```
# Mount Google Drive to access dataset
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Import necessary libraries
import pandas as pd
import numpy as np
import nltk
```

```
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

from gensim.models import word2vec
from gensim.models import Word2Vec

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

```
def pprocess(text):
    # Load the NLTK stop words
    stop_words = set(stopwords.words('english'))
    # Initialize the NLTK stemmer
    stemmer = PorterStemmer()
    tokens = word_tokenize(text)

    # Lowercase the tokens
    tokens = [token.lower() for token in tokens]

    # Remove punctuation and special characters
    tokens = [token for token in tokens if token.isalpha()]

    # Remove stop words
    tokens = [token for token in tokens if token not in stop_words]

    # Stem the tokens
    tokens = [stemmer.stem(token) for token in tokens]

    return tokens
```

```
def preprocess(text):
    ret = []
    # Load the NLTK stop words
    stop_words = set(stopwords.words('english'))
    # Initialize the NLTK stemmer
    stemmer = PorterStemmer()

    for tex in text:
        # Tokenize the sentence
        tokens = word_tokenize(tex)

        # Lowercase the tokens
        tokens = [token.lower() for token in tokens]

        # Remove punctuation and special characters
        tokens = [token for token in tokens if token.isalpha()]

        # Remove stop words
        tokens = [token for token in tokens if token not in stop_words]

        # Stem the tokens
        tokens = [stemmer.stem(token) for token in tokens]
        ret.append(tokens)

    return ret
```

```
# Load the dataset into a DataFrame
df_0 = pd.read_csv('/content/drive/MyDrive/bick/train00.csv')
df_1 = pd.read_csv('/content/drive/MyDrive/bick/train01.csv')
df_2 = pd.read_csv('/content/drive/MyDrive/bick/train02.csv')
df_3 = pd.read_csv('/content/drive/MyDrive/bick/train03.csv')
```

```
train_df0 = df_0['Review Text'].to_frame()
train_df0.columns = ['text']

train_df1 = df_1['Review'].to_frame()
train_df1.columns = ['text']

train_df2 = df_2['Review'].to_frame()
train_df2.columns = ['text']

train_df3 = df_3['review_text'].to_frame()
train_df3.columns = ['text']

train_df = pd.concat([train_df0, train_df1, train_df2, train_df3], ignore_index=True)
#train_df['corpus'] = ''
```

```
train_df
```

	text
0	Absolutely wonderful - silky and sexy and comf...
1	Love this dress! it's sooo pretty. i happene...
2	I had such high hopes for this dress and reall...
3	I love, love, love this jumpsuit. it's fun, fl...
4	This shirt is very flattering to all due to th...
...	...
70453	I oot this dress in the blue. it fits great--h...
70454	I was very patient with this dress. i was wait...
70455	The deep v doesn't gape, and flatters the neck...
70456	I saw this dress online this morning, went int...
70457	Super cute jacket .perfect for fall i can't st...

70458 rows × 1 columns

```
train_df['text'].isna().sum()
```

```
2535
```

```
train_df = train_df.dropna(axis=0)
train_df = train_df.drop_duplicates(['text'], keep='first')
train_df = train_df.reset_index(drop=True)
```

```
train_df['corpus'] = train_df['text'].apply(pprocess)
```

```
train_df
```

```
corpus = train_df['corpus'].to_list()
```

```
0    Absolutely wonderful - silky and sexy and comf...    [absolut, wonder, silki, sexi, comfort]
```

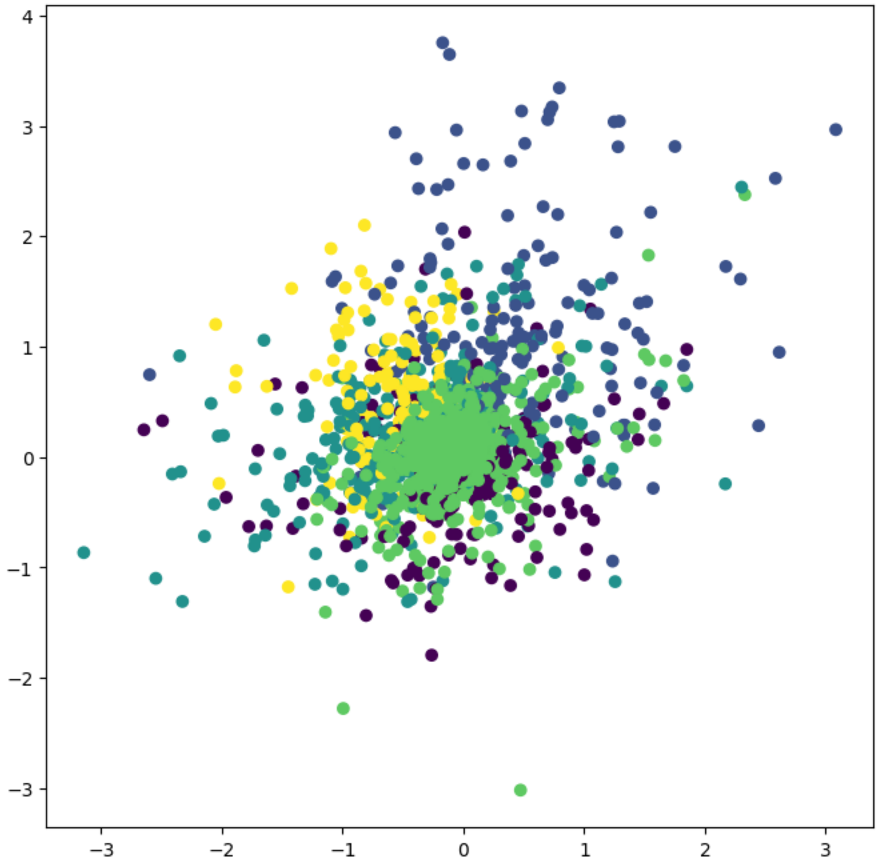
```
# Train a Word2Vec model on the preprocessed corpus
model = Word2Vec(corpus, min_count=1, vector_size=100)
```

```
# Compute the similarity between the input sentence and each sentence in the text file
input_sentence = "As you put on the shirt, you suddenly realize that it feels like someone else's skin clinging uncomfortab
input_tokens = pprocess(input_sentence)
similar_sentences = []
for i in range(len(corpus)):
    if len(corpus[i]) == 0 or len(input_tokens) == 0:
        continue
    similarity = model.wv.n_similarity(input_tokens, corpus[i])
    similar_sentences.append((train_df.loc[i, 'text'], similarity))

# Sort the similar sentences by similarity score in descending order and output the top n sentences
n = 30
top_similar_sentences = sorted(similar_sentences, key=lambda x: x[1], reverse=True)[:n]
tmp = 1
for sentence, similarity in top_similar_sentences:
    print(f"\n{tmp}) sentence: \n\t", sentence, "\nsimilarity: ", similarity)
    tmp += 1
```

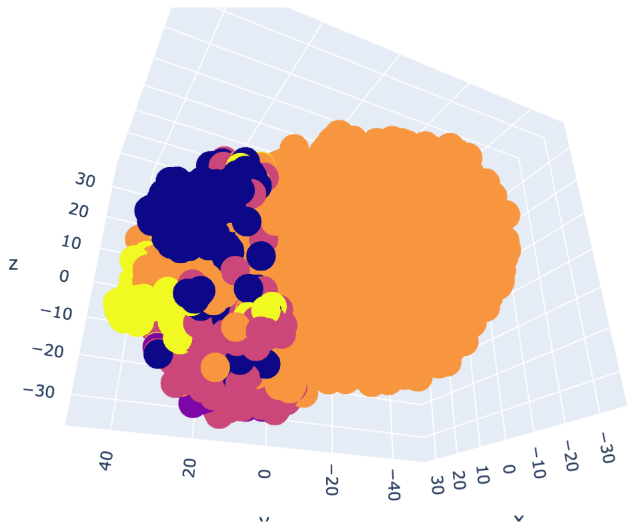
```

1) sentence: 
	 Lovely sweater. it fits baggy and form fitted somehow. i like the multi-fabric look. however, the main body 
similarity:  0.8000628

2) sentence: 
	 I got this 25% off before christmas. i immediately started wearing it, thinking it would be a staple. i like
similarity:  0.7970948

3) sentence: 
	 Nice fabric, great color, would be lovely ... on someone with a straight-and-narrow body type. for ladies wi
similarity:  0.7910863

4) sentence: 
	 Nice fabric, magnificent color, would be lovely ... on someone with a straight-and-narrow body type. for lad
similarity:  0.79052174

5) sentence: 
	 I was hoping this clothe was a bit thicker, but when i tried it on the fabric was very thin and sheer. becau
similarity:  0.7869623

6) sentence: 
	 I was hoping this dress was a bit thicker, but when i tried it on the fabric was very thin and sheer. becaus
similarity:  0.7840034

7) sentence: 
	 I thought this was love at first sight. the color and design are both stunning. and it drapes just perfectly
similarity:  0.7836629

8) sentence: 
	 I loved the idea of this clothe, but i find that it isn't very flattering. i think the arm holes are actuall

the material is soft, but a bit stuffy and i could tell it would give me crazy static in my hair. 
similarity:  0.7835511

9) sentence: 
	 I loved the idea of this dress, but i find that it isn't very flattering. i think the arm holes are actually

the material is soft, but a bit stuffy and i could tell it would give me crazy static in my hair. 
similarity:  0.78336215

10) sentence: 
	 Wow. bailey 44 knocks it out of the park with this sensual, i-am-woman-hear-me-purr body-slimming clothe. we
similarity:  0.77883756

11) sentence: 
	 This top is really beautiful, but looks completely horrible on my 5'9" apple shaped body. in the words of my
similarity:  0.77763677

12) sentence: 
	 The material was thin, showed every lump you'd rather hide and clung to the body. even the sleeves were tigh
similarity:  0.77623314

13) sentence: 
	 I thought this was love at first sight. the color and design are both stunning. and it drapes just excellent
similarity:  0.7758187
```

```
import gensim
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Load trained word2vec model
# model = gensim.models.Word2Vec.load('path/to/trained/model')

# Get embedding matrix and list of words
embeddings = model.wv.vectors
words = model.wv.index_to_key

# Cluster embeddings using K-Means
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(embeddings)

# Plot the clusters using the first two dimensions of the embeddings
plt.figure(figsize=(8, 8))
plt.scatter(embeddings[:, 0], embeddings[:, 1], c=clusters, cmap='viridis')
plt.show()

```

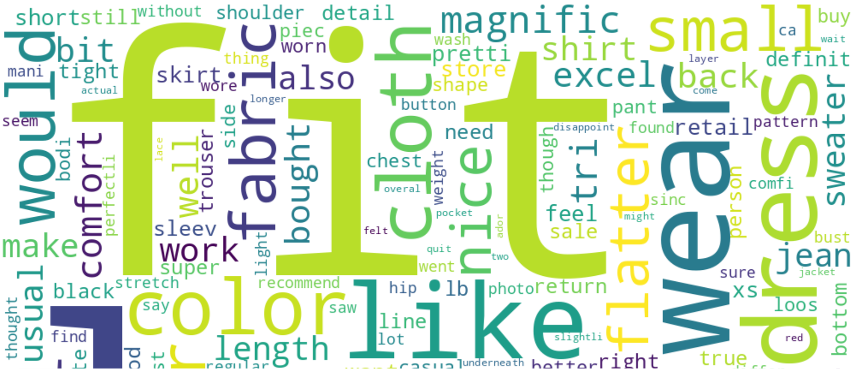
```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will
  warnings.warn(
```

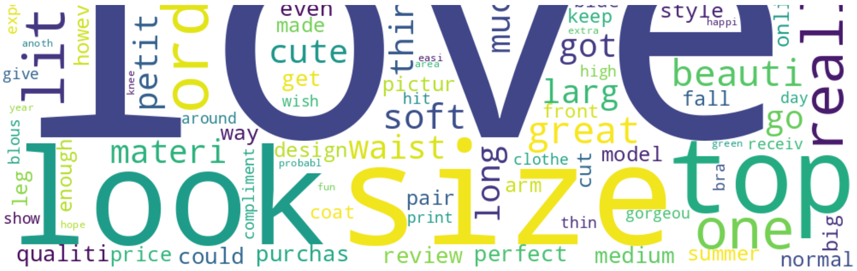


```
import gensim
import plotly.express as px
from sklearn.manifold import TSNE

# Load trained word2vec model
# model = gensim.models.Word2Vec.load('path/to/trained/model')

# Get embedding matrix and list of words
embeddings = model.wv.vectors
words = model.wv.index_to_key

# Use t-SNE to project embeddings into 2D space
tsne = TSNE(n_components=3, random_state=42)
embeddings_3d = tsne.fit_transform(embeddings)

# Create a dataframe with the 2D embeddings and their corresponding words
df = pd.DataFrame({'x': embeddings_3d[:, 0], 'y': embeddings_3d[:, 1], 'z': embeddings_3d[:, 2], 'word': words})

# Use Plotly to create a scatter plot of the embeddings
fig = px.scatter_3d(df, x='x', y='y', z='z', color=clusters, hover_data=['word'])
fig.show()
```



```
# Get the top 100 most common words
top_words = model.wv.index_to_key[:500]

# Assign frequency values based on how often each word appears in the model

word_frequencies = {}
for word in top_words:
    word_frequencies[word] = model.wv.get_vecattr(word, "count")

from wordcloud import WordCloud

# Create the word cloud object
wordcloud = WordCloud(width=1000, height=800, background_color="white")

# Generate the word cloud
wordcloud.generate_from_frequencies(word_frequencies)

# Display the word cloud
import matplotlib.pyplot as plt
plt.figure(figsize=(10,8), facecolor=None)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()

```





✓ 6초 오후 6:58에 완료됨