```
# Install necessary libraries
!pip install -q nltk spacy gensim wordcloud pyLDAvis bokeh
!python -m spacy download en_core_web_sm

# Import libraries
import nltk
import re
import numpy as np
import pandas as pd
import gensim
import spacy
import logging
import warnings
import gensim.corpora as corpora
import matplotlib.pyplot as plt
from pprint import pprint
from nltk.corpus import stopwords
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel
from wordcloud import WordCloud, STOPWORDS
import matplotlib.colors as mcolors
from sklearn.manifold import TSNE
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
import pyLDAvis
import pyLDAvis.gensim_models

# Settings
warnings.filterwarnings("ignore", category=DeprecationWarning)
nltk.download('stopwords')
stop_words = stopwords.words('english')

# Load dataset (adjust path if necessary)
from google.colab import files
uploaded = files.upload()

import io
df = pd.read_csv(io.BytesIO(uploaded[next(iter(uploaded))]))
print(df.head())

# Ensure the review column exists
assert 'review_text' in df.columns, "Expected a column named 'review_text'"

# Preprocessing
def sent_to_words(sentences):
    for sent in sentences:
        sent = re.sub('\s+', ' ', sent)  # Remove newline characters
        sent = re.sub("'", "", sent)     # Remove single quotes
        yield gensim.utils.simple_preprocess(str(sent), deacc=True)

# Convert to list
```

```
all_reviews = df['review_text'].astype(str).values.tolist()
reviews_words = list(sent_to_words(all_reviews))

# Build bigram and trigram models
bigram = gensim.models.Phrases(reviews_words, min_count=5, threshold=10)
trigram = gensim.models.Phrases(bigram[reviews_words], threshold=10)

bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# Load SpaCy once
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])

# Function for text processing
def process_words(texts, stop_words=stop_words, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    texts = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]
    texts = [bigram_mod[doc] for doc in texts]
    texts = [trigram_mod[bigram_mod[doc]] for doc in texts]

    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])

    texts_out = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in
texts_out]
    return texts_out

# Process reviews
data_final = process_words(reviews_words)

# Create Dictionary and Corpus
id2word = corpora.Dictionary(data_final)
corpus = [id2word.doc2bow(text) for text in data_final]

# Build LDA model
lda_model = gensim.models.LdaModel(
    corpus=corpus,
    id2word=id2word,
    num_topics=7,
    random_state=100,
    update_every=1,
    chunksize=10,
    passes=10,
    alpha='symmetric',
    iterations=100,
    per_word_topics=True
)

# Print topics
pprint(lda_model.print_topics())
```

```
# Generate Word Clouds for each topic
cols = [color for name, color in mcolors.TABLEAU_COLORS.items()]
topics = lda_model.show_topics(formatted=False)
cloud = WordCloud(
   stopwords=stop_words,
   background_color='white',
   width=2500,
   height=1800,
   max_words=10,
   colormap='tab10'
)

fig, axes = plt.subplots(3, 3, figsize=(12, 12), sharex=True, sharey=True)
for i, ax in enumerate(axes.flatten()):
   if i >= len(topics):
      ax.axis('off')
      continue
   fig.add_subplot(ax)
   topic_words = dict(topics[i][1])
   cloud.generate_from_frequencies(topic_words, max_font_size=300)
   ax.imshow(cloud, interpolation='bilinear')
   ax.set_title(f'Topic {i}', fontdict=dict(size=16))
   ax.axis('off')
plt.tight_layout()
plt.show()

# Get topic weights
topic_weights = []
for i, row_list in enumerate(lda_model[corpus]):
   topic_weights.append([w for i, w in row_list[0]])

arr = pd.DataFrame(topic_weights).fillna(0).values
arr = arr[np.amax(arr, axis=1) > 0.35]  # Optional filtering
topic_num = np.argmax(arr, axis=1)

# t-SNE for dimensionality reduction
tsne_model = TSNE(n_components=2, verbose=1, random_state=0, angle=.99, init='pca',
perplexity=2)
tsne_lda = tsne_model.fit_transform(arr)

# Plot t-SNE results with Bokeh
output_notebook()
mycolors = np.array([color for name, color in mcolors.TABLEAU_COLORS.items()])
plot = figure(title="t-SNE Clustering of LDA Topics", width=900, height=700)
plot.scatter(x=tsne_lda[:, 0], y=tsne_lda[:, 1], color=mycolors[topic_num])
show(plot)

# LDAvis interactive visualization
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim_models.prepare(lda_model, corpus, dictionary=id2word)
```

vis

```
# Fix NumPy dtype issue
!pip install --upgrade --force-reinstall numpy
```