Assignment: Word2Vec Model Visualization using t-SNE Objective: In this assignment, you will learn how to visualize word embeddings created by a Word2Vec model (CBOW or Skip-gram) using t-SNE (t-Distributed Stochastic Neighbor Embedding). You will also work with assertions to ensure the correctness of the data flow.

Tasks: Preprocessing and Model Training: Write a Python function that accepts a list of sentences, preprocesses the text, and trains a Word2Vec model (either CBOW or Skip-gram). Use a small sample of sentences (similar to those given below) to train the Word2Vec model. Word Embedding Visualization: Select a trained model (either CBOW or Skip-gram). Extract the word vectors from the model and store them in a variable. Use t-SNE (from sklearn.manifold) to reduce the dimensionality of the word vectors to 2D. Visualization: Plot the 2D word vectors using matplotlib and display the word labels. Ensure that each word is positioned based on its similarity to other words in the embedding space. simple_preprocess:=It produces clean and consistent tokenized text that can be directly fed into models or used for further processing

In [1]:
```
#gensim.utils.simple_preprocess is a utility function from the gensim library
from gensim.utils import simple_preprocess
```

In [2]:
```
# Input sentences
sentences = [
    "Natural language processing and machine learning are exciting fields.",
    "Word2Vec is a popular method for creating word embeddings.",
    "CBOW and Skip-gram are two types of Word2Vec models."
]

# Preprocessing
#Applies Gensim's simple_preprocess function to each sentence in the sentences l
preprocessed_sentences = [simple_preprocess(sentence) for sentence in sentences]

# Print the actual output to debug
print("Actual Preprocessed Sentences:", preprocessed_sentences)

# Updated expected sentences
expected_sentences = [
    ['natural', 'language', 'processing', 'and', 'machine', 'learning', 'are', '
    ['word', 'vec', 'is', 'popular', 'method', 'for', 'creating', 'word', 'embed
    ['cbow', 'and', 'skip', 'gram', 'are', 'two', 'types', 'of', 'word', 'vec', 
]
```

```
Actual Preprocessed Sentences: [['natural', 'language', 'processing', 'and', 'mac
hine', 'learning', 'are', 'exciting', 'fields'], ['word', 'vec', 'is', 'popular',
'method', 'for', 'creating', 'word', 'embeddings'], ['cbow', 'and', 'skip', 'gra
m', 'are', 'two', 'types', 'of', 'word', 'vec', 'models']]
```

In [4]:
```
from gensim.models import Word2Vec
# Train CBOW and Skip-gram models
#min_count=1 means all words that appear at least once in the corpus will be inc
#The dimensionality of the word embeddings each word will be represented as a 50
#sg=0 means the Continuous Bag of Words (CBOW) model is used.
# BEGIN SOLUTION
cbow_model = Word2Vec(preprocessed_sentences, vector_size=50,
                      window=3, min_count=1, sg=0)
skipgram_model = Word2Vec(preprocessed_sentences,
```

```
                vector_size=50, window=3, min_count=1, sg=1)
# END SOLUTION
```

In [6]:

```
# CBOW vector for 'language'
#wv is an attribute of trained word2vec
#cbow_model.wv['language']:This retrieves the word vector
#for the word 'language'
#from the vocabulary of the cbow_model.
# BEGIN SOLUTION
cbow_vector = cbow_model.wv['language']
print("CBOW Vector for 'language':", cbow_vector)
# Skip-gram vector for 'language'
skipgram_vector = skipgram_model.wv['language']
print("Skip-gram Vector for 'language':", skipgram_vector)
# Similarity comparisons
cbow_similarity = cbow_model.wv.similarity('language',
                                           'processing')
skipgram_similarity = skipgram_model.wv.similarity('language',
                                                   'machine')
print(cbow_similarity)
print(skipgram_similarity)
# END SOLUTION
```

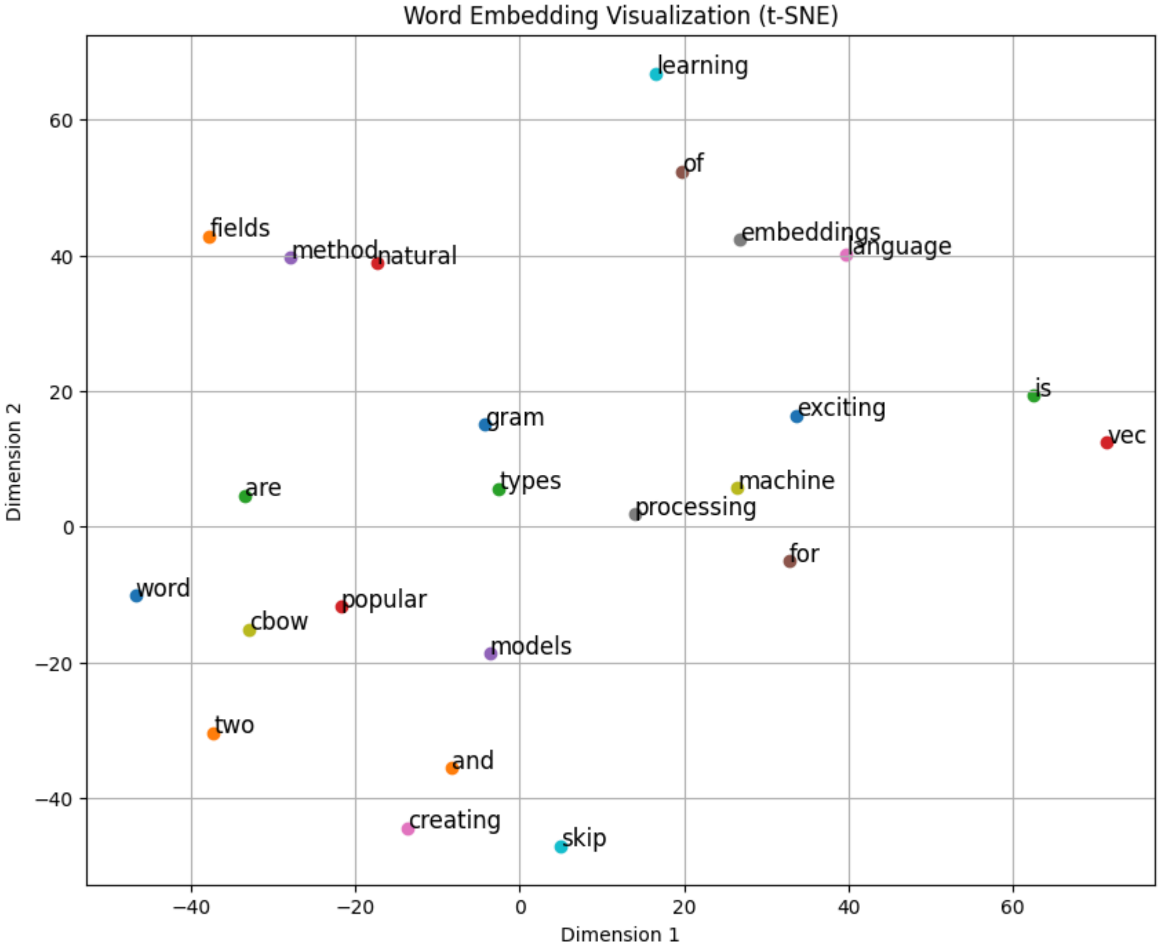
```
CBOW Vector for 'language': [-0.01648339  0.01859842 -0.00039611 -0.00393434  0.0
0920413 -0.00819227
  0.00548813  0.013882    0.01212855 -0.01502589  0.01876515  0.00934304
  0.00793339 -0.01248859  0.0169206  -0.00430269  0.01765161 -0.01072529
 -0.0162621   0.01364475  0.00334022 -0.004396    0.01902776  0.01898638
 -0.01954934  0.00500796  0.01231197  0.00774455  0.0040454   0.00086309
  0.00134816 -0.00764242 -0.01428238 -0.00417767  0.00784265  0.01764069
  0.0185197  -0.01195131 -0.01880593  0.01952545  0.00686199  0.01033132
  0.01256619 -0.00561146  0.01464742  0.00566262  0.0057407  -0.00476109
 -0.00625691 -0.00473869]
Skip-gram Vector for 'language': [-0.01648363  0.01860168 -0.00039618 -0.00393355
0.00920536 -0.00819438
  0.00548537  0.01388096  0.01213131 -0.01502552  0.01876256  0.00934442
  0.00793736 -0.01249054  0.01691916 -0.00430375  0.01764755 -0.0107291
 -0.01626193  0.01364436  0.00334088 -0.00439349  0.01902569  0.01898476
 -0.01955001  0.00500788  0.01231466  0.00774178  0.00404745  0.00086536
  0.00134797 -0.00764366 -0.01428161 -0.00417519  0.00784366  0.01763926
  0.01851777 -0.01194969 -0.01880298  0.01952852  0.00685894  0.01033415
  0.01256573 -0.00560637  0.01464841  0.00565847  0.00574082 -0.00475956
 -0.00625414 -0.00473926]
0.112580754
-0.08931184
```

In [8]:

```
#t-SNE (t-distributed Stochastic Neighbor Embedding) is a popular dimensionality
#reduction technique used to visualize high-dimensional data, such as word embed
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Select a model for visualization (e.g., CBOW or Skip-gram)
model = cbow_model  # Change to skipgram_model if needed

# Get words and vectors
words = list(model.wv.key_to_index.keys())
vectors = model.wv[words]

# Reduce dimensions using t-SNE
tsne = TSNE(n_components=2, random_state=42, perplexity=5, n_iter=500)
```

```
reduced_vectors = tsne.fit_transform(vectors)

# Plot the words
plt.figure(figsize=(10, 8))
for i, word in enumerate(words):
    plt.scatter(reduced_vectors[i, 0], reduced_vectors[i, 1], label=word)
    plt.text(reduced_vectors[i, 0] + 0.01, reduced_vectors[i, 1] + 0.01,
             word, fontsize=12)

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

```
/opt/conda/lib/python3.10/site-packages/sklearn/manifold/_t_sne.py:1162: FutureWa
rning: 'n_iter' was renamed to 'max_iter' in version 1.5 and will be removed in
1.7.
  warnings.warn(
```



In [ ]: