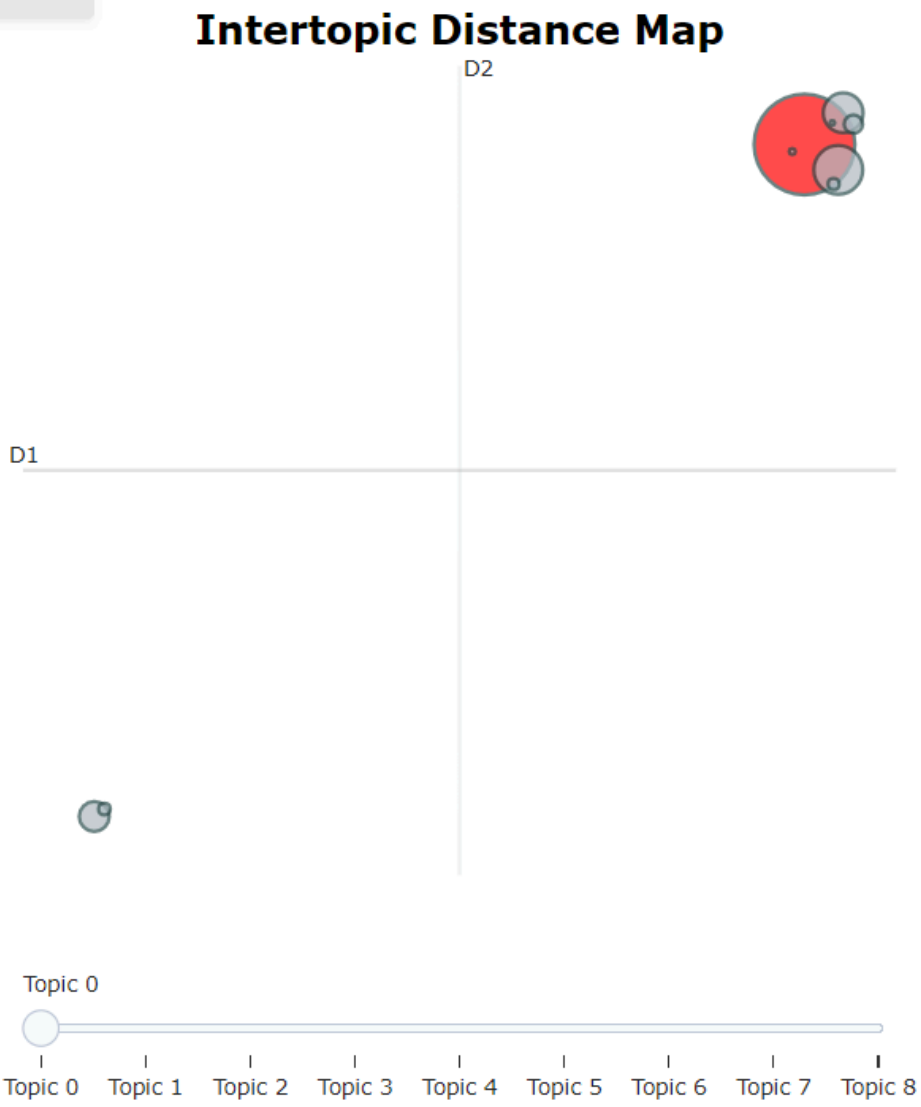
- **Bertopic model with UMAP model for reducing dimensionality and HDBScan model for clustering the embeddings.**

```
umap_model = UMAP(n_neighbors=15, n_components=5, min_dist=0.0,
metric='cosine')
```

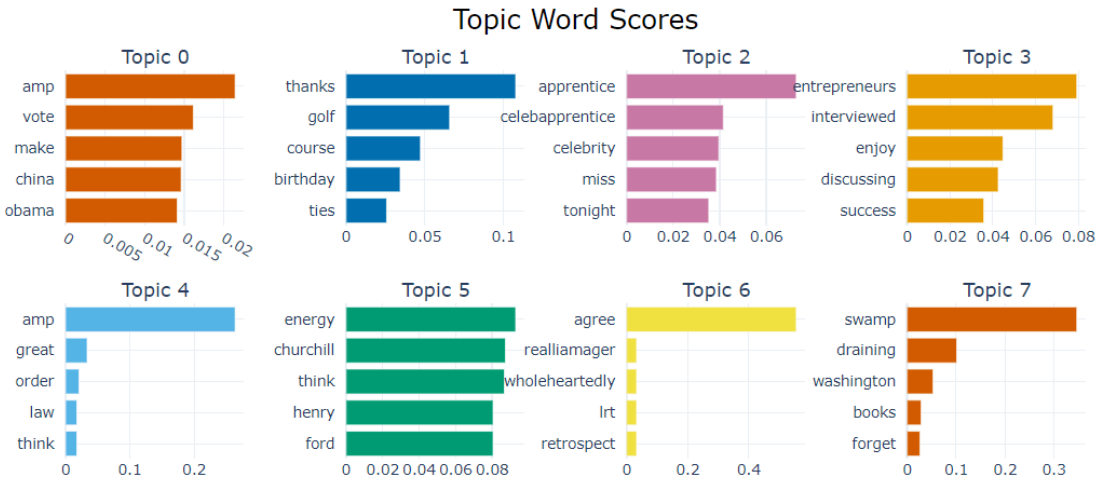
```
hdbscan_model = HDBSCAN(min_cluster_size=15, metric='euclidean',
cluster_selection_method='eom', prediction_data=True)
```

```
topic_model = BERTopic(
                embedding_model=embedding_model,    # Step 1 - Extract embeddings
                  umap_model=umap_model,              # Step 2 - Reduce dimensionality
                  hdbscan_model=hdbscan_model,        # Step 3 - Cluster reduced embeddings
                  vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                  ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                  representation_model=representation_model,
                  nr_topics=10                        # Step 6 - Diversify topic words
)
```

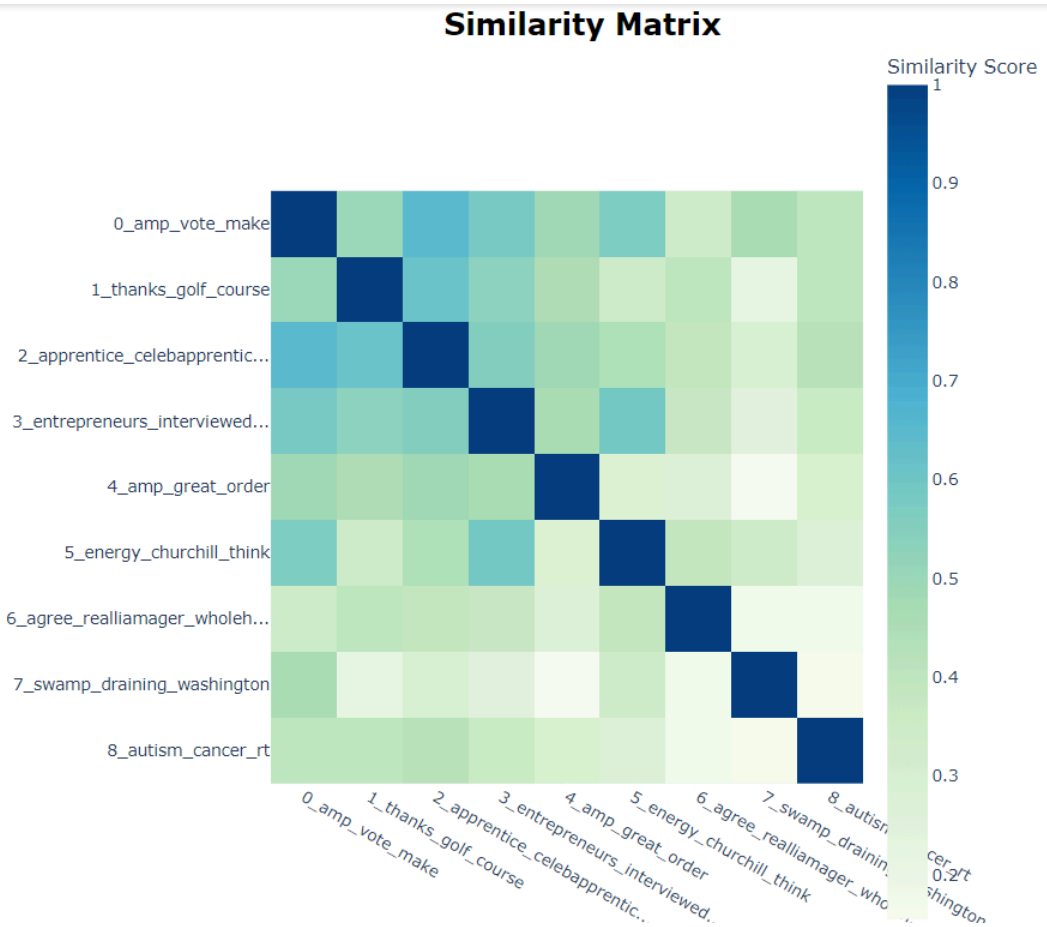
Add text cell



```
topic_model.visualize_barchart()
```



[34]



```
tweets = pd.DataFrame({"Tweets": filtered_text,
                       "ID": range(len(filtered_text)),
                       "Topic": topics})
tweets_per_topic = tweets.groupby(['Topic'], as_index=False).agg({'Tweets': ' '.join})
cleaned_topics = topic_model._preprocess_text(tweets_per_topic.Tweets.values)

# Extract vectorizer and analyzer from BERTopic
vectorizer = topic_model.vectorizer_model
analyzer = vectorizer.build_analyzer()

# Extract features for Topic Coherence evaluation
words = vectorizer.get_feature_names_out()
tokens = [analyzer(doc) for doc in cleaned_topics]
dictionary = corpora.Dictionary(tokens)
corpus = [dictionary.doc2bow(token) for token in tokens]
topic_words = [[words for words, _ in topic_model.get_topic(topic)]
               for topic in range(len(set(topics))-1)]

# Evaluate
coherence_model = CoherenceModel(topics=topic_words,
                                 texts=tokens,
                                 corpus=corpus,
                                 dictionary=dictionary,
                                 coherence='c_v')
coherence = coherence_model.get_coherence()
```

```
print(coherence)
```

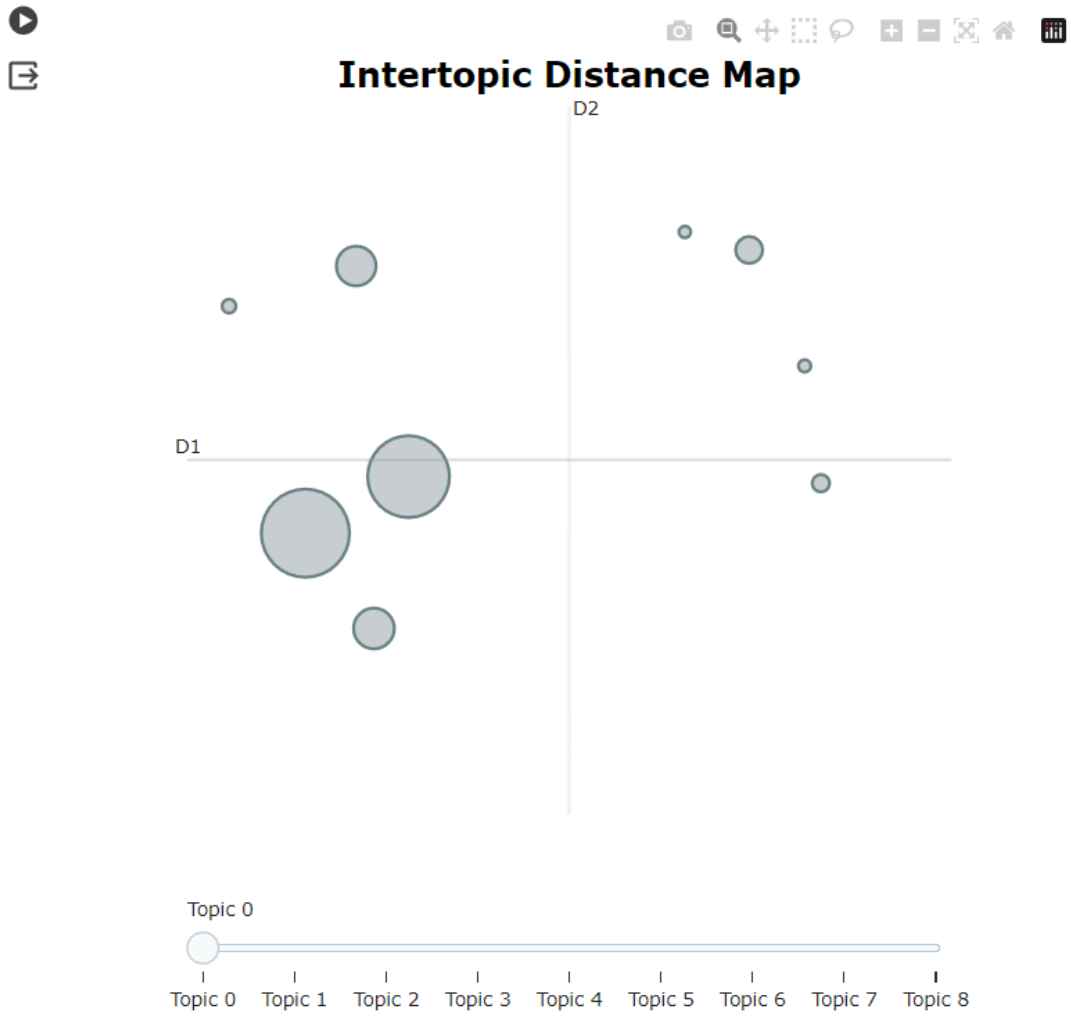
```
0.542773383230092
```

- **Bertopic model with truncated SVD model for reducing dimensionality and HDBScan model for clustering the embeddings.**

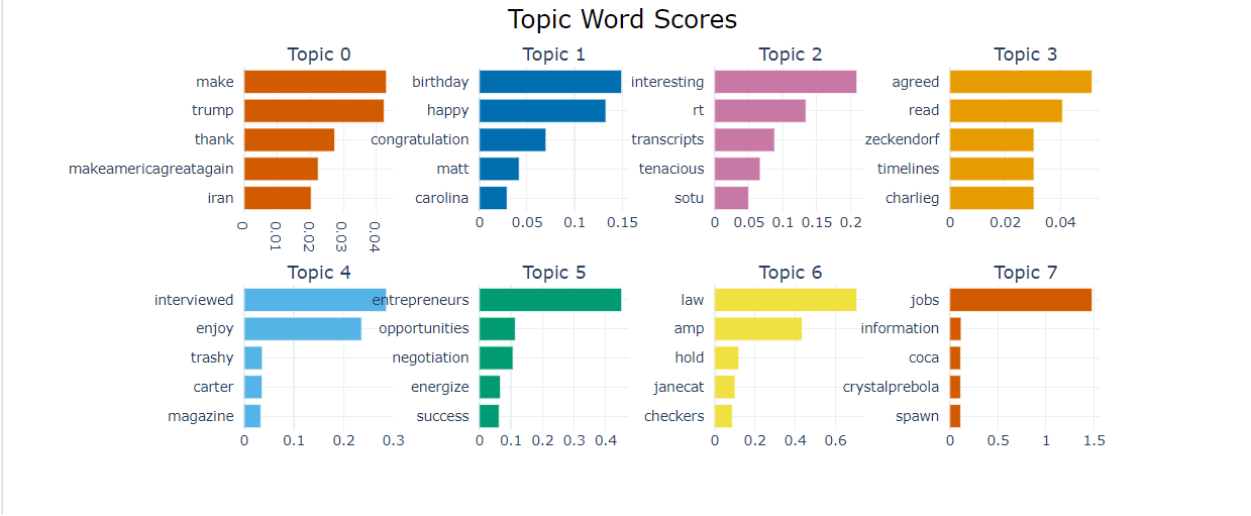
```
from sklearn.decomposition import TruncatedSVD
truncated_svd_model = TruncatedSVD(n_components=5,
algorithm='randomized',n_iter=5, random_state=None, tol=0.0)
```

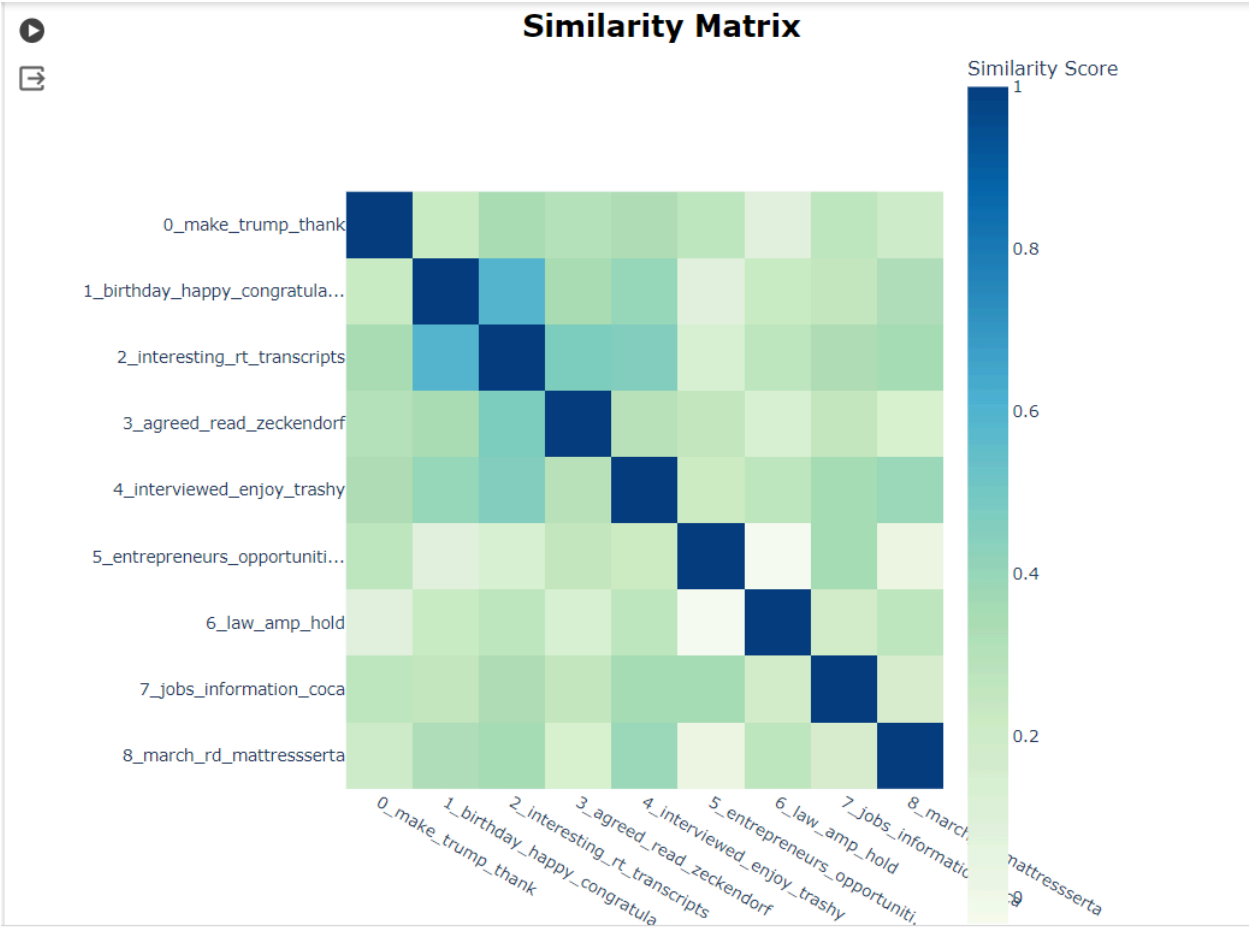
```
hdbscan_model = HDBSCAN(min_cluster_size=15, metric='euclidean',
cluster_selection_method='eom', prediction_data=True)
```

```
topic_model = BERTopic(
                embedding_model=embedding_model,    # Step 1 - Extract embeddings
                  umap_model=truncated_svd_model,              # Step 2 - Reduce dimensionality
                  hdbscan_model=hdbscan_model,        # Step 3 - Cluster reduced embeddings
                  vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                  ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                  representation_model=representation_model,
                  nr_topics=10                        # Step 6 - Diversify topic words
)
```



```
topic_model.visualize_barchart()
```





```
[58] tweets = pd.DataFrame({"Tweets": filtered_text,
                            "ID": range(len(filtered_text)),
                            "Topic": topics})
     tweets_per_topic = tweets.groupby(['Topic'], as_index=False).agg({'Tweets': ' '.join})
     cleaned_topics = topic_model._preprocess_text(tweets_per_topic.Tweets.values)

     # Extract vectorizer and analyzer from BERTopic
     vectorizer = topic_model.vectorizer_model
     analyzer = vectorizer.build_analyzer()

     # Extract features for Topic Coherence evaluation
     words = vectorizer.get_feature_names_out()
     tokens = [analyzer(doc) for doc in cleaned_topics]
     dictionary = corpora.Dictionary(tokens)
     corpus = [dictionary.doc2bow(token) for token in tokens]
     topic_words = [[words for words, _ in topic_model.get_topic(topic)]
                    for topic in range(len(set(topics))-1)]

     # Evaluate
     coherence_model = CoherenceModel(topics=topic_words,
                                      texts=tokens,
                                      corpus=corpus,
                                      dictionary=dictionary,
                                      coherence='c_v')
     coherence = coherence_model.get_coherence()
```

```
print(coherence)
```

```
0.49688017596428274
```

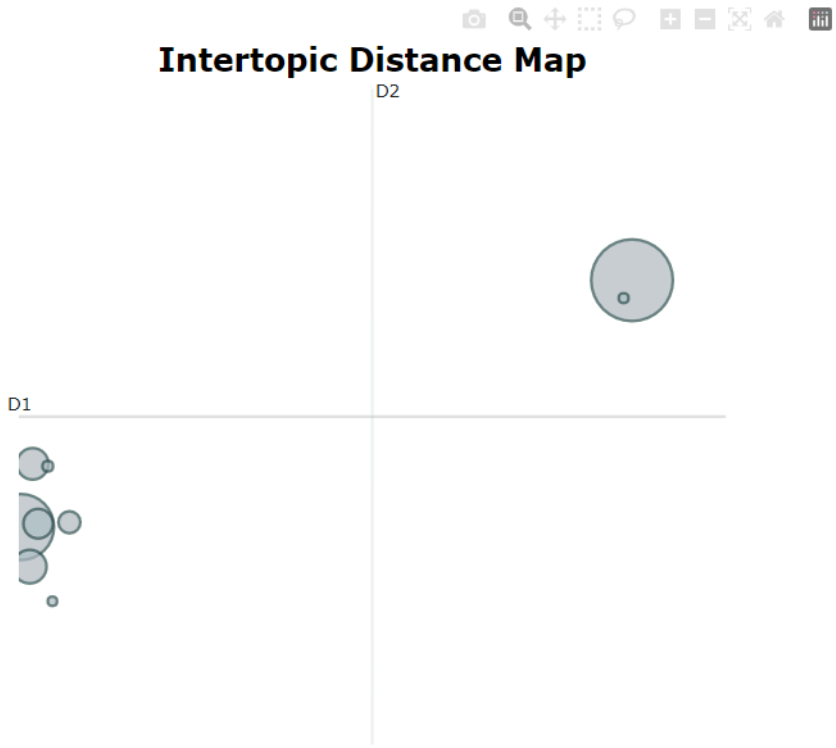
- **Bertopic model with PCA model for reducing dimensionality and HDBScan model for clustering the embeddings.**

```
from sklearn.decomposition import PCA
pca_model = PCA(n_components=5, copy=True, whiten=False,
svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10,
power_iteration_normalizer='auto', random_state=None)
```

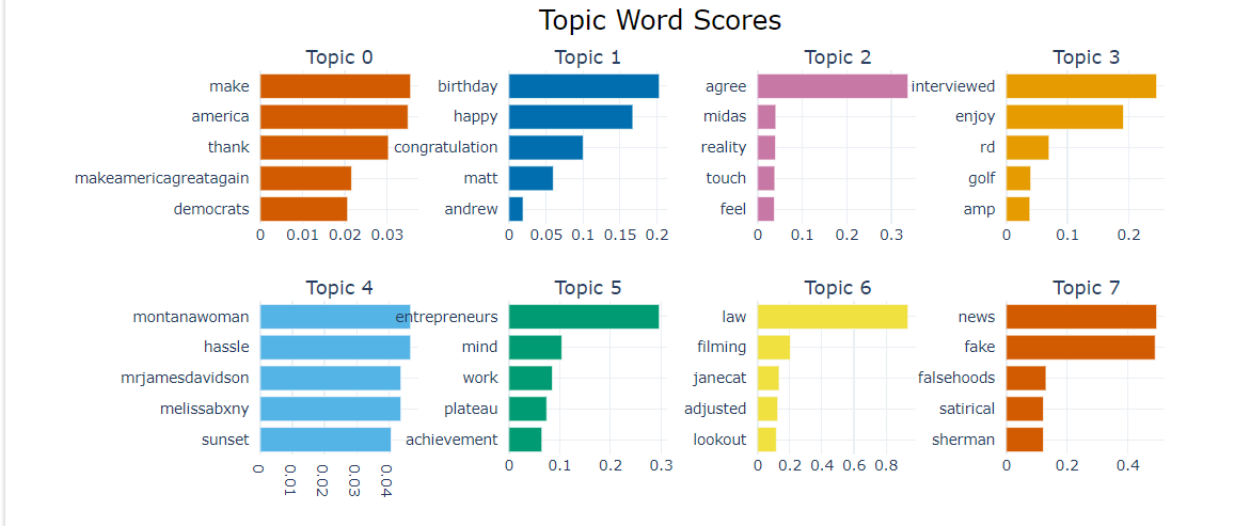
```
hdbscan_model = HDBSCAN(min_cluster_size=15, metric='euclidean',
cluster_selection_method='eom', prediction_data=True)
```

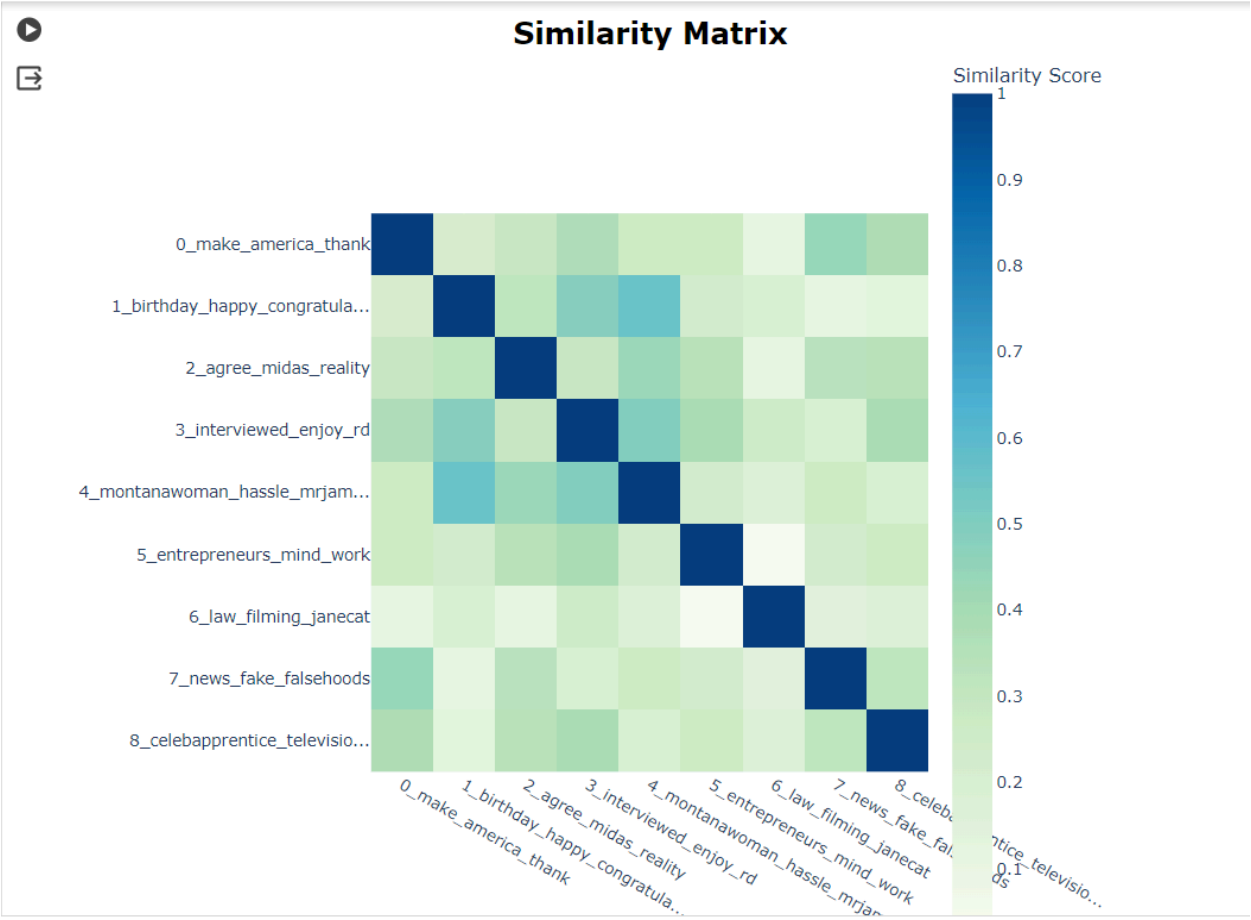
```python
topic_model = BERTopic(
                embedding_model=embedding_model,    # Step 1 - Extract embeddings
                  umap_model=pca_model,              # Step 2 - Reduce dimensionality
                  hdbscan_model=hdbscan_model,        # Step 3 - Cluster reduced embeddings
                  vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                  ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                  representation_model=representation_model,
                  nr_topics=10                        # Step 6 - Diversify topic words
)
```

```python
topic_model.visualize_topics()
```



```
topic_model.visualize_barchart()
```





```
[82] tweets = pd.DataFrame({"Tweets": filtered_text,
                            "ID": range(len(filtered_text)),
                            "Topic": topics})
     tweets_per_topic = tweets.groupby(['Topic'], as_index=False).agg({'Tweets': ' '.join})
     cleaned_topics = topic_model._preprocess_text(tweets_per_topic.Tweets.values)

     # Extract vectorizer and analyzer from BERTopic
     vectorizer = topic_model.vectorizer_model
     analyzer = vectorizer.build_analyzer()

     # Extract features for Topic Coherence evaluation
     words = vectorizer.get_feature_names_out()
     tokens = [analyzer(doc) for doc in cleaned_topics]
     dictionary = corpora.Dictionary(tokens)
     corpus = [dictionary.doc2bow(token) for token in tokens]
     topic_words = [[words for words, _ in topic_model.get_topic(topic)]
                    for topic in range(len(set(topics))-1)]

     # Evaluate
     coherence_model = CoherenceModel(topics=topic_words,
                                      texts=tokens,
                                      corpus=corpus,
                                      dictionary=dictionary,
                                      coherence='c_v')
     coherence = coherence_model.get_coherence()
```

```
print(coherence)
```

```
0.5502179535418598
```

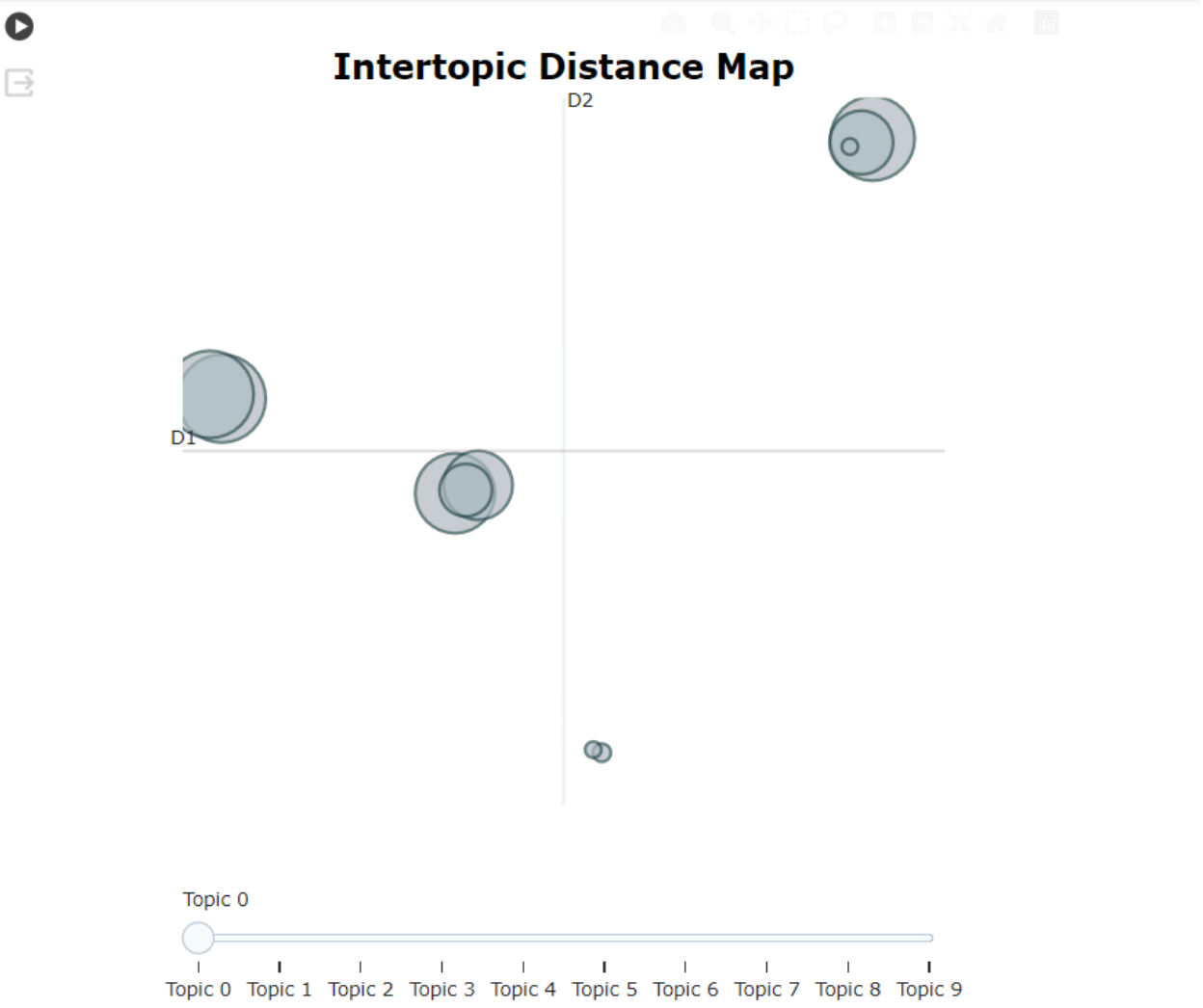
- **Bertopic model with UMAP model for reducing dimensionality and K means model for clustering the embeddings.**

```
umap_model = UMAP(n_neighbors=15, n_components=5, min_dist=0.0,
metric='cosine')
```

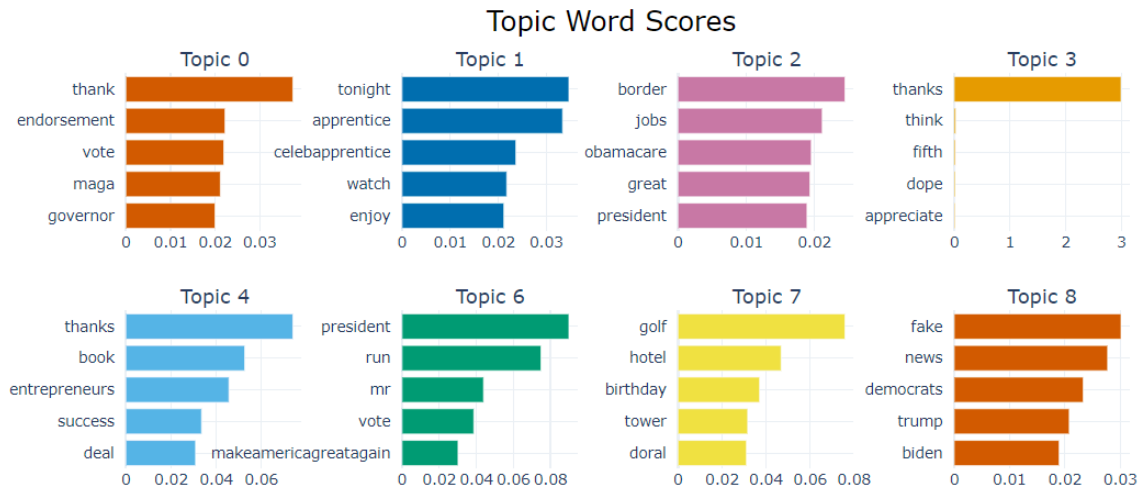
```
from sklearn.cluster import KMeans
```

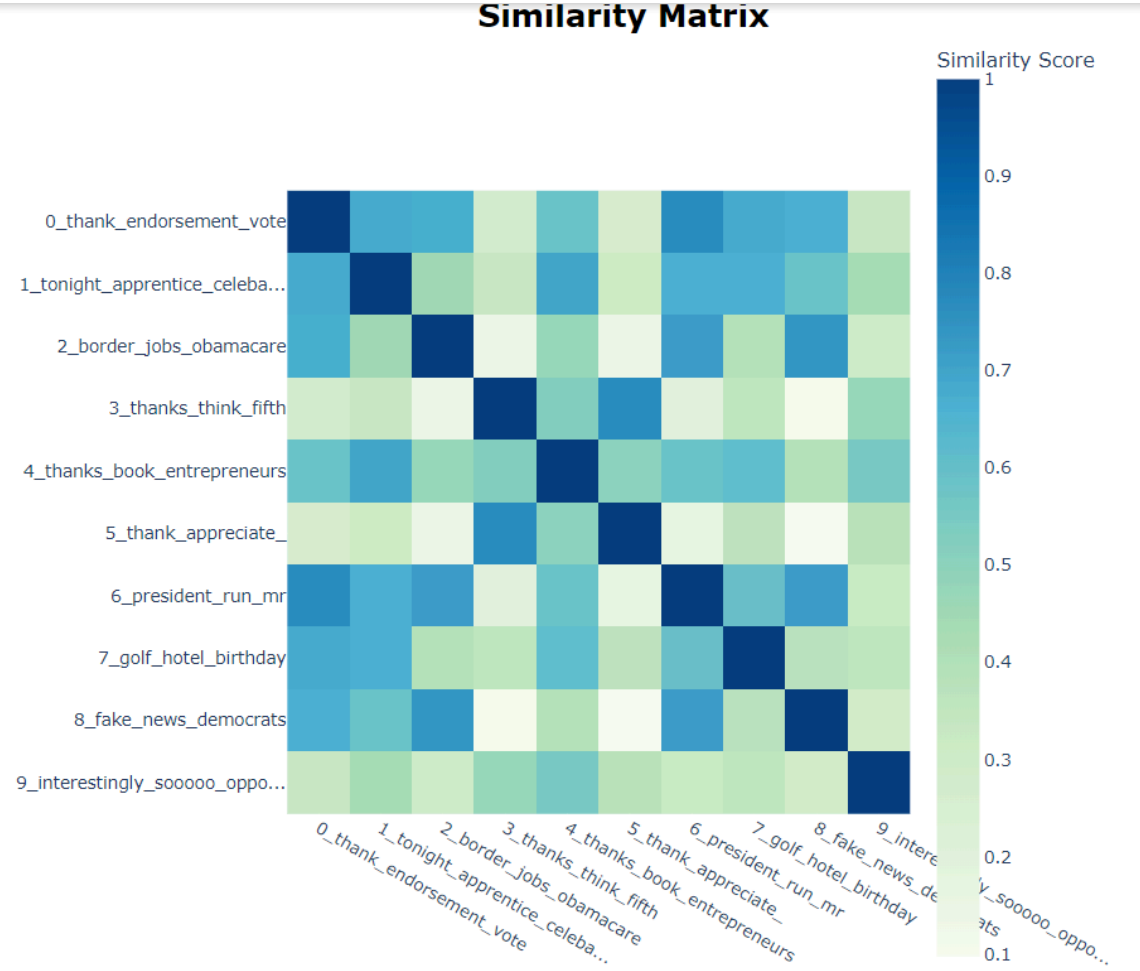
```
kmeans_model = KMeans(n_clusters=10, init='k-means++', n_init=10,
max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True)
```

```
topic_model = BERTopic(
                embedding_model=embedding_model,    # Step 1 - Extract embeddings
                  umap_model=umap_model,              # Step 2 - Reduce dimensionality
                  hdbscan_model=kmeans_model,        # Step 3 - Cluster reduced embeddings
                  vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                  ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                  representation_model=representation_model,
                  nr_topics=10                        # Step 6 - Diversify topic words
)
```



```
topic_model.visualize_barchart()
```





```
[103] tweets = pd.DataFrame({"Tweets": filtered_text,
                             "ID": range(len(filtered_text)),
                             "Topic": topics})
      tweets_per_topic = tweets.groupby(['Topic'], as_index=False).agg({'Tweets': ' '.join})
      cleaned_topics = topic_model._preprocess_text(tweets_per_topic.Tweets.values)

      # Extract vectorizer and analyzer from BERTopic
      vectorizer = topic_model.vectorizer_model
      analyzer = vectorizer.build_analyzer()

      # Extract features for Topic Coherence evaluation
      words = vectorizer.get_feature_names_out()
      tokens = [analyzer(doc) for doc in cleaned_topics]
      dictionary = corpora.Dictionary(tokens)
      corpus = [dictionary.doc2bow(token) for token in tokens]
      topic_words = [[words for words, _ in topic_model.get_topic(topic)]
                     for topic in range(len(set(topics))-1)]

      # Evaluate
      coherence_model = CoherenceModel(topics=topic_words,
                                       texts=tokens,
                                       corpus=corpus,
                                       dictionary=dictionary,
                                       coherence='c_v')
      coherence = coherence_model.get_coherence()
```

```
print(coherence)
```

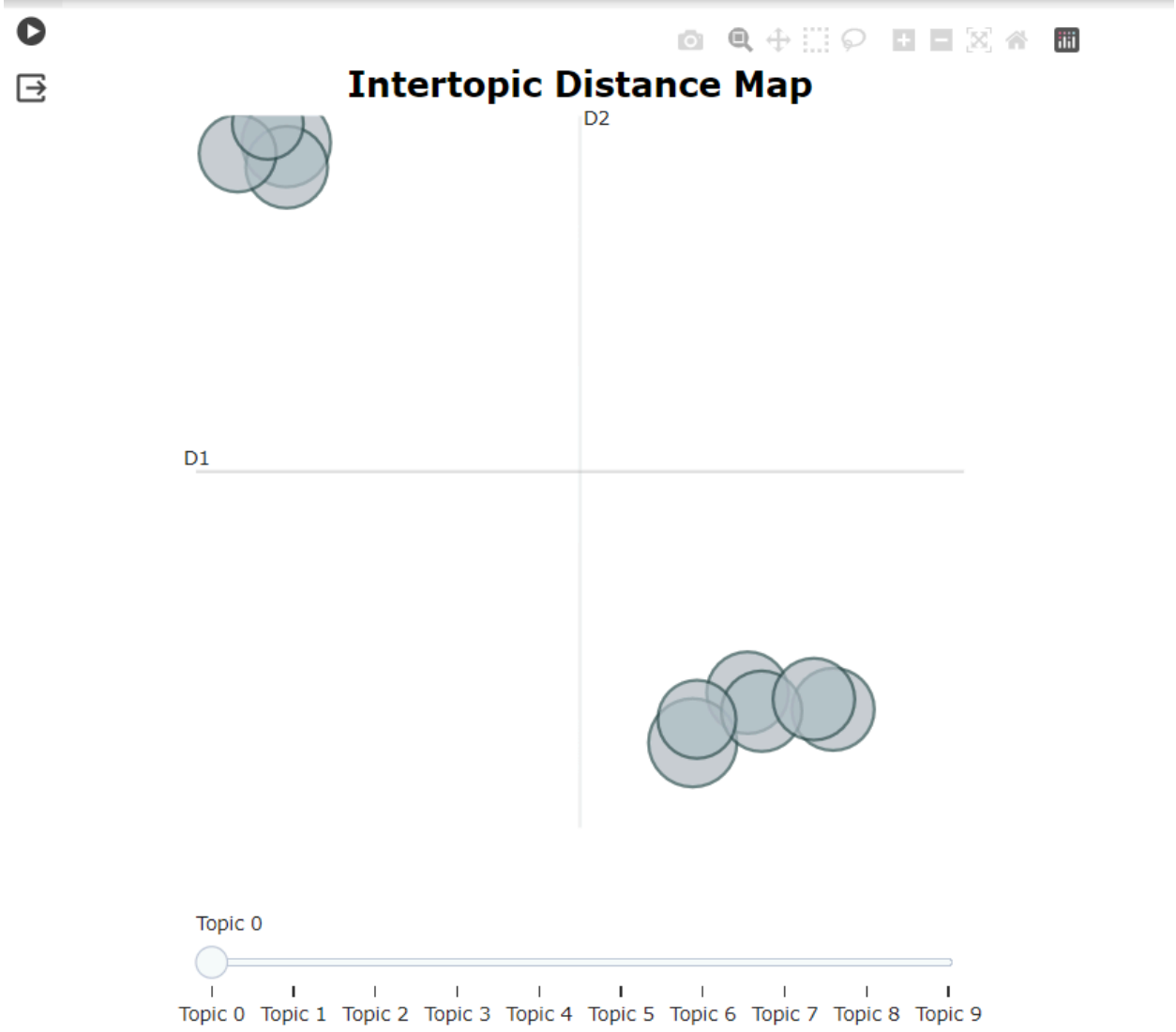
```
0.6364225489698937
```

- **Bertopic model with Truncated SVD model for reducing dimensionality and K means model for clustering the embeddings.**

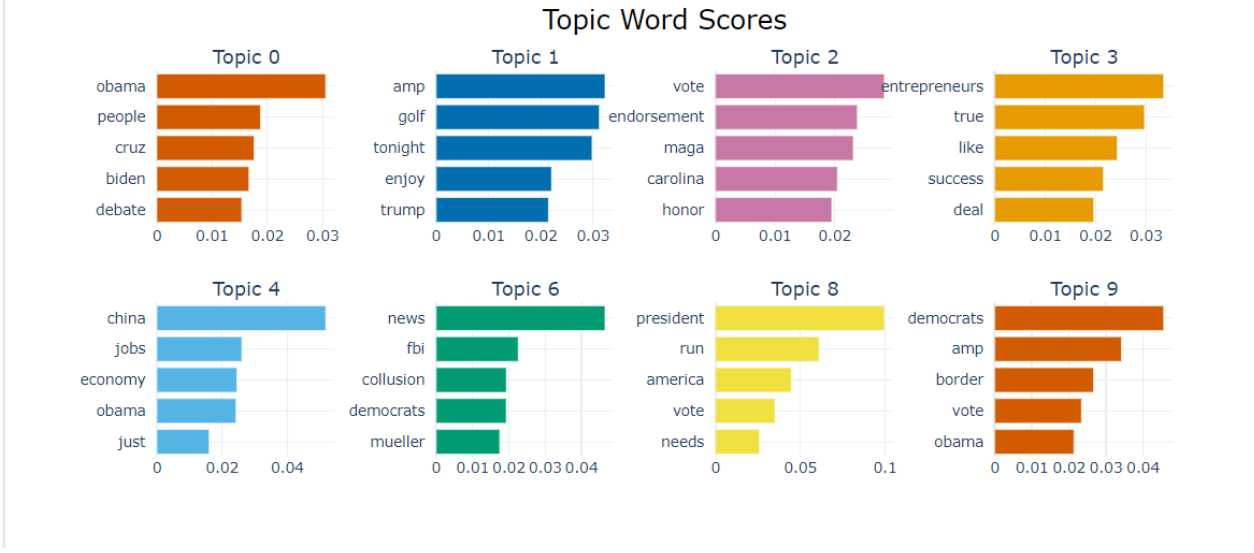
```
from sklearn.decomposition import TruncatedSVD
```

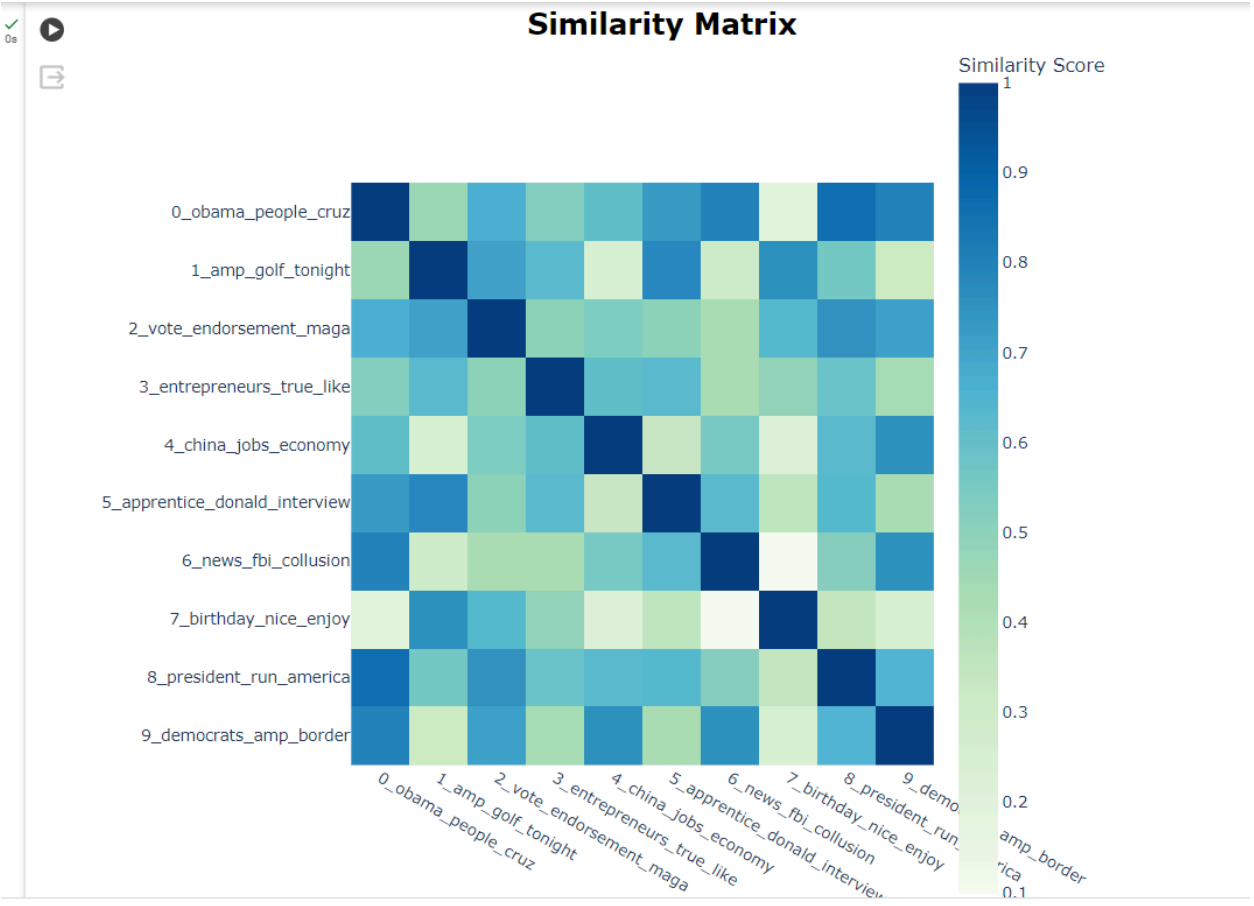
```
truncated_svd_model = TruncatedSVD(n_components=5,
algorithm='randomized',n_iter=5, random_state=None, tol=0.0)
```

```
[164] topic_model = BERTopic(
                      embedding_model=embedding_model,    # Step 1 - Extract embeddings
                        umap_model=truncated_svd_model,             # Step 2 - Reduce dimensionality
                        hdbscan_model=kmeans_model,        # Step 3 - Cluster reduced embeddings
                        vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                        ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                        representation_model=representation_model,
                        nr_topics=10                        # Step 6 - Diversify topic words
    )
```



```
topic_model.visualize_barchart()
```





```
tweets = pd.DataFrame({"Tweets": filtered_text,
                       "ID": range(len(filtered_text)),
                       "Topic": topics})
tweets_per_topic = tweets.groupby(['Topic'], as_index=False).agg({'Tweets': ' '.join})
cleaned_topics = topic_model._preprocess_text(tweets_per_topic.Tweets.values)

# Extract vectorizer and analyzer from BERTopic
vectorizer = topic_model.vectorizer_model
analyzer = vectorizer.build_analyzer()

# Extract features for Topic Coherence evaluation
words = vectorizer.get_feature_names_out()
tokens = [analyzer(doc) for doc in cleaned_topics]
dictionary = corpora.Dictionary(tokens)
corpus = [dictionary.doc2bow(token) for token in tokens]
topic_words = [[words for words, _ in topic_model.get_topic(topic)]
               for topic in range(len(set(topics))-1)]

# Evaluate
coherence_model = CoherenceModel(topics=topic_words,
                                 texts=tokens,
                                 corpus=corpus,
                                 dictionary=dictionary,
                                 coherence='c_v')
coherence = coherence_model.get_coherence()
```

```
print(coherence)
```

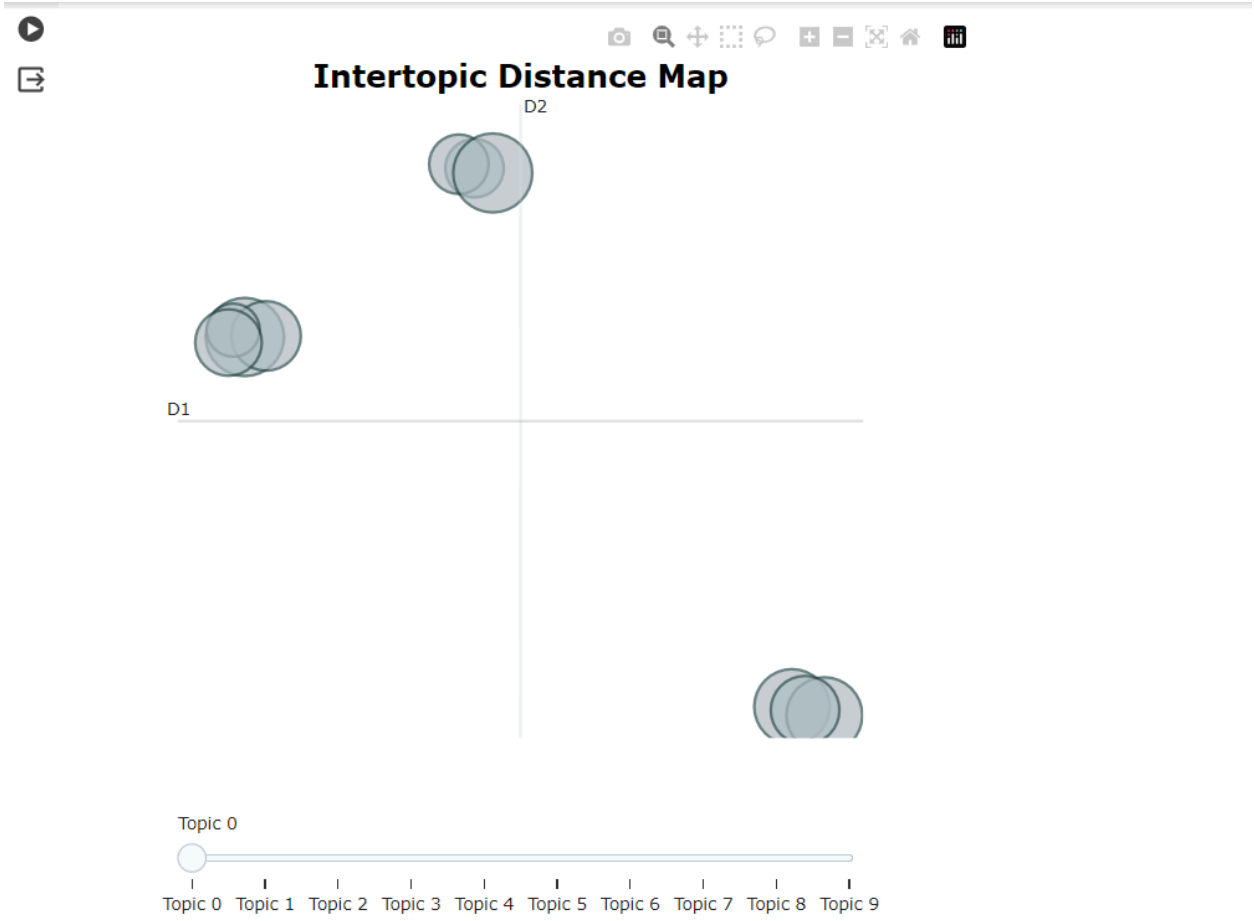
```
0.581137607634149
```

- **Bertopic model with PCA model for reducing dimensionality and K means model for clustering the embeddings.**

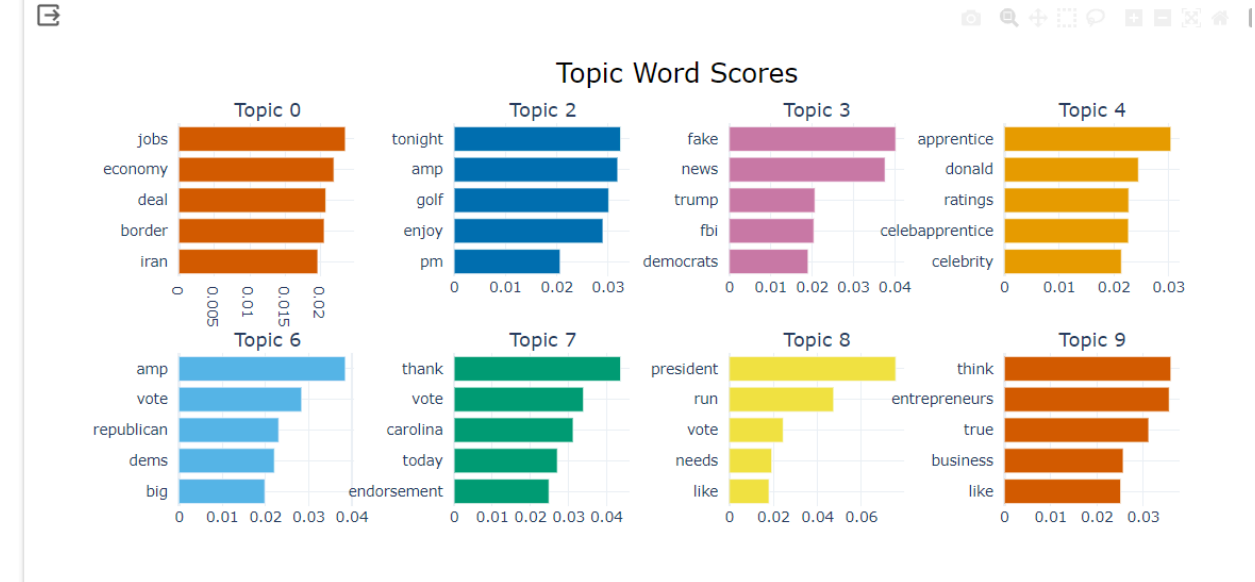
```
pca_model = PCA(n_components=5, copy=True, whiten=False,
svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10,
power_iteration_normalizer='auto', random_state=None)
```

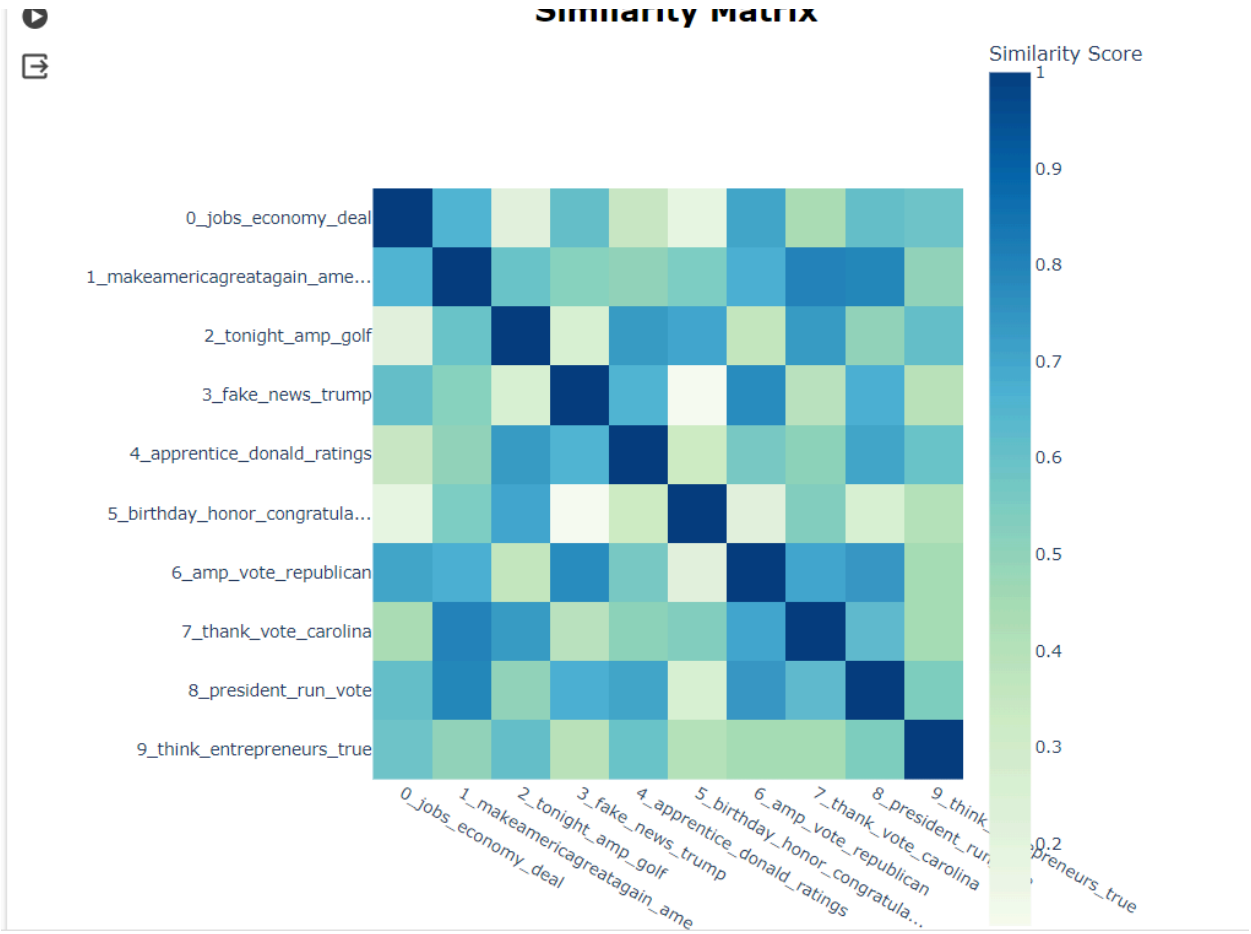
```
kmeans_model = KMeans(n_clusters=10, init='k-means++', n_init=10,
max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True)
```

```
topic_model = BERTopic(
                embedding_model=embedding_model,    # Step 1 - Extract embeddings
                  umap_model=pca_model,             # Step 2 - Reduce dimensionality
                  hdbscan_model=kmeans_model,       # Step 3 - Cluster reduced embeddings
                  vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                  ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                  representation_model=representation_model,
                  nr_topics=10                        # Step 6 - Diversify topic words
)
```




```
topic_model.visualize_barchart()
```




Similarity Matrix
Similarity Score
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0_jobs_economy_deal
1_makeamericagreatagain_ame...
2_tonight_amp_golf
3_fake_news_trump
4_apprentice_donald_ratings
5_birthday_honor_congratula...
6_amp_vote_republican
7_thank_vote_carolina
8_president_run_vote
9_think_entrepreneurs_true

```
tweets = pd.DataFrame({"Tweets": filtered_text,
                       "ID": range(len(filtered_text)),
                       "Topic": topics})
tweets_per_topic = tweets.groupby(['Topic'], as_index=False).agg({'Tweets': ' '.join})
cleaned_topics = topic_model._preprocess_text(tweets_per_topic.Tweets.values)

# Extract vectorizer and analyzer from BERTopic
vectorizer = topic_model.vectorizer_model
analyzer = vectorizer.build_analyzer()

# Extract features for Topic Coherence evaluation
words = vectorizer.get_feature_names_out()
tokens = [analyzer(doc) for doc in cleaned_topics]
dictionary = corpora.Dictionary(tokens)
corpus = [dictionary.doc2bow(token) for token in tokens]
topic_words = [[words for words, _ in topic_model.get_topic(topic)]
               for topic in range(len(set(topics))-1)]

# Evaluate
coherence_model = CoherenceModel(topics=topic_words,
                                 texts=tokens,
                                 corpus=corpus,
                                 dictionary=dictionary,
                                 coherence='c_v')
coherence = coherence_model.get_coherence()
```

```
print(coherence)
```

```
0.5822388061619714
```

- **Bertopic model with UMAP model for reducing dimensionality and BIRCH model for clustering the embeddings.**

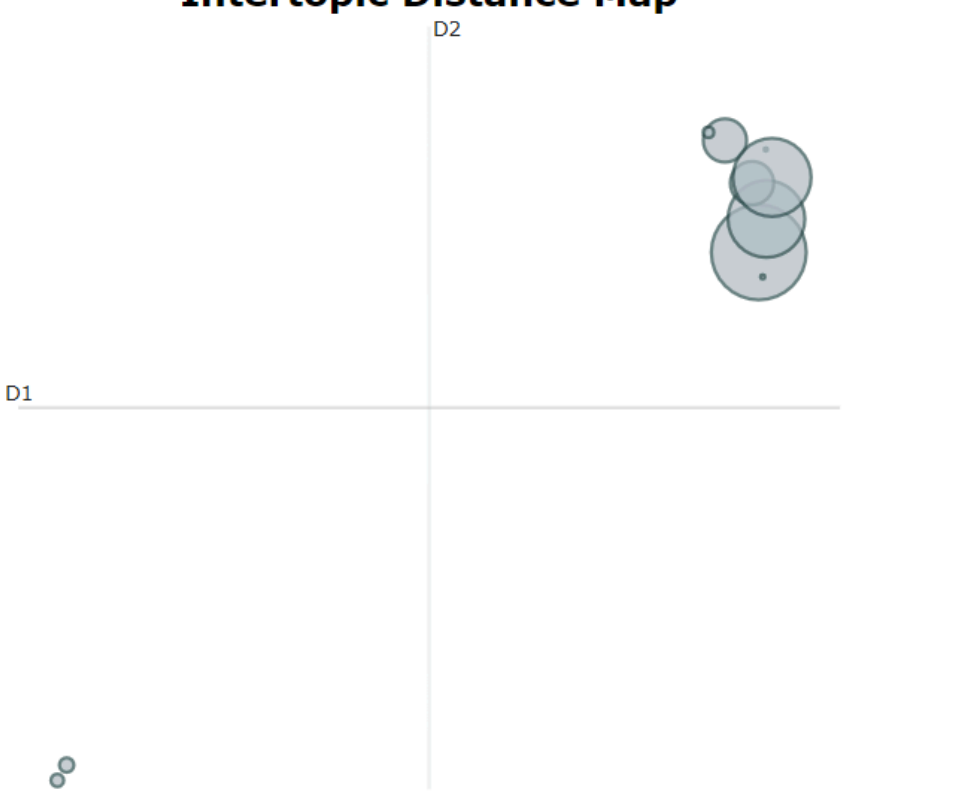
```
umap_model = UMAP(n_neighbors=15, n_components=5, min_dist=0.0,
metric='cosine')
```

```
from sklearn.cluster import Birch
```

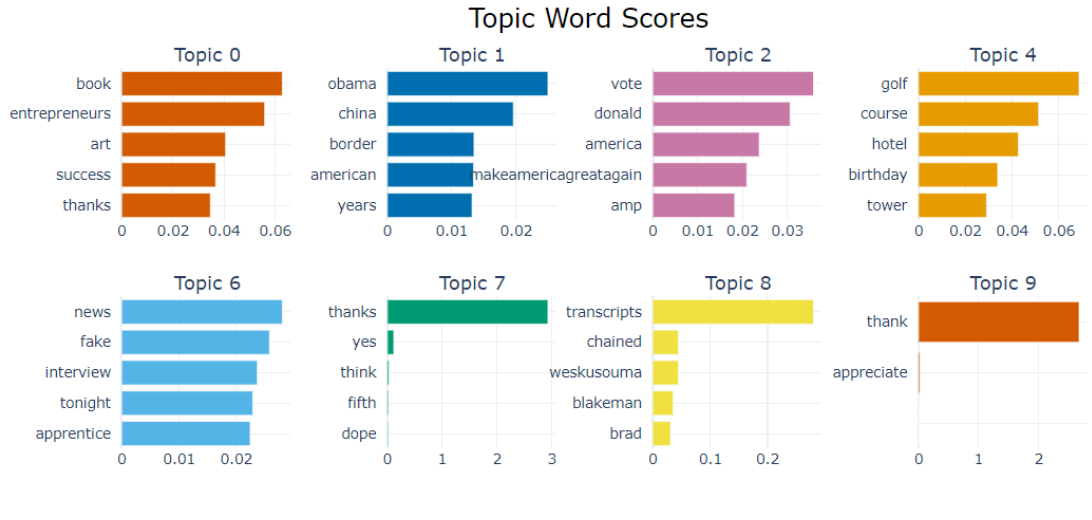
```
birch_model = Birch( threshold=0.5, branching_factor=50, n_clusters=10,
compute_labels=True, copy=True)
```

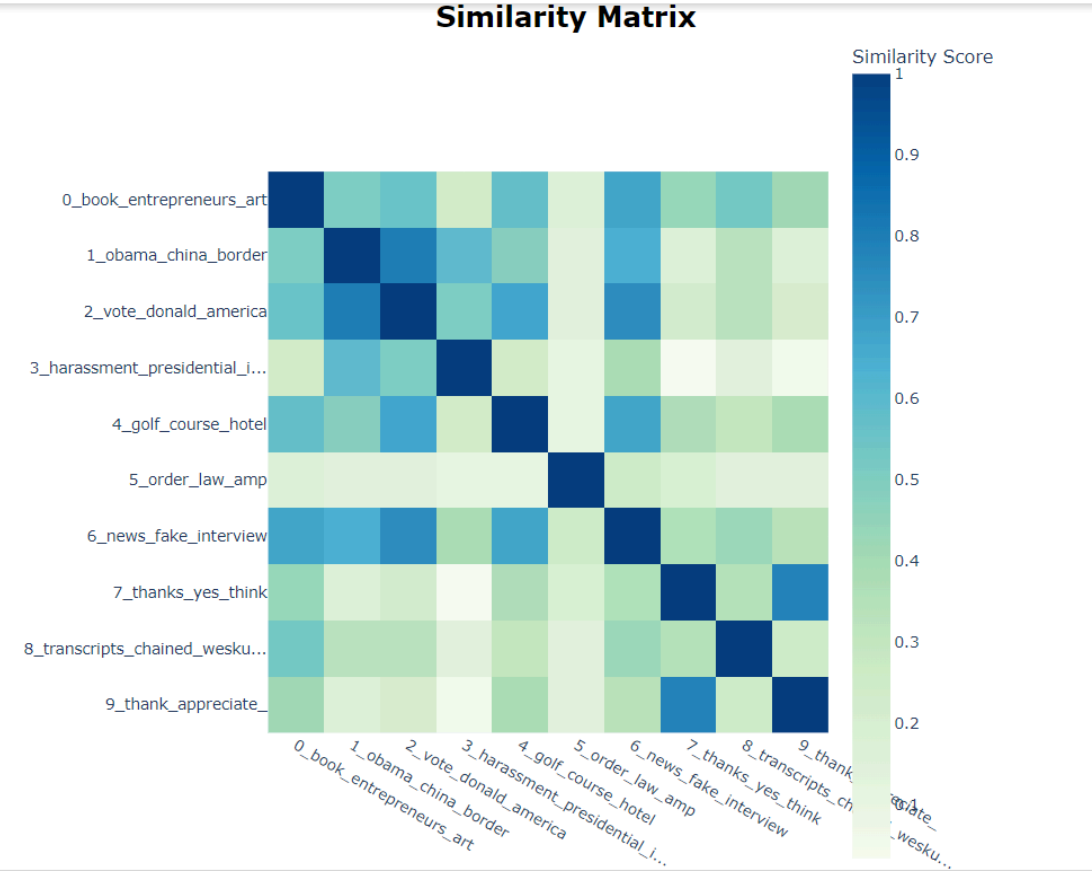
```
topic_model = BERTopic(
                embedding_model=embedding_model,    # Step 1 - Extract embeddings
                  umap_model=umap_model,              # Step 2 - Reduce dimensionality
                  hdbscan_model=birch_model,        # Step 3 - Cluster reduced embeddings
                  vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                  ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                  representation_model=representation_model,
                  nr_topics=10                        # Step 6 - Diversify topic words
)
```

```
topic_model.visualize_topics()
```



```
topic_model.visualize_barchart()
```





```
[126] tweets = pd.DataFrame({"Tweets": filtered_text,
                             "ID": range(len(filtered_text)),
                             "Topic": topics})
      tweets_per_topic = tweets.groupby(['Topic'], as_index=False).agg({'Tweets': ' '.join})
      cleaned_topics = topic_model._preprocess_text(tweets_per_topic.Tweets.values)

      # Extract vectorizer and analyzer from BERTopic
      vectorizer = topic_model.vectorizer_model
      analyzer = vectorizer.build_analyzer()

      # Extract features for Topic Coherence evaluation
      words = vectorizer.get_feature_names_out()
      tokens = [analyzer(doc) for doc in cleaned_topics]
      dictionary = corpora.Dictionary(tokens)
      corpus = [dictionary.doc2bow(token) for token in tokens]
      topic_words = [[words for words, _ in topic_model.get_topic(topic)]
                     for topic in range(len(set(topics))-1)]

      # Evaluate
      coherence_model = CoherenceModel(topics=topic_words,
                                       texts=tokens,
                                       corpus=corpus,
                                       dictionary=dictionary,
                                       coherence='c_v')
      coherence = coherence_model.get_coherence()
```

```
print(coherence)
```

```
0.5203635700205597
```

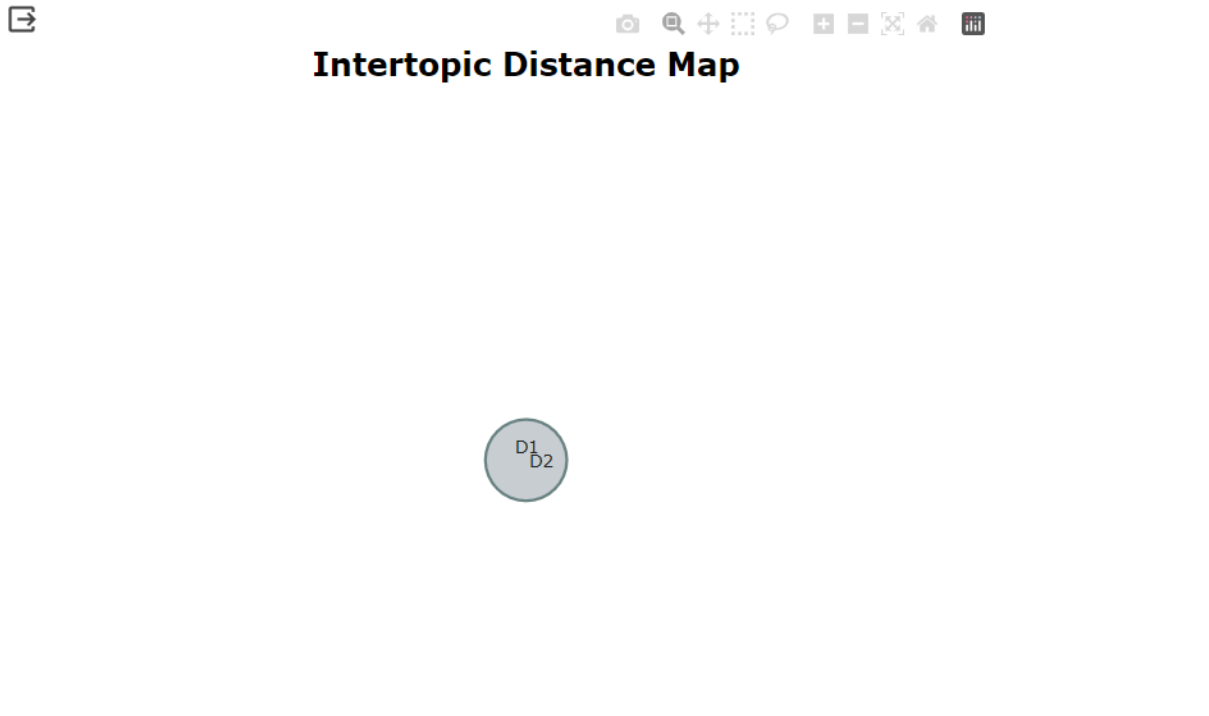
- **Bertopic model with PCA model for reducing dimensionality and BIRCH model for clustering the embeddings.**

```
pca_model = PCA(n_components=5, copy=True, whiten=False,
svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10,
power_iteration_normalizer='auto', random_state=None)
```

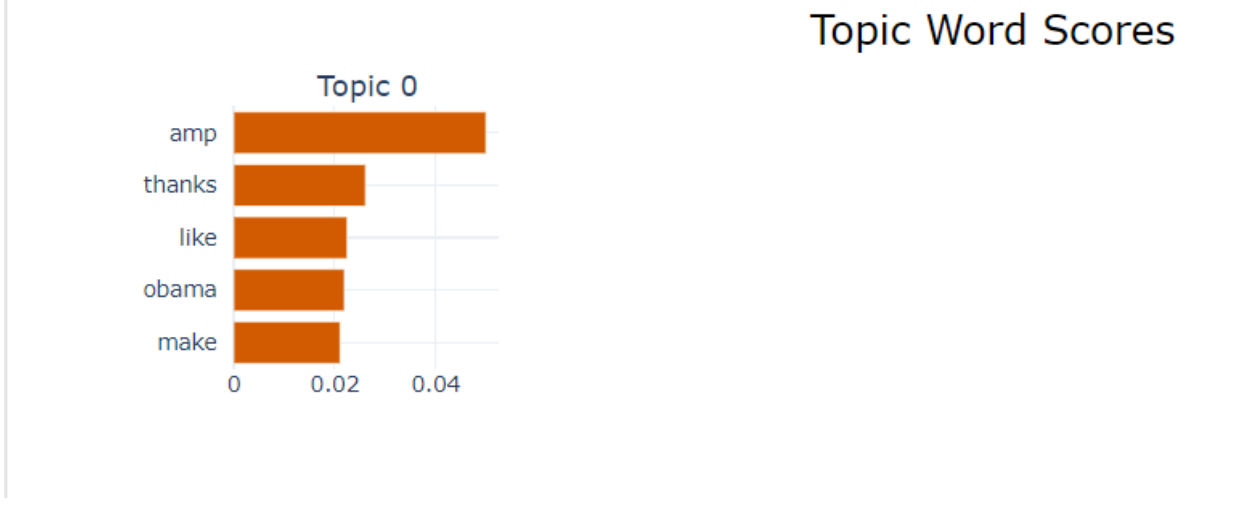
```
birch_model = Birch( threshold=0.5, branching_factor=50, n_clusters=10,
compute_labels=True, copy=True)
```

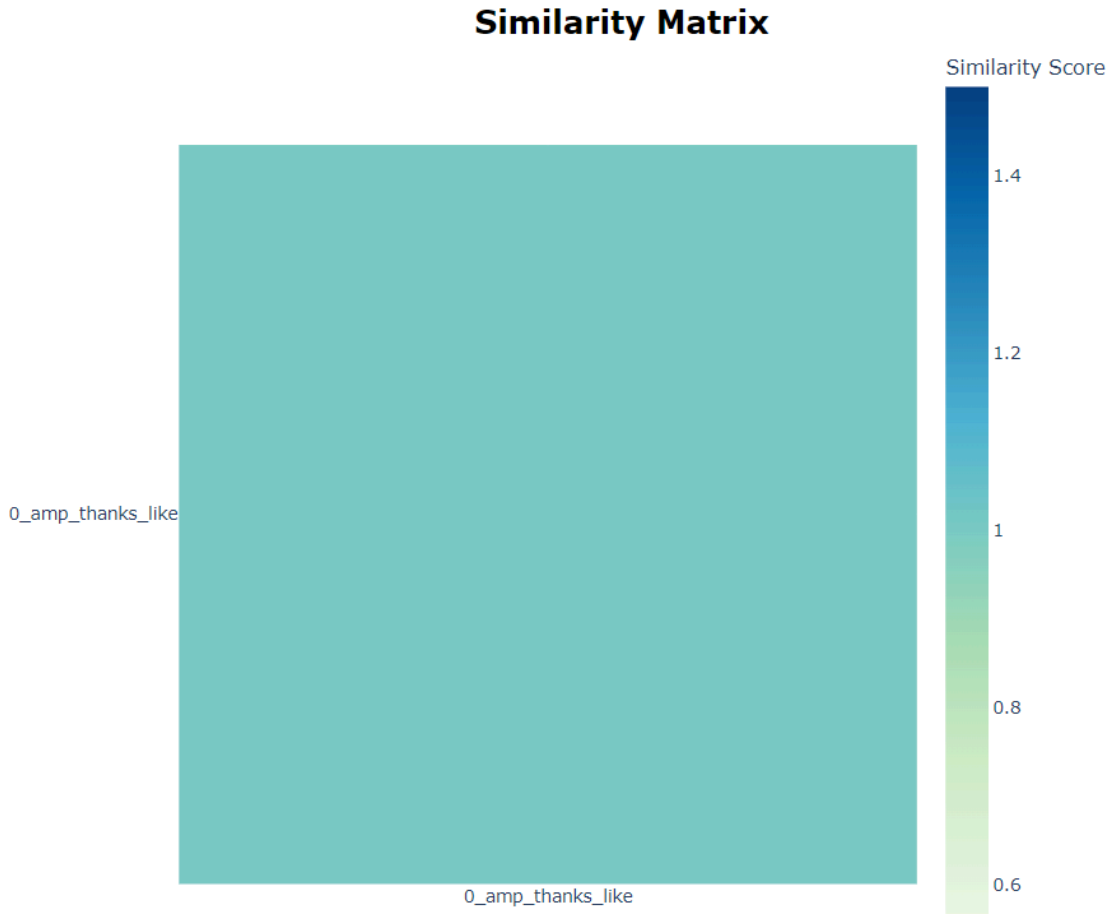
```
[103] topic_model = BERTopic(
                      embedding_model=embedding_model,    # Step 1 - Extract embeddings
                        umap_model=pca_model,              # Step 2 - Reduce dimensionality
                        hdbscan_model=birch_model,        # Step 3 - Cluster reduced embeddings
                        vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                        ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                        representation_model=representation_model,
                        nr_topics=10                        # Step 6 - Diversify topic words
    )
```

```
topic_model.visualize_topics()
```



```
topic_model.visualize_barchart()
```





Only one topic.

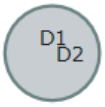
- **Bertopic model with Truncated SVD model for reducing dimensionality and BIRCH model for clustering the embeddings.**

```
from sklearn.decomposition import TruncatedSVD
```

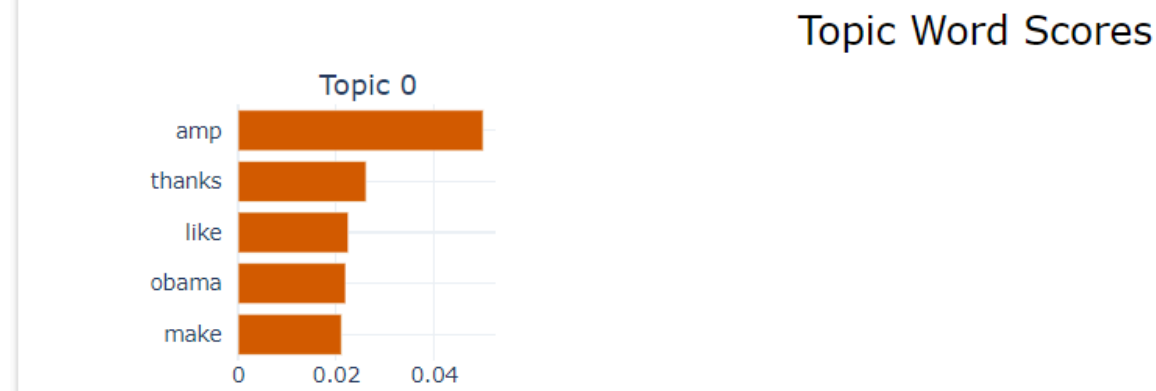
```
truncated_svd_model = TruncatedSVD(n_components=5,
algorithm='randomized',n_iter=5, random_state=None, tol=0.0)
```

```
topic_model = BERTopic(
                embedding_model=embedding_model,    # Step 1 - Extract embeddings
                  umap_model=truncated_svd_model,              # Step 2 - Reduce dimensionality
                  hdbscan_model=birch_model,        # Step 3 - Cluster reduced embeddings
                  vectorizer_model=vectorizer_model,  # Step 4 - Tokenize topics
                  ctfidf_model=ctfidf_model,          # Step 5 - Extract topic words
                  representation_model=representation_model,
                  nr_topics=10                        # Step 6 - Diversify topic words
)
```

```
topic_model.visualize_topics()
```



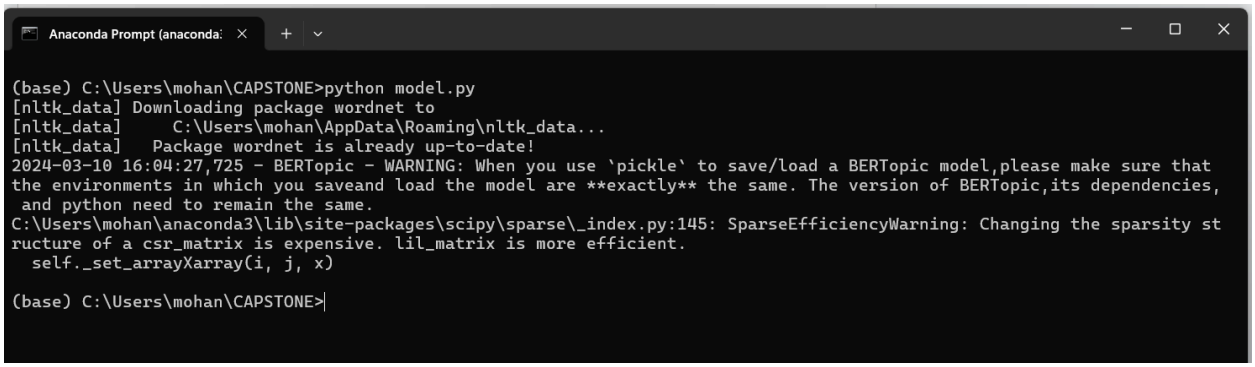
```
topic_model.visualize_barchart()
```



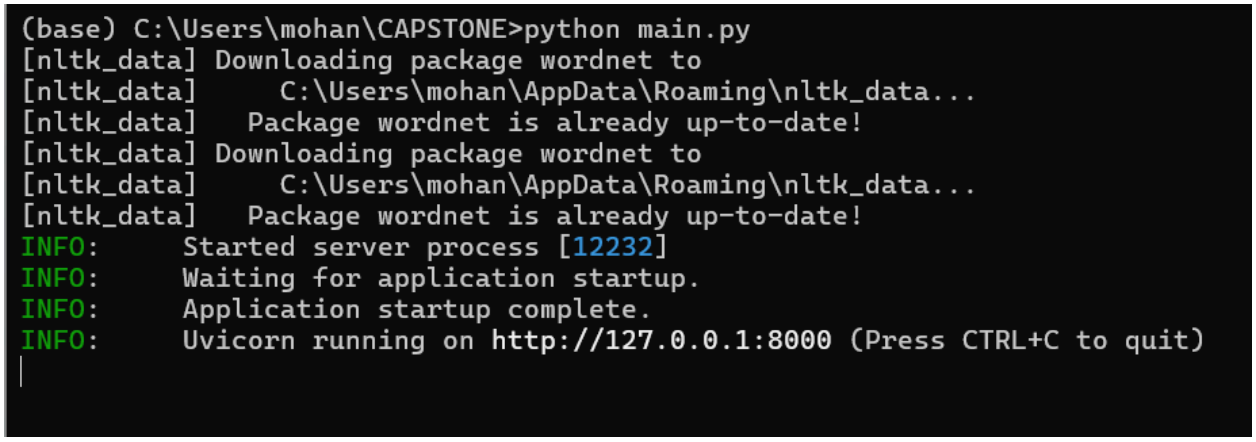
Only one topic.

Proceeding with Model offering high coherence:
model.py → BERTOPIC model
main.py → API service which uses the BERTOPIC model
frontEnd.py → Gets the input from the user and consumes the API to return topic information.

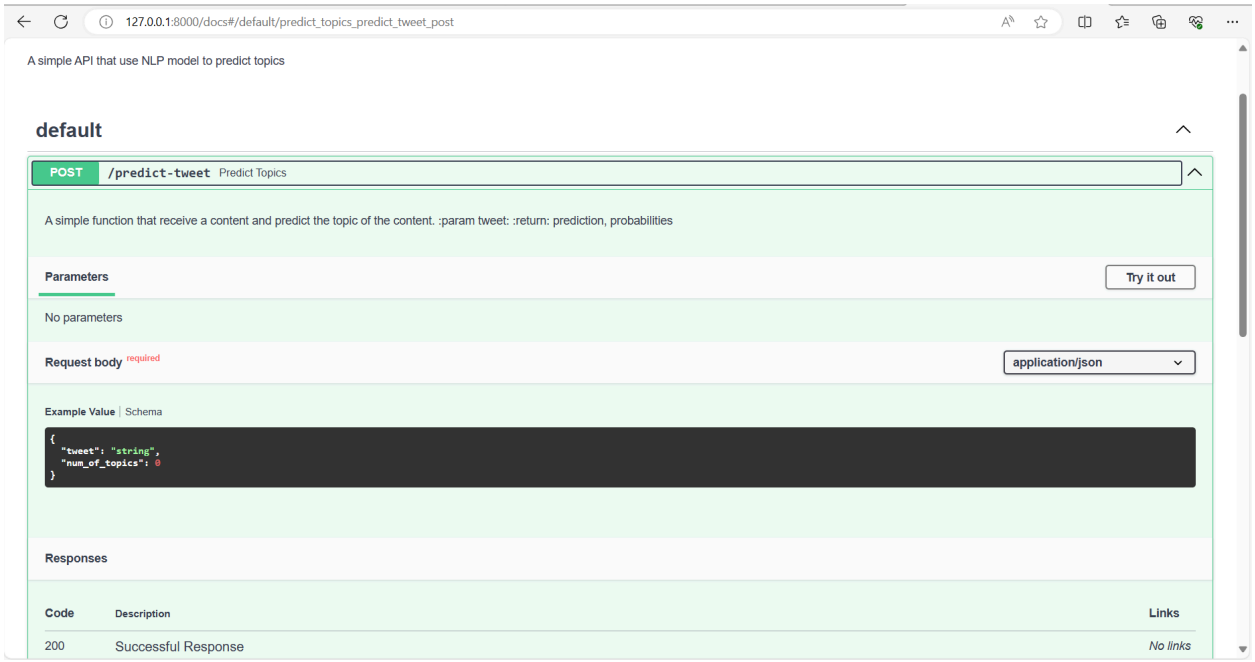
model.py:

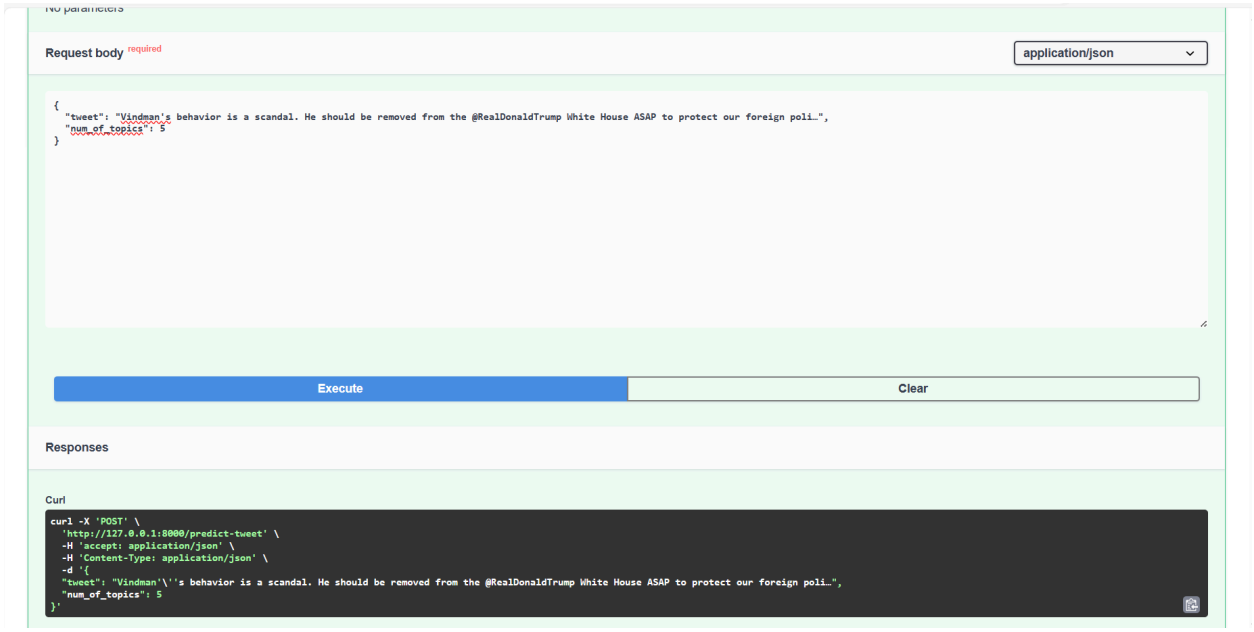


main.py:



API Swagger o/p:


127.0.0.1:8000/docs#/default/predict_topics_predict_tweet_post
A simple API that use NLP model to predict topics
default
POST
/predict-tweet
Predict Topics
A simple function that receive a content and predict the topic of the content. :param tweet: :return: prediction, probabilities
Parameters
Try it out
No parameters
Request body
required
application/json
Example Value
Schema
{
"tweet": "string",
"num_of_topics": 0
}
Responses
Code
Description
Links
200
Successful Response
No links


Request body
required
application/json
{
"tweet": "Vindman's behavior is a scandal. He should be removed from the @RealDonaldTrump White House ASAP to protect our foreign poli…",
"num_of_topics": 5
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/predict-tweet' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"tweet": "Vindman'\''s behavior is a scandal. He should be removed from the @RealDonaldTrump White House ASAP to protect our foreign poli…",
"num_of_topics": 5
}'

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict-tweet' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "tweet": "Vindman\''s behavior is a scandal. He should be removed from the @RealDonaldTrump White House ASAP to protect our foreign poli…",
  "num_of_topics": 5
}'
```

Request URL

```
http://127.0.0.1:8000/predict-tweet
```

Server response

Code Details

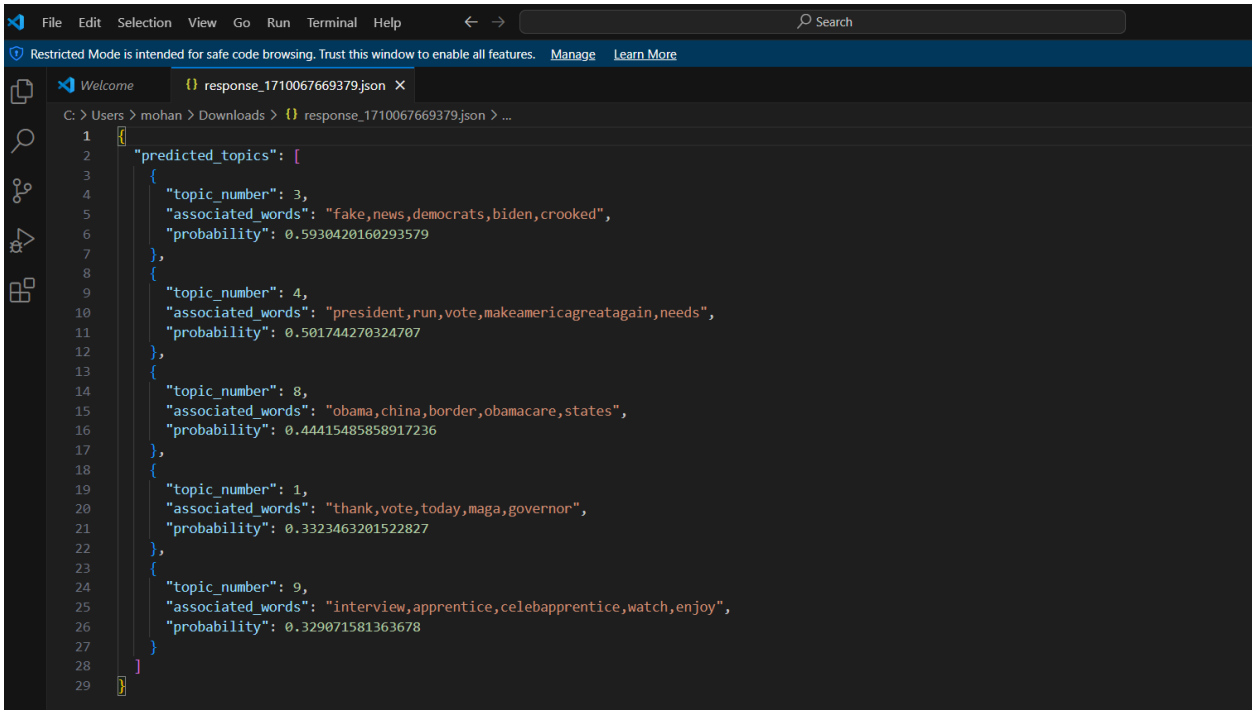
200

Response body

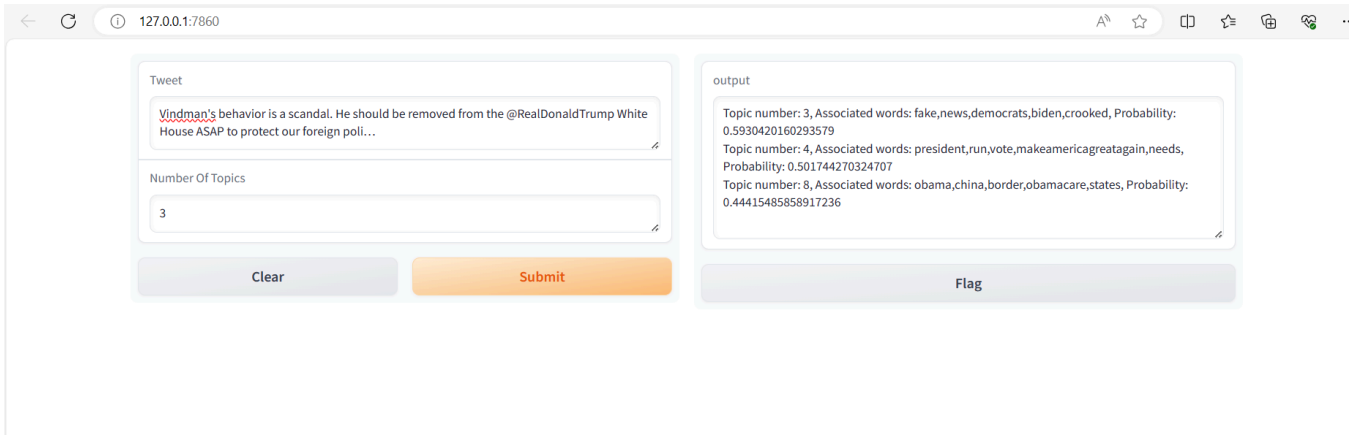

```
{
  "predicted_topics": [
    {
      "topic_number": 3,
      "associated_words": "fake,news,democrats,biden,crooked",
      "probability": 0.5930420160293579
    },
    {
      "topic_number": 4,
      "associated_words": "president,run,vote,makeamericagreatagain,needs",
      "probability": 0.50174427032470 7
    },
    {
      "topic_number": 8,
      "associated_words": "obama,china,border,obamacare,states",
      "probability": 0.44415485858917236
    },
    {
      "topic_number": 1,
      "associated_words": "thank,vote,today,maga,governor",
      "probability": 0.3323463201522827
    },
    {
      "topic_number": 9,
      "associated_words": "interview,apprentice,celebapprentice,watch,enjoy",
      "probability": 0.329071581363678
    }
```


Download

```
(base) C:\Users\mohan\CAPSTONE>python main.py
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\mohan\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\mohan\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
INFO:     Started server process [12232]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:51837 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:51843 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:51843 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:51872 - "POST /predict-tweet HTTP/1.1" 200 OK
```


File Edit Selection View Go Run Terminal Help
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Welcome
response_1710067669379.json
C: > Users > mohan > Downloads > response_1710067669379.json > ...
{
"predicted_topics": [
{
"topic_number": 3,
"associated_words": "fake,news,democrats,biden,crooked",
"probability": 0.5930420160293579
},
{
"topic_number": 4,
"associated_words": "president,run,vote,makeamericagreatagain,needs",
"probability": 0.501744270324707
},
{
"topic_number": 8,
"associated_words": "obama,china,border,obamacare,states",
"probability": 0.44415485858917236
},
{
"topic_number": 1,
"associated_words": "thank,vote,today,maga,governor",
"probability": 0.3323463201522827
},
{
"topic_number": 9,
"associated_words": "interview,apprentice,celebapprentice,watch,enjoy",
"probability": 0.329071581363678
}
]
}

frontEnd.py:


127.0.0.1:7860
Tweet
Vindman's behavior is a scandal. He should be removed from the @RealDonaldTrump White House ASAP to protect our foreign poli…
Number Of Topics
3
Clear
Submit
output
Topic number: 3, Associated words: fake,news,democrats,biden,crooked, Probability: 0.5930420160293579
Topic number: 4, Associated words: president,run,vote,makeamericagreatagain,needs, Probability: 0.501744270324707
Topic number: 8, Associated words: obama,china,border,obamacare,states, Probability: 0.44415485858917236
Flag