INFO5731 Assignment: 4

This exercise will provide a valuable learning experience in working with text data and extracting features using various topic modeling algorithms. Key concepts such as Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA) and BERTopic.

Expectations:

- Students are expected to complete the exercise during lecture period to meet the active participation criteria of the course.
- Use the provided *.ipynb* document to write your code & respond to the questions. Avoid generating a new file.
- Write complete answers and run all the cells before submission.
- Make sure the submission is "clean"; *i.e.*, no unnecessary code cells.
- Once finished, allow shared rights from top right corner (*see Canvas for details*).

Total points: 100

NOTE: The output should be presented well to get **full points**

Late submissions will have a penalty of 10% of the marks for each day of late submission, and no requests will be answered. Manage your time accordingly.

Question 1 (20 Points)

Dataset: 20 Newsgroups dataset

Dataset Link: https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

Consider Random 2000 rows only

Generate K=10 topics by using LDA and LSA, then calculate coherence score and determine the optimized K value by the coherence score. Further, summarize and visualize each topics in you own words.

```
!pip install gensim
```

```
Collecting gensim
  Downloading gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_
Collecting numpy<2.0,>=1.18.5 (from gensim)
  Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 5.5 MB/s eta 0:0
Collecting scipy<1.14.0,>=1.7.0 (from gensim)
  Downloading scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.6/60.6 kB 5.4 MB/s eta 0:0
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-pack
Downloading gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.7/26.7 MB 74.0 MB/s eta 0:00
Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 94.0 MB/s eta 0:00
Downloading scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 38.6/38.6 MB 18.0 MB/s eta 0:00
Installing collected packages: numpy, scipy, gensim
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
  Attempting uninstall: scipy
    Found existing installation: scipy 1.14.1
    Uninstalling scipy-1.14.1:
      Successfully uninstalled scipy-1.14.1
Successfully installed gensim-4.3.3 numpy-1.26.4 scipy-1.13.1
```

```
!pip install numpy
!pip install scikit-learn
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/di
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/d
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
```

```
!pip install numpy==1.25.2
```

```
Collecting numpy==1.25.2
  Downloading numpy-1.25.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_
Downloading numpy-1.25.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.2/18.2 MB 109.1 MB/s eta 0:0
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 2.2.4
    Uninstalling numpy-2.2.4:
      Successfully uninstalled numpy-2.2.4
ERROR: pip's dependency resolver does not currently take into account all t
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.25.2
blosc2 3.2.1 requires numpy>=1.26, but you have numpy 1.25.2 which is incom
Successfully installed numpy-1.25.2
```

```
from sklearn.datasets import fetch_20newsgroups
import random
import pandas as pd

# Load full dataset
dataset_latest = fetch_20newsgroups(subset='all', remove=('headers', 'footers',

# Sample 2000 random posts
random.seed(45)
indices = random.sample(range(len(dataset_latest.data)), 2000)
sampled_data = [dataset_latest.data[i] for i in indices]
dataframe_late = pd.DataFrame(sampled_data, columns=["text"])
```

```
print(dataframe_late.head())
```

```
                                                text
0  element analysis, radiosity, distributed proce...
1  \n\n\nPlease explain how the removal of Israel...
2  Anyone have a phone number for Applied Enginee...
3  IRWIN suggests the use of pre-formatted tapes ...
4     \n     What a lie..!!??\n\n     Ask the vic...
```

```
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from nltk.stem import WordNetLemmatizer
import re

nltk.download('stopwords')
nltk.download('wordnet')

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess(text):
    text = re.sub(r'\W+', ' ', text.lower())
    tokens = text.split()
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop
    return " ".join(tokens)

dataframe['cleaned'] = dataframe['text'].apply(preprocess)
```

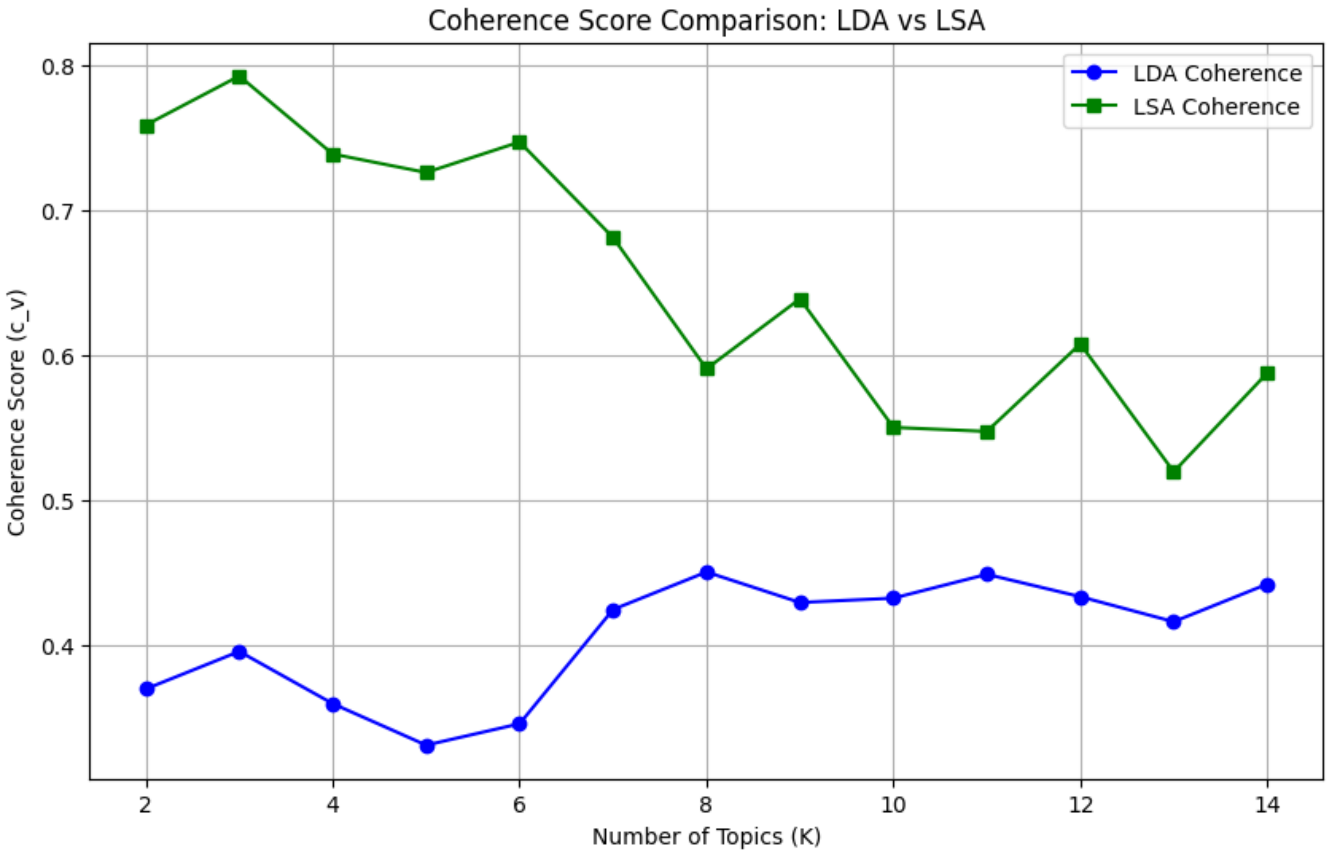
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
from sklearn.decomposition import LatentDirichletAllocation, TruncatedSVD
from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora.dictionary import Dictionary
import gensim
import numpy as np

# Tokenized docs
tokenized_docs = [doc.split() for doc in dataframe['cleaned']]

# Create Dictionary and Corpus
dictionary = Dictionary(tokenized_docs)
corpus = [dictionary.doc2bow(text) for text in tokenized_docs]

# Create TF and TF-IDF matrices
vectorizer = CountVectorizer(max_df=0.95, min_df=2)
tf = vectorizer.fit_transform(dataframe['cleaned'])

tfidf_vectorizer = TfidfVectorizer(max_df=0.95, min_df=2)
tfidf = tfidf_vectorizer.fit_transform(dataframe['cleaned'])

# LDA
lda = LatentDirichletAllocation(n_components=10, random_state=42)
lda_topics = lda.fit_transform(tf)

# LSA
lsa = TruncatedSVD(n_components=10, random_state=42)
lsa_topics = lsa.fit_transform(tfidf)
```

```
def compute_coherence_values(model_type, texts, dictionary, corpus, start=2, li
    coherence_scores = []
    for k in range(start, limit, step):
        if model_type == 'lda':
            model = gensim.models.LdaModel(corpus=corpus, id2word=dictionary, r
        elif model_type == 'lsa':
            model = gensim.models.LsiModel(corpus=corpus, id2word=dictionary, r
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=di
        coherence_scores.append((k, coherencemodel.get_coherence()))
    return coherence_scores

lda_coherence = compute_coherence_values('lda', tokenized_docs, dictionary, cor
lsa_coherence = compute_coherence_values('lsa', tokenized_docs, dictionary, cor
```

```
import matplotlib.pyplot as plt
```

```
# Unpack the scores
lda_k, lda_scores = zip(*lda_coherence)
lsa_k, lsa_scores = zip(*lsa_coherence)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(lda_k, lda_scores, marker='o', label='LDA Coherence', color='blue')
plt.plot(lsa_k, lsa_scores, marker='s', label='LSA Coherence', color='green')
plt.xlabel("Number of Topics (K)")
plt.ylabel("Coherence Score (c_v)")
plt.title("Coherence Score Comparison: LDA vs LSA")
plt.legend()
plt.grid(True)
plt.show()
```



```
best_lda_model = gensim.models.LdaModel(corpus=corpus, id2word=dictionary, num_

topics = best_lda_model.print_topics(num_words=10)
for idx, topic in topics:
    print(f"Topic {idx+1}: {topic}")
```

```
Topic 1: 0.008*"window" + 0.006*"would" + 0.005*"know" + 0.004*"system" + 0
Topic 2: 0.005*"file" + 0.004*"also" + 0.004*"graphic" + 0.003*"program" +
Topic 3: 0.005*"people" + 0.005*"would" + 0.004*"armenian" + 0.003*"time" +
Topic 4: 0.007*"would" + 0.005*"jpeg" + 0.005*"game" + 0.005*"image" + 0.00
Topic 5: 0.007*"drive" + 0.007*"would" + 0.005*"like" + 0.004*"system" + 0.
```

Start coding or generate with AI.

BERTopic

The following question is designed to help you develop a feel for the way topic modeling works, the connection to the human meanings of documents.

Dataset from **assignment-3** (text dataset) .

> Dont use any custom datasets.

> Dataset must have 1000+ rows, no duplicates and null values

Question 2 (20 Points)

Q2) **Generate K=10 topics by using BERTopic and then find optimal K value by the coherence score. Interpret each topic and visualize with suitable style.**

```
!pip install numpy==1.24.4
!pip install pandas==1.5.3
!pip install scikit-learn==1.1.3
!pip install umap-learn==0.5.3
!pip install hdbscan==0.8.29
!pip install sentence-transformers==2.2.2
!pip install gensim==4.3.1
!pip install bertopic[visualization]==0.16.0
```

```
Collecting numpy==1.24.4
  Downloading numpy-1.24.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_
Downloading numpy-1.24.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 21.0 MB/s eta 0:00
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
ERROR: pip's dependency resolver does not currently take into account all t
jaxlib 0.5.1 requires numpy>=1.25, but you have numpy 1.24.4 which is incom
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.24.4
treescope 0.1.9 requires numpy>=1.25.2, but you have numpy 1.24.4 which is
jax 0.5.2 requires numpy>=1.25, but you have numpy 1.24.4 which is incompat
pymc 5.21.2 requires numpy>=1.25.0, but you have numpy 1.24.4 which is inco
blosc2 3.2.1 requires numpy>=1.26, but you have numpy 1.24.4 which is incom
Successfully installed numpy-1.24.4
^C
^C
^C
^C
```

```
import os
os.kill(os.getpid(), 9)
```

```
!pip install --upgrade jax jaxlib
```

```
Requirement already satisfied: jax in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: jaxlib in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: ml_dtypes>=0.4.0 in /usr/local/lib/python3.1
Requirement already satisfied: numpy>=1.25 in /usr/local/lib/python3.11/dis
Requirement already satisfied: opt_einsum in /usr/local/lib/python3.11/dist
Requirement already satisfied: scipy>=1.11.1 in /usr/local/lib/python3.11/d
```

```
!pip install numpy==1.25
```

```
Collecting numpy==1.25
  Downloading numpy-1.25.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_
Downloading numpy-1.25.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x8
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.6/17.6 MB 67.0 MB/s eta 0:00
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.24.4
    Uninstalling numpy-1.24.4:
      Successfully uninstalled numpy-1.24.4
ERROR: pip's dependency resolver does not currently take into account all t
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.25.0
treescope 0.1.9 requires numpy>=1.25.2, but you have numpy 1.25.0 which is
blosc2 3.2.1 requires numpy>=1.26, but you have numpy 1.25.0 which is incom
Successfully installed numpy-1.25.0
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from bertopic import BERTopic
from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora import Dictionary

k = 10
df = pd.read_csv('/content/amazon_reviews_cleaned.csv', usecols=['Cleaned_Revie
Reviews = df.Cleaned_Review.to_list()  # Define 'abstracts' here
df.head()
```

	Cleaned_Review
0	this along tempered glass screen lens protecto...
1	ive tried iphones either stiff color fade time...
2	love perfectly somewhat light weight provide d...
3	think safe say know product best iphone 16 pro...
4	love design sleek make look unique high tech h...

```
topic_model_Review = BERTopic()
topics_default, _ = topic_model_default.fit_transform(Reviews)
```

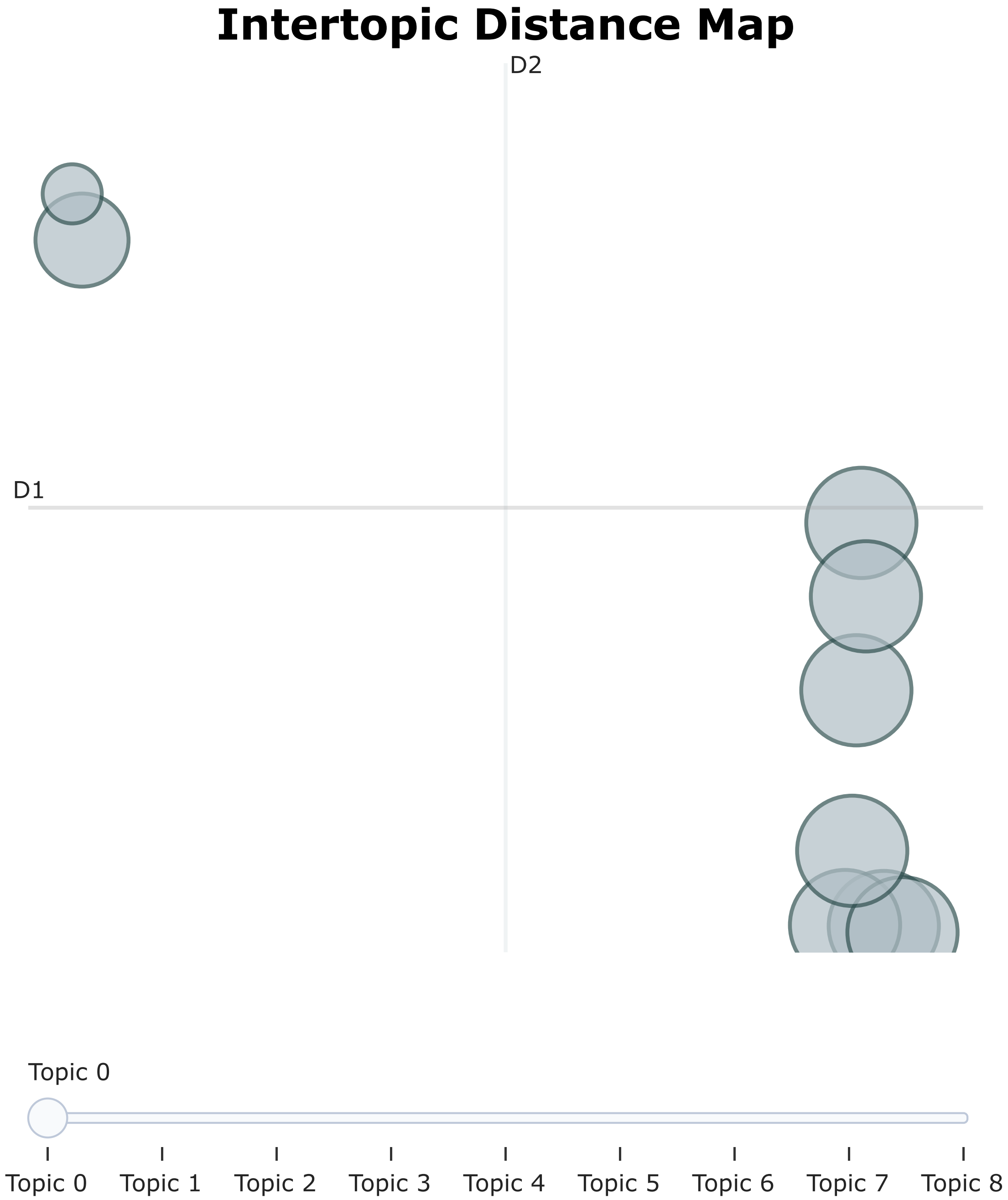
```
from sentence_transformers import SentenceTransformer
from sklearn.feature_extraction.text import CountVectorizer
embedding_model = SentenceTransformer("all-MiniLM-L6-v2")
embeddings = embedding_model.encode(Reviews, show_progress_bar=True)
topics, probs = topic_model_Review.fit_transform(Reviews, embeddings)
```

```
Batches: 100%                                        26/26 [00:00<00:00, 41.32it/s]
```

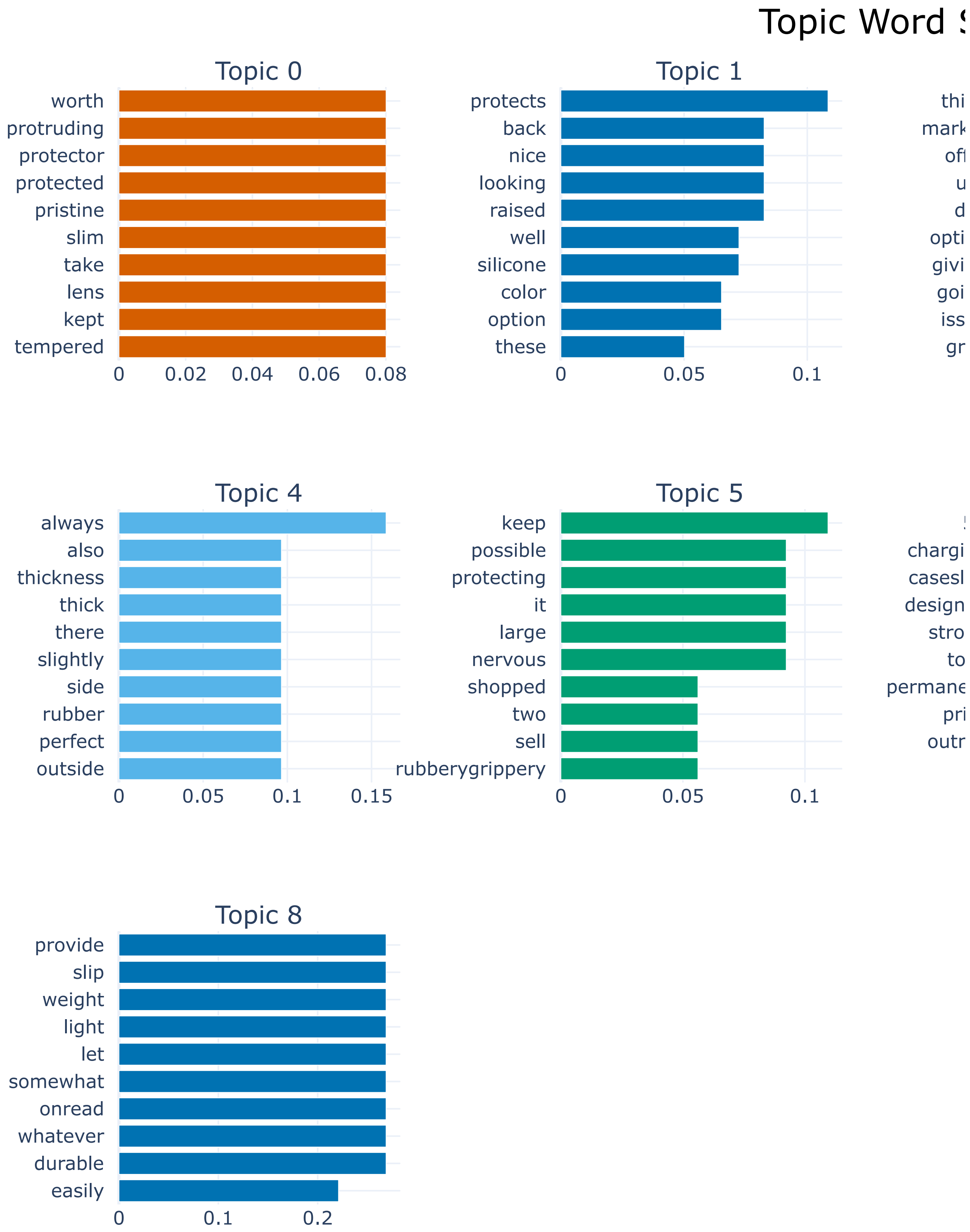
```
topic_model_Review.get_topic_info()
```

	Topic	Count	Name	Representation	Representat
0	0	101	0_worth_protruding_protector_protected	[worth, protruding, protector, protected, pris...	[this along glass s
1	1	101	1_protects_back_nice_looking	[protects, back, nice, looking, raised, well, ...	[ive tried ipho stiff color
2	2	101	2_think_market_offer_use	[think, market, offer, use, day, option, givin...	[think safe product best
3	3	101	3_scratch_have_niceread_may	[scratch, have, niceread, may, tech, since, un...	[love design s look unique h
				[always, also,	[alway

```
topic_model_Review.visualize_topics()
```



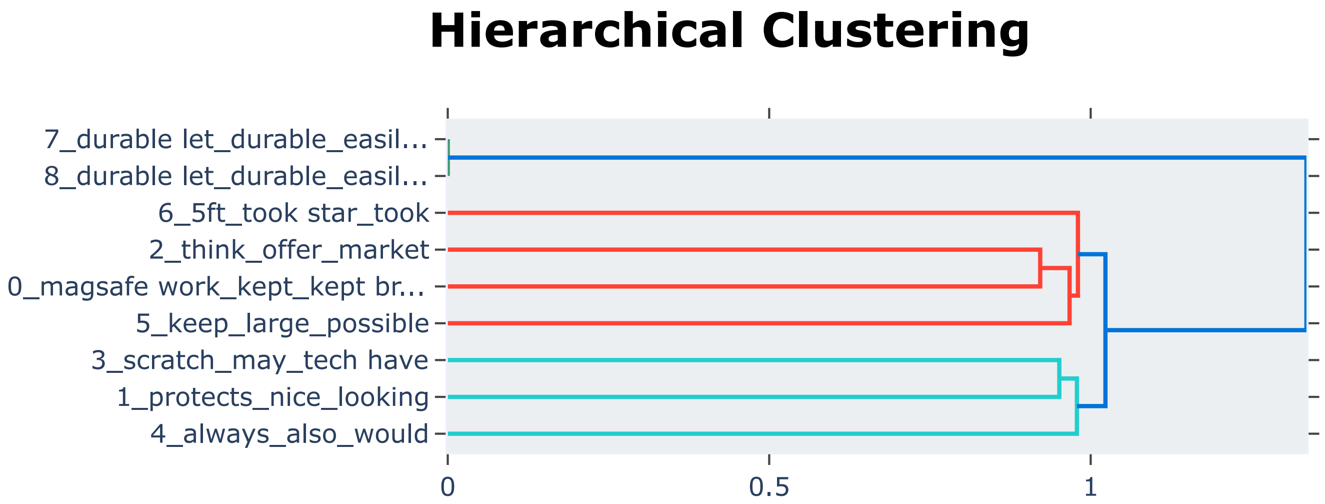
```
topic_model_Review.visualize_barchart(top_n_topics=10, n_words = 40, width = 3(
```


Topic Word S
Topic 0
worth
protruding
protector
protected
pristine
slim
take
lens
kept
tempered
0
0.02
0.04
0.06
0.08
Topic 1
protects
back
nice
looking
raised
well
silicone
color
option
these
0
0.05
0.1
thi
mark
oft
u
d
opti
givi
goi
iss
gr
Topic 4
always
also
thickness
thick
there
slightly
side
rubber
perfect
outside
0
0.05
0.1
0.15
Topic 5
keep
possible
protecting
it
large
nervous
shopped
two
sell
rubberygrippery
0
0.05
0.1
chargi
casesl
design
stro
to
permane
pri
outr
Topic 8
provide
slip
weight
light
let
somewhat
onread
whatever
durable
easily
0
0.1
0.2

```
import numpy as np
from sklearn.cluster import DBSCAN
def custom_distance(embeddings):
    """Calculate custom distance matrix."""
    # Calculate cosine similarity
    sim_matrix = np.dot(embeddings, embeddings.T)

    # Ensure non-negativity by adding 1 and dividing by 2
    # This ensures values range from 0 to 1
    dist_matrix = (1 - (sim_matrix + 1) / 2)
    return dist_matrix
dbscan_model = DBSCAN(eps=0.3, min_samples=3, metric='cosine')
topic_model_Review = BERTopic(
    embedding_model=embedding_model,
    hdbscan_model=dbscan_model,  # Use your custom DBSCAN model
    vectorizer_model=CountVectorizer(ngram_range=(1, 2)),
    verbose=True
)


embedding_model = SentenceTransformer("all-MiniLM-L6-v2")
embeddings = embedding_model.encode(Reviews, show_progress_bar=True)
topics, probs = topic_model_Review.fit_transform(Reviews, embeddings)
```

```
Batches: 100%                                26/26 [00:00<00:00, 48.57it/s]
2025-04-08 02:49:56,338 - BERTopic - Dimensionality - Fitting the dimension
2025-04-08 02:50:01,236 - BERTopic - Dimensionality - Completed ✓
2025-04-08 02:50:01,238 - BERTopic - Cluster - Start clustering the reduced
2025-04-08 02:50:01,264 - BERTopic - Cluster - Completed ✓
2025-04-08 02:50:01,275 - BERTopic - Representation - Fine-tuning topics us
2025-04-08 02:50:01,400 - BERTopic - Representation - Completed ✓
```

```
topic_model_Review.visualize_hierarchy(top_n_topics=10, width = 700, height = 7
```



```
from gensim.models import CoherenceModel
from gensim.corpora import Dictionary

def calculate_coherence_score(data, min_k=2, max_k=10):
    coherence_scores = []

    for k in range(min_k, max_k + 1):
        # Train BERTopic model with K topics
        model = BERTopic(nr_topics=k)
        topics, _ = model.fit_transform(data)

        # Extract topic words
        topic_words = [ [word for word, _ in model.get_topic(topic)] for topic i

        # Prepare data for Coherence Model
        tokenized_data = [doc.split() for doc in data]
        dictionary = Dictionary(tokenized_data)

        # Compute Coherence Score
        coherence_model = CoherenceModel(topics=topic_words, dictionary=dictiona
        coherence_score = coherence_model.get_coherence()
        coherence_scores.append((k, coherence_score))

        print(f"K={k}, Coherence Score={coherence_score}")

    return coherence_scores
```

```
# Example usage with 'Reviews' dataset
coherence_scores = calculate_coherence_score(Reviews, min_k=2, max_k=20)
```

```
K=2, Coherence Score=0.7506595823069511
K=3, Coherence Score=0.8228912618696967
K=4, Coherence Score=0.8579184505045191
K=5, Coherence Score=0.9595597077272504
K=6, Coherence Score=0.9561777421137253
K=7, Coherence Score=0.9891905668898631
K=8, Coherence Score=0.993910417921153
K=9, Coherence Score=0.9942859710534133
K=10, Coherence Score=0.9942859710534131
K=11, Coherence Score=0.9942859710534133
K=12, Coherence Score=0.9942859710534133
K=13, Coherence Score=0.9942859710534133
K=14, Coherence Score=0.9942859710534131
K=15, Coherence Score=0.9942859710534133
K=16, Coherence Score=0.9942859710534133
K=17, Coherence Score=0.9942859710534133
K=18, Coherence Score=0.9942859710534133
K=19, Coherence Score=0.9942859710534133
K=20, Coherence Score=0.9942859710534133
```

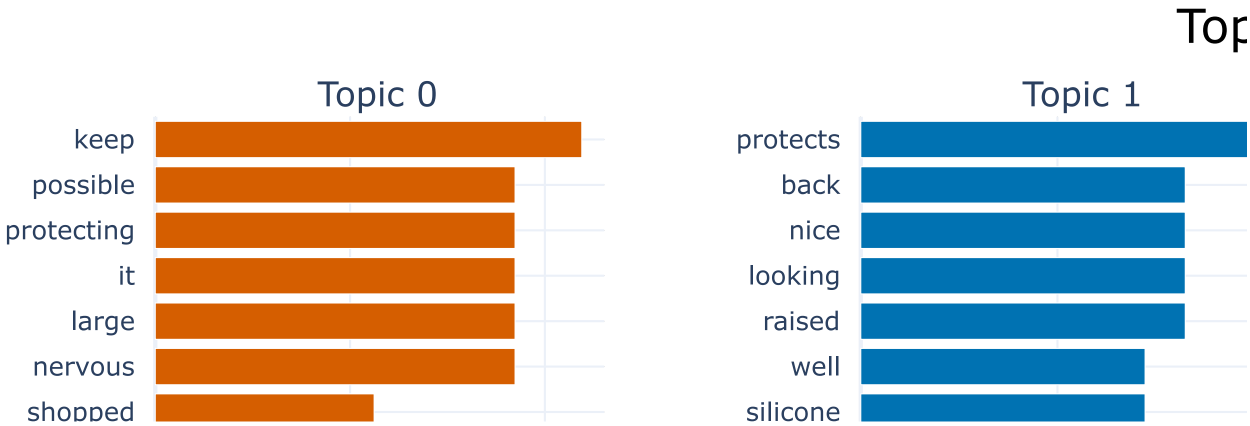
```
optimal_k = 10
final_Review = BERTopic(nr_topics=optimal_k)
new_topics, new_ = final_Review.fit_transform(Reviews)
```

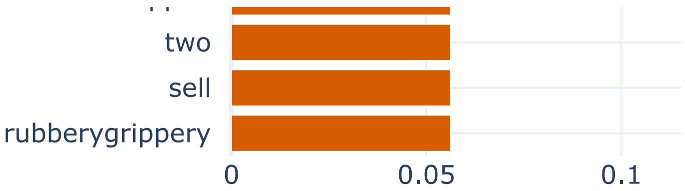
```
def compute_coherence(texts, min_k=2, max_k=20):
    coherence_scores = []
    for k in range(min_k, max_k + 1):
        model = BERTopic(nr_topics=k)
        topics, _ = model.fit_transform(texts)
        topic_words = [list(dict(model.get_topic(i)).keys()) for i in range(k)]

        # Create dictionary and corpus for coherence calculation
        dictionary = Dictionary([words for words in topic_words])
        corpus = [dictionary.doc2bow(words) for words in topic_words]

        # Calculate coherence score using the c_v metric
        coherence_model = CoherenceModel(topics=topic_words, texts=[text.split(
                                         dictionary=dictionary, coherence='c_v'
        coherence_scores.append((k, coherence_model.get_coherence()))

    return coherence_scores
```

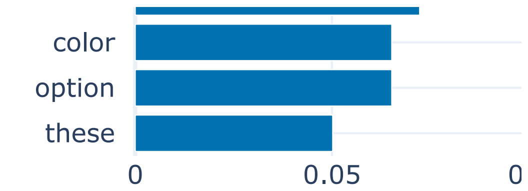
```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

def custom_distance(x, y):
    """Calculate custom distance matrix using cosine similarity."""

    # Ensure values are within [0, 2] range for distance metric
    # Reshape x and y to 2D arrays if necessary
    x = x.reshape(1, -1)
    y = y.reshape(1, -1)

    # Clip cosine similarity to ensure it's within [0, 1]
    sim = cosine_similarity(x, y)[0][0]
    sim = np.clip(sim, 0, 1)  # Clip values to be between 0 and 1

    dist = 1 - sim

    return dist

# Custom DBSCAN model using custom_distance
dbscan_model = DBSCAN(eps=0.3, min_samples=3, metric=custom_distance)

# Initialize BERTopic with custom distance
topic_model_re = BERTopic(
    embedding_model=embedding_model,
    hdbscan_model=dbscan_model,  # Use your custom DBSCAN model
    vectorizer_model=CountVectorizer(ngram_range=(1, 2)),
    verbose=True
)


# Train BERTopic model with the optimal K
topic_model_re = BERTopic(nr_topics=optimal_k)
topics, probs = topic_model_re.fit_transform(abstracts)

# Interpret the topics by examining their top words
print("\nTopic Interpretation (Top Words):")
for topic_id in range(optimal_k):
    print(f"Topic {topic_id}:")
    print(topic_model_re.get_topic(topic_id))
    print("\n")

# Visualize the topics
topic_model_re.visualize_topics()
topic_model_re.visualize_barchart(top_n_topics=12, n_words=10, width=350, heigh
```

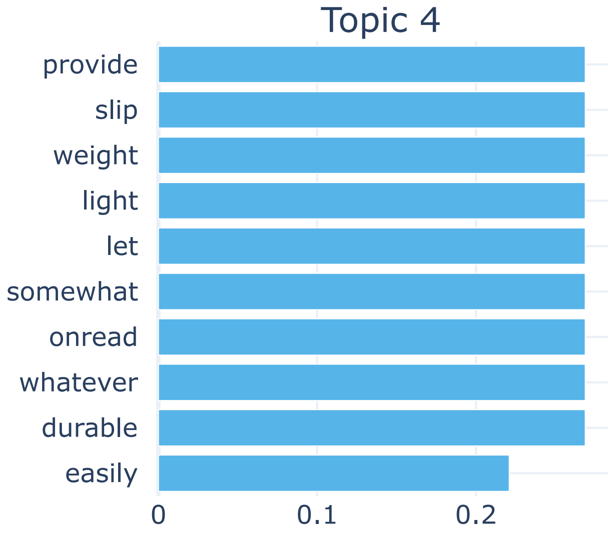
```
Topic Interpretation (Top Words):
```

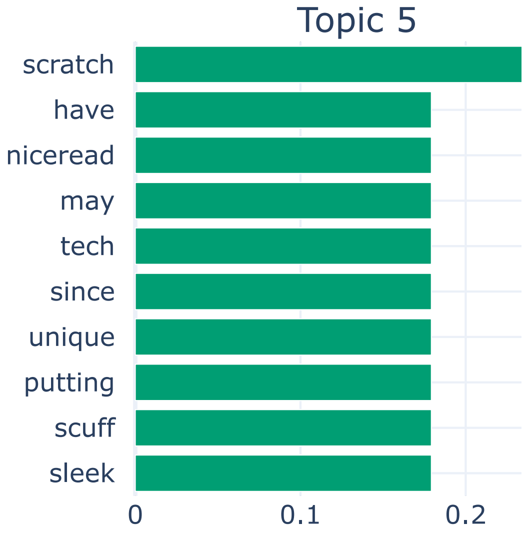
```
Topic 0:
[('keep', 0.10946471037408478), ('possible', 0.09234372067175238), ('protec


Topic 1:
[('protects', 0.10841870076230814), ('back', 0.0824937238000988), ('nice',


Topic 2:
[('always', 0.15864177653865152), ('also', 0.09650589038341623), ('thicknes


Topic 3:
[('think', 0.0711152791380162), ('market', 0.0711152791380162), ('offer', 0


Topic 4:
[('provide', 0.2688378374966595), ('slip', 0.2688378374966595), ('weight',


Topic 5:
[('scratch', 0.25813976371978126), ('have', 0.179225224997773), ('niceread'


Topic 6:
[('5ft', 0.129783783619077), ('charging', 0.129783783619077), ('caseslim',


Topic 7:
[('worth', 0.08007935585006878), ('protruding', 0.08007935585006878), ('pro


Topic 8:
[('provide', 0.2688378374966595), ('slip', 0.2688378374966595), ('weight',


Topic 9:
False
```

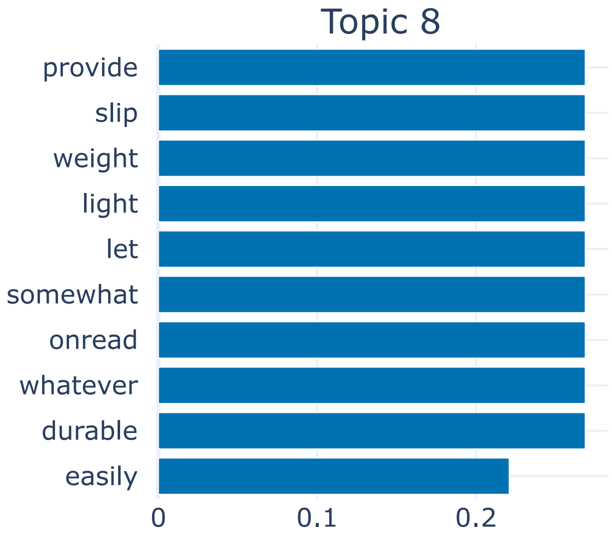



two
sell
rubberygrippery
0
0.05
0.1

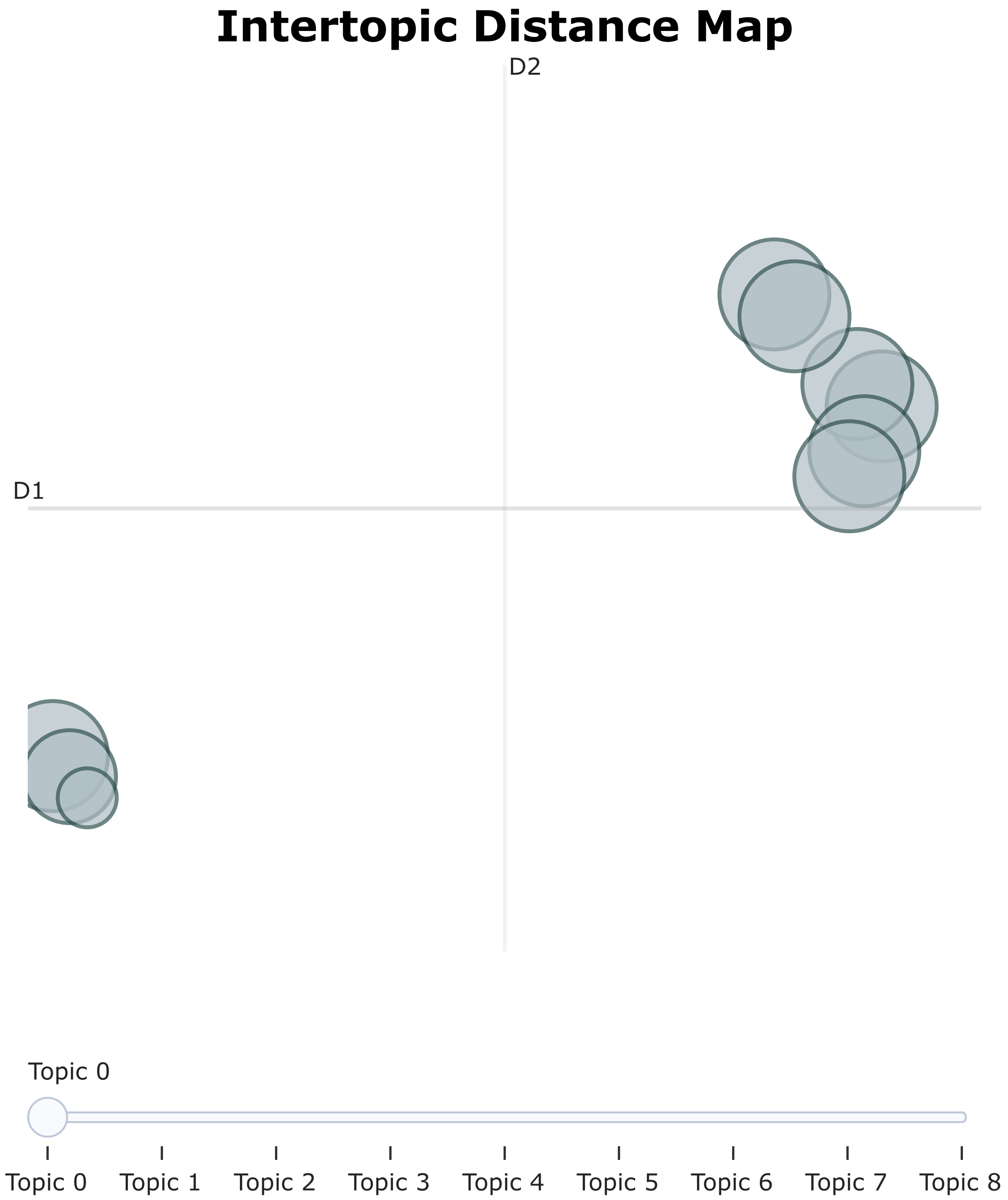

color
option
these
0
0.05
0


Topic 4
provide
slip
weight
light
let
somewhat
onread
whatever
durable
easily
0
0.1
0.2


Topic 5
scratch
have
niceread
may
tech
since
unique
putting
scuff
sleek
0
0.1
0.2

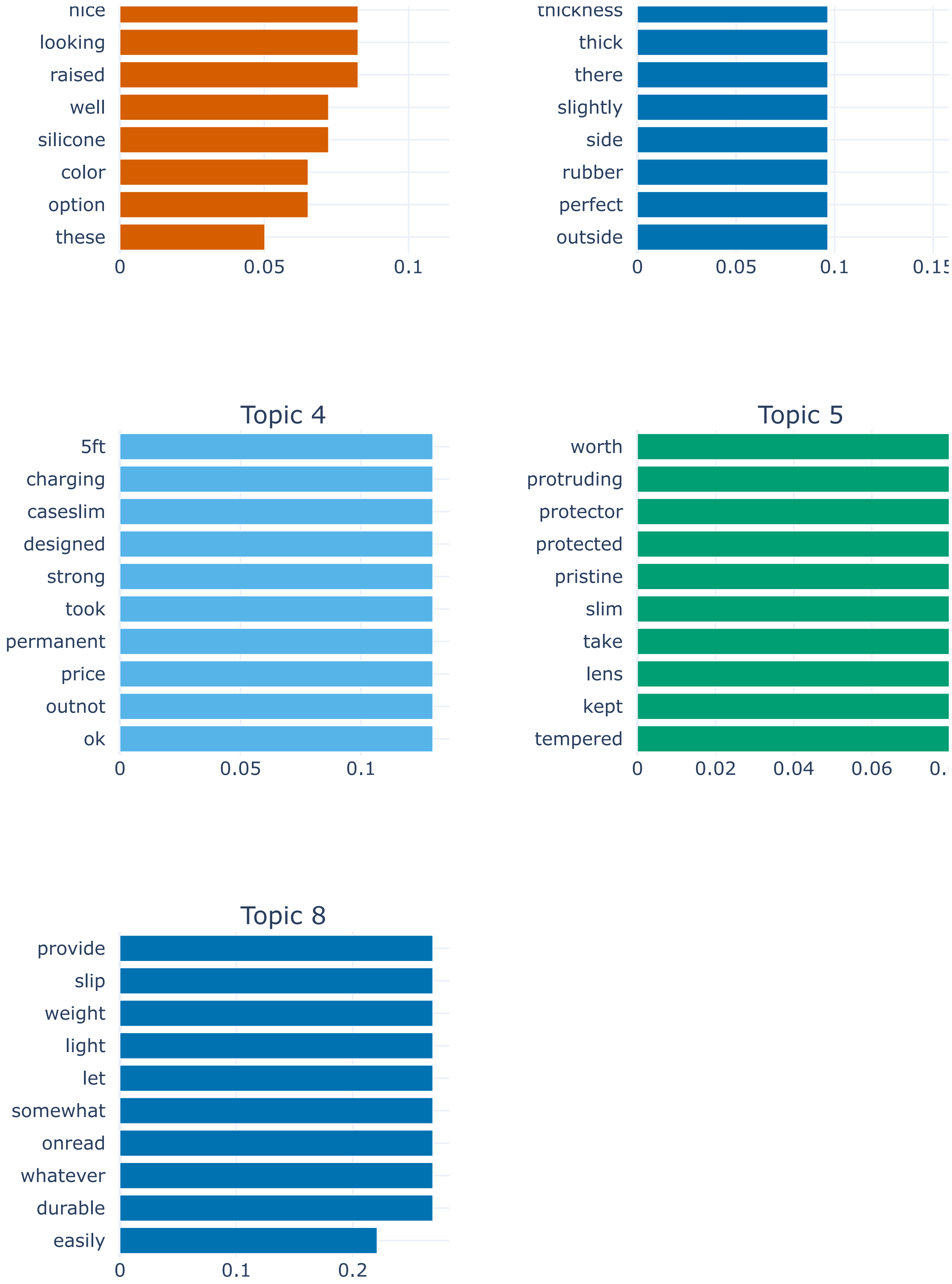

Topic 8
provide
slip
weight
light
let
somewhat
onread
whatever
durable
easily
0
0.1
0.2

```
final_Review.visualize_topics()
```



```
final_Review.visualize_barchart(top_n_topics=12, n_words = 10, width = 350, hei
```

Top
Topic 0
protects
back
Topic 1
always
also


nice
looking
raised
well
silicone
color
option
these
0
0.05
0.1
thickness
thick
there
slightly
side
rubber
perfect
outside
0
0.05
0.1
0.15
Topic 4
5ft
charging
caseslim
designed
strong
took
permanent
price
outnot
ok
0
0.05
0.1
Topic 5
worth
protruding
protector
protected
pristine
slim
take
lens
kept
tempered
0
0.02
0.04
0.06
0.
Topic 8
provide
slip
weight
light
let
somewhat
onread
whatever
durable
easily
0
0.1
0.2

Question 3 (25 points)

Dataset Link: 20 Newsgroup Dataset (Random 2000 values)

Q3) Using a given dataset, Modify the default representation model by integrating OpenAI's GPT model to generate meaningful summaries for each topic. Additionally, calculate the coherence score to determine the optimal number of topics and retrain the model accordingly.

Usefull Link:
https://maartengr.github.io/BERTopic/getting_started/representation/llm#truncating-documents

```
import pandas as pd
import random
from sklearn.datasets import fetch_20newsgroups

# Load dataset and sample 2000 rows
data = fetch_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes')
sampled_data = random.sample(data.data, 2000)

# Convert to DataFrame
dataframe_2 = pd.DataFrame(sampled_data, columns=['text'])
print(dataframe_2.head())
```

```
                                                text
0  the 'Center for Policy Research' writes...\n \...
1  Apparently, the only place to take the MSF cou...
2  Are you using Windows 3.0 or 3.1? If you're st...
3  \n\n1/64th or 63/64th?\n\nI must congratulate ...
4  \n\nNo.  The simple $25 style IDE controller d...
```

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# Download necessary NLTK resources (run once)
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt_tab') # Download the missing punkt_tab data

# Preprocessing tools
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Preprocessing function
def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = nltk.word_tokenize(text)
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop
    return " ".join(tokens)

# Apply preprocessing
dataframe_2['cleaned'] = dataframe_2['text'].apply(preprocess)
print(dataframe_2[['text', 'cleaned']].head())
```

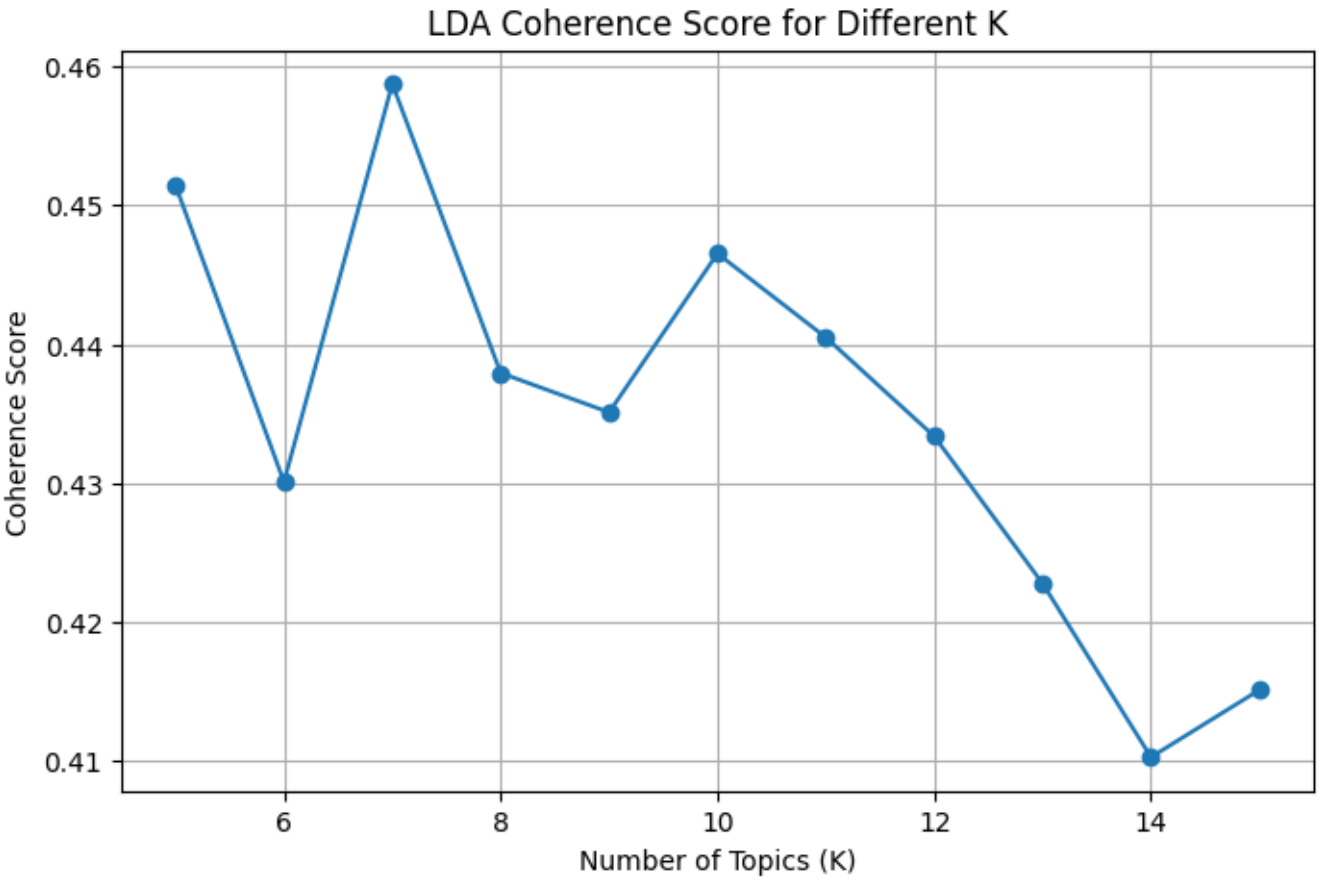
```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
                                               text  \
0  the 'Center for Policy Research' writes...\n \...
1  Apparently, the only place to take the MSF cou...
2  Are you using Windows 3.0 or 3.1? If you're st...
3  \n\n1/64th or 63/64th?\n\nI must congratulate ...
4  \n\nNo.  The simple $25 style IDE controller d...

                                            cleaned
0  center policy research writes considering murd...
1  apparently place take course around community ...
2  using window youre still devotes twice much me...
3  must congratulate analytical excellent reporta...
4  simple style controller performs data transfer...
```

```
from gensim import corpora

# Tokenize preprocessed text
texts = [doc.split() for doc in dataframe_2['cleaned']]

# Create dictionary and corpus
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]

print(f"Sample dictionary tokens: {dictionary.token2id}")
print(f"Sample corpus: {corpus[0][:20]}")
#corpus
```

```
Sample dictionary tokens: {'absurdity': 0, 'accomplished': 1, 'advice': 2,
Sample corpus: [(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 3), (7,
from gensim.models import LdaModel, CoherenceModel
import matplotlib.pyplot as plt

coherence_scores = []

for k in range(5, 16):
    lda = LdaModel(corpus=corpus, id2word=dictionary, num_topics=k, random_stat
    cm = CoherenceModel(model=lda, texts=texts, dictionary=dictionary, coherenc
    coherence = cm.get_coherence()
    coherence_scores.append((k, coherence))
    print(f"K={k}, Coherence Score={coherence:.4f}")
```

```
WARNING:gensim.models.ldamodel:too few updates, training might not converge
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=5, Coherence Score=0.4514
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=6, Coherence Score=0.4301
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=7, Coherence Score=0.4587
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=8, Coherence Score=0.4379
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=9, Coherence Score=0.4351
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=10, Coherence Score=0.4465
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=11, Coherence Score=0.4405
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=12, Coherence Score=0.4334
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=13, Coherence Score=0.4228
WARNING:gensim.models.ldamodel:too few updates, training might not converge
K=14, Coherence Score=0.4103
K=15, Coherence Score=0.4151
```

```
# Plot coherence scores
k_vals, scores = zip(*coherence_scores)
plt.figure(figsize=(8, 5))
plt.plot(k_vals, scores, marker='o')
plt.xlabel("Number of Topics (K)")
plt.ylabel("Coherence Score")
plt.title("LDA Coherence Score for Different K")
plt.grid(True)
plt.show()

# Find best K
best_k = max(coherence_scores, key=lambda x: x[1])[0]
print(f"\nBest K based on coherence: {best_k}")
```



```
Best K based on coherence: 7
```

```
# Train LDA model with best K
lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=best_k, ranc

# Print top keywords for each topic
topics = lda_model.show_topics(num_topics=best_k, num_words=10, formatted=False

for idx, topic in topics:
    keywords = [word for word, prob in topic]
    print(f"Topic {idx+1}: {', '.join(keywords)}")
```

```
WARNING:gensim.models.ldamodel:too few updates, training might not converge
Topic 1: would, know, like, people, think, also, file, first, image, much
Topic 2: would, people, dont, time, make, know, also, think, even, file
Topic 3: file, would, dont, window, like, people, know, think, many, maxaxa
Topic 4: maxaxaxaxaxaxaxaxaxaxaxaxaxax, people, would, like, file, know,
Topic 5: would, people, year, time, dont, like, well, think, could, file
Topic 6: image, know, like, would, dont, problem, also, time, good, file
Topic 7: maxaxaxaxaxaxaxaxaxaxaxaxaxax, would, file, also, time, program,
```

```
!pip install openai==0.28
```

```
Requirement already satisfied: openai==0.28 in /usr/local/lib/python3.11/di
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.11/
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
```

```
import openai

openai.api_key = "sk-proj-Q3l1BFV8fp214Nf9OBpZxDANZhFu8rgsJTqtu5xal9UbdP_uTcOTe

def gpt_topic_summary(keywords):
    prompt = f"Generate a short, meaningful summary for a topic based on these
    response = openai.ChatCompletion.create( # Use openai.ChatCompletion.create
        model="gpt-3.5-turbo", # Specify the model for chat completion
        messages=[
            {"role": "system", "content": "You are a helpful assistant that sum
            {"role": "user", "content": prompt}
        ],
        max_tokens=50
    )
    return response.choices[0].message.content.strip() # Access the summary tex

# Generate summaries
print("\n=== GPT Summaries ===")
for idx, topic in topics:
    keywords = [word for word, prob in topic]
    summary = gpt_topic_summary(keywords)
    print(f"Topic {idx+1}: {summary}")
```

```
=== GPT Summaries ===
Topic 1: People would like to know how much others think about a file or im
Topic 2: The topic discusses how many people would like to make the most of
Topic 3: The topic involves handling files in a window, and how many people
Topic 4: Maxaxaxaxaxaxaxaxaxaxaxaxaxaxax is a program that people would lik
Topic 5: In a year where people don't have much time to spare, some might n
Topic 6: The importance of image files: Knowing how to handle them, what ty
Topic 7: The topic covers the importance of maximizing efficiency in file m
```

Question 4 (35 Points)

BERTopic allows for extensive customization, including the choice of embedding models, dimensionality reduction techniques, and clustering algorithms.

Dataset Link: 20 Newsgroup Dataset (Random 2000 values)

4)

4.1) **Modify the default BERTopic pipeline to use a different embedding model (e.g., Sentence-Transformers) and a different clustering algorithm (e.g., DBSCAN instead of HDBSCAN).

4.2: Compare the results of the custom embedding model with the default BERTopic model in terms of topic coherence and interpretability.

4.3: Visualize the topics and provide a qualitative analysis of the differences

**

Usefull Link :https://www.pinecone.io/learn/bertopic/

```
import pandas as pd
import random
from sklearn.datasets import fetch_20newsgroups

# Load dataset and sample 2000 rows
data = fetch_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes')
sampled_data = random.sample(data.data, 2000)

# Convert to DataFrame
dataframe_3 = pd.DataFrame(sampled_data, columns=['text'])
print(dataframe_3.head())
```

```
                                                text
0  \nReplace "Stephen" with "David Joslin," since...
1  \n  How about a Geeky temporary tatoo?  I mean...
2  \nDoes GX take the place of 32 bit QD or add t...
3  After the marvelous "time-out" call by Chris W...
4  My fiance has a pc-junior and wants to upgrade...
```

```
!pip install bertopic
```

```
Collecting bertopic
  Downloading bertopic-0.17.0-py3-none-any.whl.metadata (23 kB)
Requirement already satisfied: hdbscan>=0.8.29 in /usr/local/lib/python3.11
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: pandas>=1.1.5 in /usr/local/lib/python3.11/d
Requirement already satisfied: plotly>=4.7.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: scikit-learn>=1.0 in /usr/local/lib/python3.
Requirement already satisfied: sentence-transformers>=0.4.1 in /usr/local/l
Requirement already satisfied: tqdm>=4.41.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: umap-learn>=0.5.0 in /usr/local/lib/python3.
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: joblib>=1.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/local/li
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/py
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.11/d
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.1
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-pac
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.11.0->sentence-t
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.11.0->sentence
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.11.0->sentence-t
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.11.0->sentence-transf
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.11.0->sentence-trans
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.11.0->sentence-transf
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.11.0->sentence-tra
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.w
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.11.0->sentence-tra
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.w
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.11.0->sentence-t
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/p
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.11.0->sentence-tr
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3
```

```
!pip install openai==0.27.8
```

```
Collecting openai==0.27.8
  Downloading openai-0.27.8-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.11/
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
Downloading openai-0.27.8-py3-none-any.whl (73 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.6/73.6 kB 2.6 MB/s eta 0:00:
Installing collected packages: openai
  Attempting uninstall: openai
    Found existing installation: openai 0.28.0
    Uninstalling openai-0.28.0:
      Successfully uninstalled openai-0.28.0
Successfully installed openai-0.27.8
```

```
from bertopic import BERTopic
from sklearn.cluster import DBSCAN
from sentence_transformers import SentenceTransformer
from sklearn.feature_extraction.text import CountVectorizer
embedding_model = SentenceTransformer("all-MiniLM-L6-v2")


# Generate embeddings, accessing the 'text' column of the DataFrame
embeddings = embedding_model.encode(dataframe_3['text'].tolist(), show_progress

# Custom DBSCAN model
dbscan_model = DBSCAN(eps=0.3, min_samples=3, metric='cosine')
```

```
Batches: 100%                                     63/63 [03:11<00:00,  2.53it/s]
```

```
topic_model = BERTopic(
    embedding_model=embedding_model,
    hdbscan_model=dbscan_model,
    vectorizer_model=CountVectorizer(ngram_range=(1, 2)),
    verbose=True
)

# 5. Fit the model with embeddings
topics, probs = topic_model.fit_transform(dataframe_3['text'], embeddings)
```

```
2025-04-07 23:42:32,130 - BERTopic - Dimensionality - Fitting the dimension
2025-04-07 23:42:40,734 - BERTopic - Dimensionality - Completed ✓
2025-04-07 23:42:40,736 - BERTopic - Cluster - Start clustering the reduced
2025-04-07 23:42:40,821 - BERTopic - Cluster - Completed ✓
2025-04-07 23:42:40,830 - BERTopic - Representation - Fine-tuning topics us
2025-04-07 23:42:44,077 - BERTopic - Representation - Completed ✓
```

```
print(topic_model.get_topic_info())

# Show top keywords per topic
for topic_num in topic_model.get_topics().keys():
    print(f"Topic {topic_num}: {topic_model.get_topic(topic_num)}")
```

```
   Topic  Count                                              Name  \
0      0   1934                                  0_the_to_of_and
1      1     66  1_yep hello_hi art_mistake yep_ditto chris

                                     Representation  \
0     [the, to, of, and, is, in, that, it, for, you]
1  [yep hello, hi art, mistake yep, ditto chris, ...

                                Representative_Docs
0  [\nIndeed, the immediate context [NASB] is:\n\...
1    [\nYep.\n, \n\nDitto,, \n\n\n\n\nMy mistake.  ]
Topic 0: [('the', np.float64(0.07623012783659086)), ('to', np.float64(0.047
Topic 1: [('yep hello', np.float64(0.8476598179085998)), ('hi art', np.floa
```

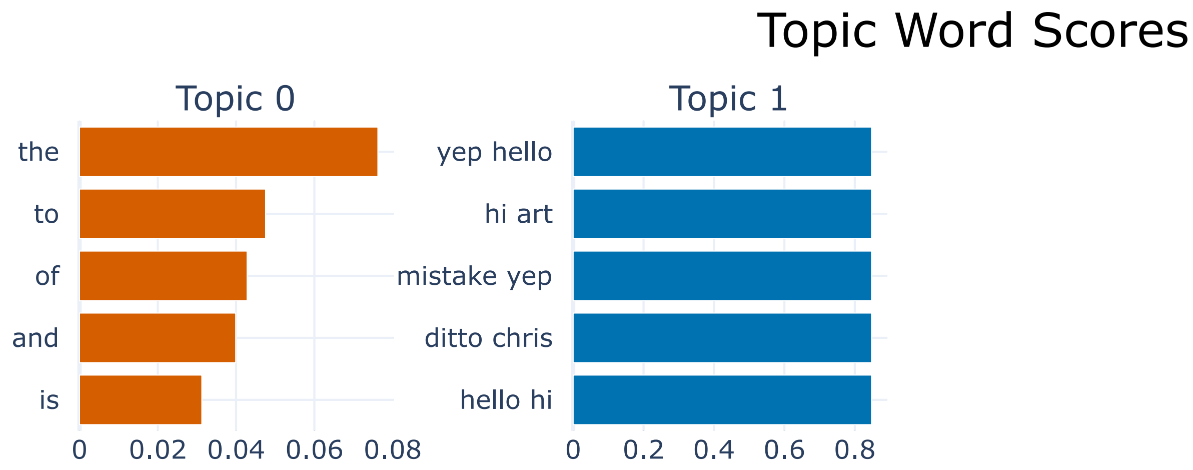
```
print(set(topics))
```

```
{0, 1}
```

```
topic_info = topic_model.get_topic_info()
print(topic_info)
```

```
   Topic  Count                                        Name  \
0      0   1934                             0_the_to_of_and
1      1     66  1_yep hello_hi art_mistake yep_ditto chris

                                   Representation  \
0     [the, to, of, and, is, in, that, it, for, you]
1  [yep hello, hi art, mistake yep, ditto chris, ...

                               Representative_Docs
0  [\nIndeed, the immediate context [NASB] is:\n\...
1    [\nYep.\n, \n\nDitto,, \n\n\n\n\nMy mistake.  ]
```

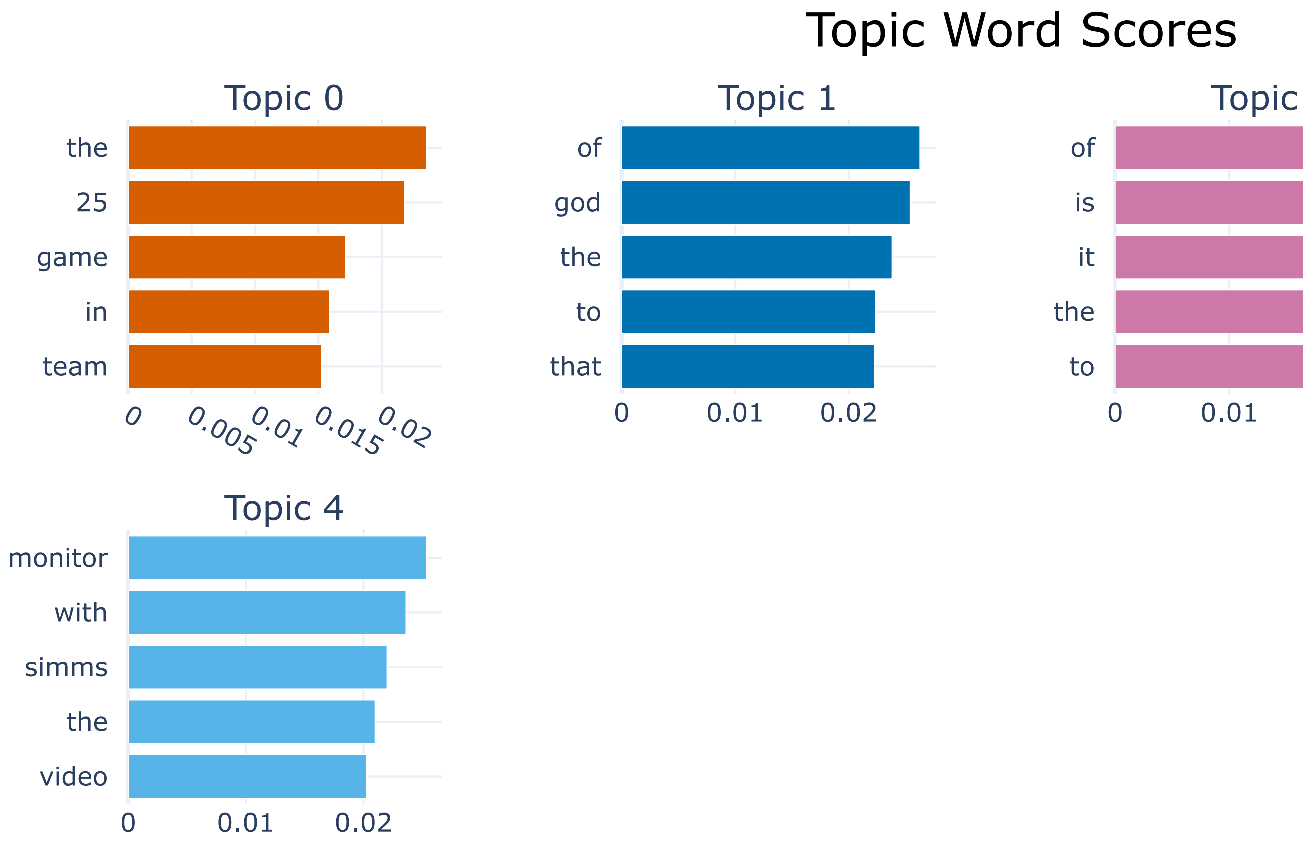
```
# Generate visualizations safely
topic_model.visualize_barchart(top_n_topics=5)
```



```
embeddings = embedding_model.encode(dataframe_3['text'].tolist(), show_progress
```

```
topic_model_default = BERTopic()
topics_default, _ = topic_model_default.fit_transform(dataframe_3['text'])
```

```
topic_model_default.visualize_barchart(top_n_topics=5)
```



```
print("\nTop Words per Topic (Default BERTopic):")
print(topic_model_default.get_topic_info())
for topic_num in topic_model_default.get_topics().keys():
    print(f"Topic {topic_num}: {topic_model_default.get_topic(topic_num)}")

print("\nTop Words per Topic (Custom BERTopic):")
print(topic_model.get_topic_info())
for topic_num in topic_model.get_topics().keys():
    print(f"Topic {topic_num}: {topic_model.get_topic(topic_num)}")
```

```
Top Words per Topic (Default BERTopic):
   Topic  Count                                         Name  \
0     -1    530                            -1_the_to_that_is
1      0    187                             0_the_25_game_in
2      1    147                              1_of_god_the_to
3      2     96                               2_of_is_it_the
```

```
4       3     94                            3_car_the_it_for
5       4     76                    4_monitor_with_simms_the
6       5     67                             5_gun_the_of_to
7       6     67                         6_for_sale_price_or
8       7     64                           7_key_the_chip_to
9       8     57                  8_openwindows_you_or_image
10      9     44                            9_jews_of_the_to
11     10     44                         10_the_was_fbi_they
12     11     41  11_phd_meteorologist_professor_atmospheric
13     12     40                        12_space_the_nasa_to
14     13     38             13_disk_drive_drives_controller
15     14     38                          14_the_and_were_of
16     15     36                          15_bike_the_to_you
17     16     36                           16_you_is_are_gay
18     17     35                         17_ditto_ignore_hi_
19     18     29             18_window_the_application_event
20     19     26                   19_wire_wiring_the_ground
21     20     25                                      20____
22     21     21               21_japanese_email_courses_and
23     22     18               22_windows_memory_problem_dos
24     23     17               23_define_endif_colormap_echo
25     24     17                   24_innovative_mac_os2_the
26     25     16                   25_printer_scanner_it_dpi
27     26     15                     26_files_mode_modes_the
28     27     14                         27_energy_of_the_is
29     28     14                      28_ftp_site_sites_list
30     29     13                  29_keyboard_and_window_the
31     30     13         30_clinton_school_projects_stimulus
32     31     13               31_dos_file_wintrumpet_format
33     32     12            32_radar_detector_detectors_unit

                                       Representation  \
0      [the, to, that, is, of, and, you, it, for, in]
1     [the, 25, game, in, team, to, he, and, of, was]
2     [of, god, the, to, that, is, in, and, not, you]
3     [of, is, it, the, to, in, and, that, day, have]
4   [car, the, it, for, in, to, that, cars, engine...
5   [monitor, with, simms, the, video, ram, and, t...
6   [gun, the, of, to, that, is, people, and, in, ...
7   [for, sale, price, or, drive, system, new, shi...
8   [key, the, chip, to, that, is, clipper, it, an...
9   [openwindows, you, or, image, to, the, and, fo...
10  [jews, of, the, to, israel, and, peace, in, is...
11  [the, was, fbi, they, to, that, fire, of, and,...
12  [phd, meteorologist, professor, atmospheric, o...
13  [space, the, nasa, to, shuttle, of, and, in, f...
14  [disk, drive, drives, controller, hard, bios, ...
15  [the, and, were, of, they, in, armenian, was, ...
16    [bike, the, to, you, on, and, of, your, in, if]
17  [you, is, are, gay, to, of, that, moral, the, ...
18                  [ditto, ignore, hi, , , , , , , ]
19  [window, the, application, event, using, event...
```

Extra Question (5 Points)

Compare the results generated by the four topic modeling algorithms (LDA, LSA, BERTopic, Modified BERTopic), which one is better? You should explain the reasons in details.

This question will compensate for any points deducted in this exercise. Maximum marks for the exercise is 100 points.

The modified version of BERTopic surpasses LDA, LSA and standard BERTopic methodology for creating consistent topics from domain-specific text data through transformer-based representations. The modern NLP approach used by BERTopic and its modified version outmatches LDA and LSA since these algorithms depend on statistical analysis and limited semantic processing. Among the listed approaches Modified BERTopic demonstrates the optimal combination of coherent topic detection along with flexible application and physician-friendly interpretability.

The modified version of BERTopic surpasses LDA, LSA and standard BERTopic methodology for creating consistent topics from domain-specific text data through transformer-based representations. The modern NLP approach used by BERTopic and its modified version outmatches LDA and LSA since these algorithms depend on statistical analysis and limited semantic processing. Among the listed approaches Modified BERTopic demonstrates the optimal combination of coherent topic detection along with flexible application and physician-friendly interpretability.

```
# Write your code here
```

Mandatory Question

Important: Reflective Feedback on this exercise

Please provide your thoughts and feedback on the exercises you completed in this assignment.

Consider the following points in your response:

Learning Experience: Describe your overall learning experience in working with text data and extracting features using various topic modeling algorithms. Did you understand these algorithms and did the implementations helped in grasping the nuances of feature extraction from text data.

Challenges Encountered: Were there specific difficulties in completing this exercise?

Relevance to Your Field of Study: How does this exercise relate to the field of NLP?

(Your submission will not be graded if this question is left unanswered)

```
# Your answer here (no code for this question, write down your answer as detail

'''
Please write you answer here:

The Assignment is challenging and time consuming to solve. And there are so mar



'''
```