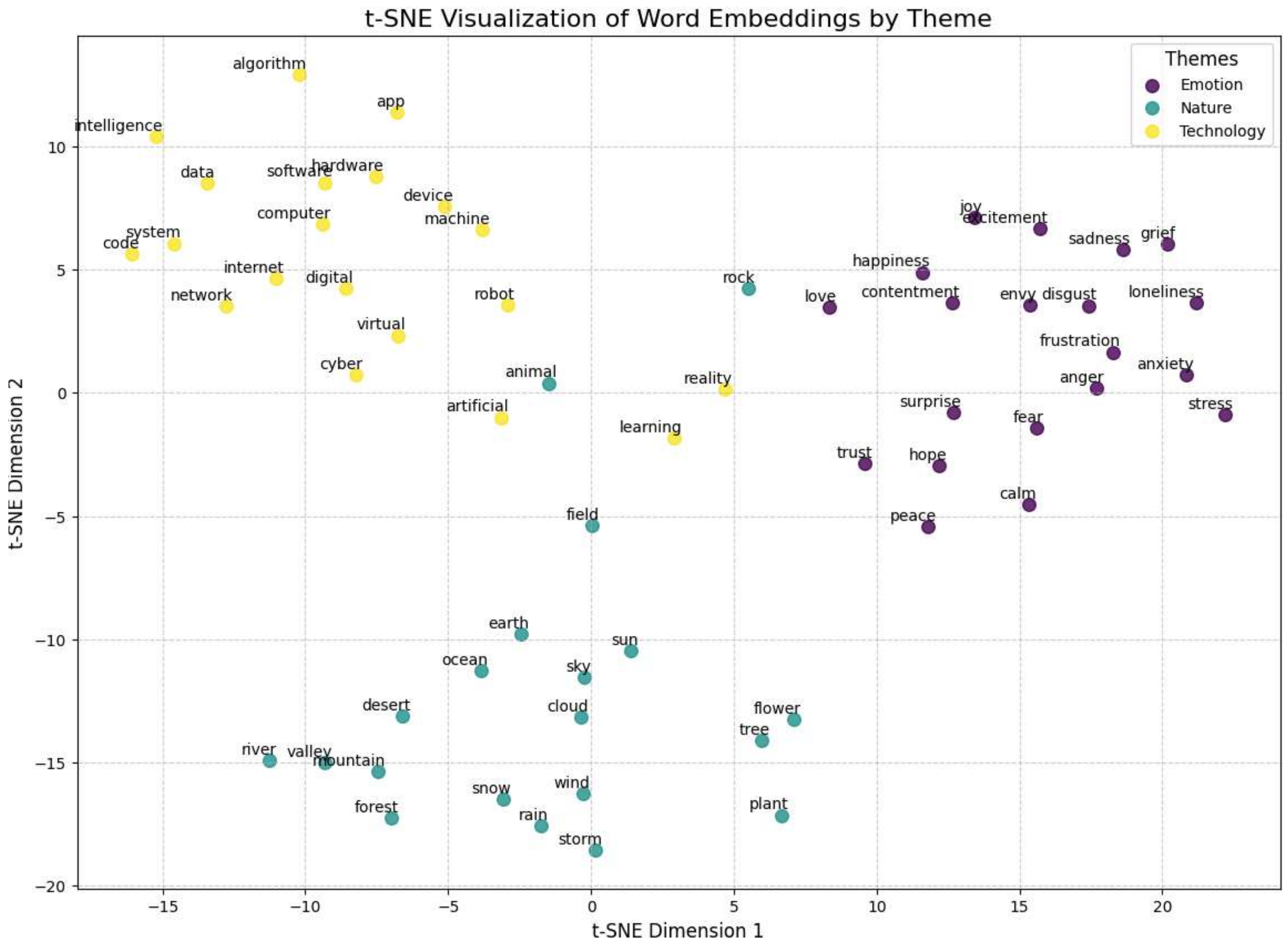
```
tsne = TSNE(n_components=2, random_state=42, perplexity=10)
tsne_embeddings = tsne.fit_transform(word_vectors_array)

print(f"Shape of original word vectors array: {word_vectors_array.shape}")
print(f"Shape of t-SNE embeddings: {tsne_embeddings.shape}")
```

```
Shape of original word vectors array: (60, 100)
Shape of t-SNE embeddings: (60, 2)
```

## **VISUALIZE WORD EMBEDDINGS**

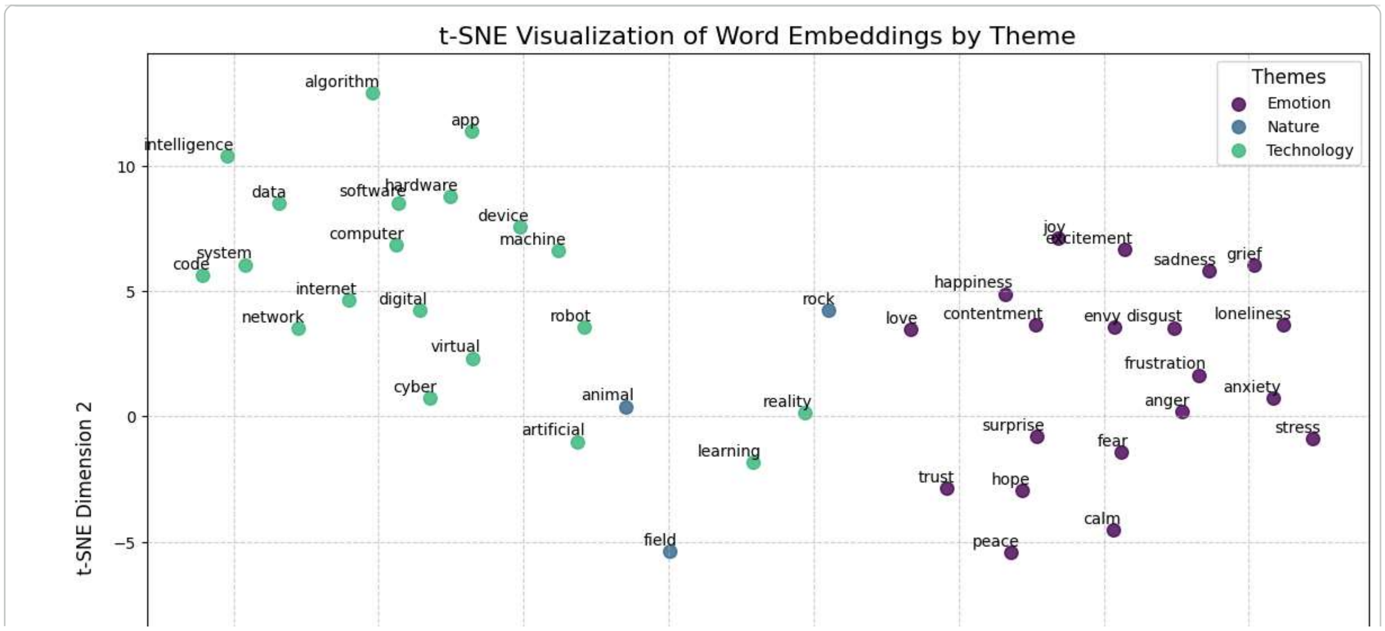
```
fig, ax = plt.subplots(figsize=(14, 10))

unique_themes = list(np.unique(word_labels))
colors = plt.cm.get_cmap('viridis', len(unique_themes))

theme_to_color = {theme: colors(i) for i, theme in enumerate(unique_themes)}

all_words_processed = nature_words + technology_words + emotion_words

for i, theme in enumerate(unique_themes):
    # Filter embeddings and words for the current theme
    theme_indices = [j for j, label in enumerate(word_labels) if label == theme]
    theme_embeddings = tsne_embeddings[theme_indices]
    current_theme_words = [all_words_processed[j] for j in theme_indices]

    ax.scatter(
        theme_embeddings[:, 0],
        theme_embeddings[:, 1],
        color=theme_to_color[theme],
        label=theme,
        s=70, # Increased scatter point size for visibility
        alpha=0.8
    )

    # Annotate each point with its corresponding word
    for k, word in enumerate(current_theme_words):
        ax.annotate(
            word,
            (theme_embeddings[k, 0], theme_embeddings[k, 1]),
            xytext=(5, 2),
            textcoords='offset points',
            ha='right',
            va='bottom'
        )

ax.set_title('t-SNE Visualization of Word Embeddings by Theme', fontsize=16)
ax.set_xlabel('t-SNE Dimension 1', fontsize=12)
ax.set_ylabel('t-SNE Dimension 2', fontsize=12)
ax.legend(title='Themes', fontsize=10, title_fontsize=12)
ax.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

```
tmp/ipython-input-3072656756.py:4: MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib 3.7 and will
  colors = plt.cm.get_cmap('viridis', len(unique_themes))
```



## PLOT VISUALIZATION

```
fig, ax = plt.subplots(figsize=(14, 10))

unique_themes = list(np.unique(word_labels))
cmap = plt.colormaps.get_cmap('viridis') # Get the colormap object

theme_to_color = {theme: cmap(i / len(unique_themes)) for i, theme in enumerate(unique_themes)} # Map themes to distinct colors

# Note: all_words_processed is not strictly needed here if embedded_words list is used.
# However, given oov_words is empty, all_words_processed correctly maps to word_labels indices in this specific execution.
# For robustness, a list of successfully embedded words would be preferred.

for i, theme in enumerate(unique_themes):
        # Filter embeddings and words for the current theme
        theme_indices = [j for j, label in enumerate(word_labels) if label == theme]
        theme_embeddings = tsne_embeddings[theme_indices]

        # Assuming all words were embedded based on previous `oov_words` being empty.
        # If oov_words were present, 'embedded_words' list should be maintained alongside word_vectors_list and word_labels.
        # For this specific run, all_words_processed has the correct order and length.
        current_theme_words = [word for idx, word in enumerate(nature_words + technology_words + emotion_words) if idx in theme_indi

        ax.scatter(
            theme_embeddings[:, 0],
            theme_embeddings[:, 1],
            color=theme_to_color[theme],
            label=theme,
            s=70, # Increased scatter point size for visibility
            alpha=0.8
        )

        # Annotate each point with its corresponding word
        for k, word in enumerate(current_theme_words):
            ax.annotate(
                word,
                (theme_embeddings[k, 0], theme_embeddings[k, 1]),
                xytext=(5, 2),
                textcoords='offset points',
                ha='right',
                va='bottom'
            )

    ax.set_title('t-SNE Visualization of Word Embeddings by Theme', fontsize=16)
    ax.set_xlabel('t-SNE Dimension 1', fontsize=12)
    ax.set_ylabel('t-SNE Dimension 2', fontsize=12)
    ax.legend(title='Themes', fontsize=10, title_fontsize=12)
    ax.grid(True, linestyle='--', alpha=0.6)
    plt.show()
```


t-SNE Visualization of Word Embeddings by Theme
Themes
Emotion
Nature
Technology
t-SNE Dimension 2
10
5
0
−5
algorithm
app
intelligence
data
software
hardware
device
computer
machine
system
code
internet
digital
network
robot
virtual
cyber
animal
artificial
reality
learning
field
rock
joy
excitement
sadness
grief
happiness
love
contentment
envy
disgust
loneliness
frustration
anxiety
anger
surprise
fear
stress
trust
hope
calm
peace