Topic Modelling with BERTopic

January 3, 2022

Topic Modelling with BERTopic

To deal with this large amount of text, we look towards topic modeling. A technique to automatically extract meaning from documents by identifying recurrent topics.

`BERTopic` is a topic modelling technqiue that leverages BERT embeddings and a class-based TF-IDF to create dense clusters allowing for easily interpretable topics whilst keeping important words in the topic descriptions.

The BerTopic algorithm contains 3 stages:

`1. Embed the textual data(documents)` In this step, the algorithm extracts document embeddings with BERT, or it can use any other embedding technique. By default, it uses the following sentence transformers "paraphrase-MiniLM-L6-v2"- This is an English BERT-based model trained specifically for semantic similarity tasks. "paraphrase-multilingual-MiniLM-L12-v2"- This is similar to the first, with one major difference is that the xlm models work for 50+ languages.

`2. Cluster Documents` It uses UMAP to reduce the dimensionality of embeddings and the HDBSCAN technique to cluster reduced embeddings and create clusters of semantically similar documents.

`3. Create a topic representation` The last step is to extract and reduce topics with class-based TF-IDF and then improve the coherence of words with Maximal Marginal Relevance.

```
<img width="512px" src='https://miro.medium.com/max/700/0*CMkR9LeJvOVJ0XGG' />
<p style="text-align: center;color:gray">Figure 1: BERT Classification Model</p>
```

This project/notebook consists of several Tasks.

- **Task 1**: Installing all dependencies.
- **Task 2**: Importing the required libraries in the environment.
- **Task 3**: Re-usable Functions
- **Task 4**: Exploratory Data Analysis
- **Task 5**: Modelling
- **Task 6**: Number of Topics Analysis
- **Task 7**: Finding Similar Topics
- **Task 8**: Assigning new keywords to existing topics generated

0.0.1 Task 1: Installing all the dependencies

[1]:
```
### Installing all the dependencies
!pip install bertopic[visualization] --quiet
```

```
Exception:
Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2851, in _dep_map
    return self.__dep_map
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2685, in __getattr__
    raise AttributeError(attr)
AttributeError: _DistInfoDistribution__dep_map
During handling of the above exception, another exception occurred:
Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-packages/pip/basecommand.py", line 209, in
main
    status = self.run(options, args)
  File "/opt/conda/lib/python3.7/site-packages/pip/commands/install.py", line
310, in run
    wb.build(autobuilding=True)
  File "/opt/conda/lib/python3.7/site-packages/pip/wheel.py", line 748, in build
    self.requirement_set.prepare_files(self.finder)
  File "/opt/conda/lib/python3.7/site-packages/pip/req/req_set.py", line 360, in
prepare_files
    ignore_dependencies=self.ignore_dependencies))
  File "/opt/conda/lib/python3.7/site-packages/pip/req/req_set.py", line 647, in
_prepare_file
    set(req_to_install.extras) - set(dist.extras)
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2810, in extras
    return [dep for dep in self._dep_map if dep]
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2853, in _dep_map
    self.__dep_map = self._compute_dependencies()
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2886, in
_compute_dependencies
    common = frozenset(reqs_for_extra(None))
  File "/opt/conda/lib/python3.7/site-
```

[2]:
```
pip install pip==8.1.1
```

```
Requirement already satisfied (use --upgrade to upgrade): pip==8.1.1 in
/opt/conda/lib/python3.7/site-packages
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
pip install numpy==1.20
```

```
Requirement already satisfied (use --upgrade to upgrade): numpy==1.20 in
/opt/conda/lib/python3.7/site-packages
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[4]:
```
!pip install WordCloud
from wordcloud import WordCloud
```

```
Requirement already satisfied (use --upgrade to upgrade): WordCloud in
/opt/conda/lib/python3.7/site-packages
Requirement already satisfied (use --upgrade to upgrade): pillow in
/opt/conda/lib/python3.7/site-packages (from WordCloud)
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.6.1 in
/opt/conda/lib/python3.7/site-packages (from WordCloud)
Requirement already satisfied (use --upgrade to upgrade): matplotlib in
/opt/conda/lib/python3.7/site-packages (from WordCloud)
Requirement already satisfied (use --upgrade to upgrade): pyparsing>=2.2.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->WordCloud)
Requirement already satisfied (use --upgrade to upgrade): kiwisolver>=1.0.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->WordCloud)
Requirement already satisfied (use --upgrade to upgrade): python-dateutil>=2.7
in /opt/conda/lib/python3.7/site-packages (from matplotlib->WordCloud)
Requirement already satisfied (use --upgrade to upgrade): cycler>=0.10 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->WordCloud)
Requirement already satisfied (use --upgrade to upgrade): six>=1.5 in
/opt/conda/lib/python3.7/site-packages (from python-
dateutil>=2.7->matplotlib->WordCloud)
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
```

[5]:
```
pip install openpyxl
```

```
Requirement already satisfied (use --upgrade to upgrade): openpyxl in
/opt/conda/lib/python3.7/site-packages
Requirement already satisfied (use --upgrade to upgrade): et-xmlfile in
```

```
/opt/conda/lib/python3.7/site-packages (from openpyxl)
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

0.0.2 Task 2: Importing the necessary libraries in the environment.

[6]:
```python
#Importing Libraries
import numpy as np
import pandas as pd
import openpyxl
from copy import deepcopy
from bertopic import BERTopic

import matplotlib.pyplot as plt

import plotly as py
import plotly.graph_objs as go
import ipywidgets as widgets
from scipy import special
import plotly.express as px

py.offline.init_notebook_mode(connected = True)
```

```
2021-12-14 09:17:41.058079: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-12-14 09:17:41.058218: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
```

[ ]:
```python
df = pd.read_csv('df.csv')
df = df.rename({'SubSegment':'Topic'}, axis = 1)
df[['Topic', 'col2']] = df['Topic'].str.split(' - ', expand=True)
df['col2'] = df['col2'].str.replace('col2','col')
df = df.rename_axis('Index').reset_index()
df.head()
```

0.0.3 Task 3: Re-usable Functions

[8]:
```python
def get_topic_val(modelname, topics):
    """
    Input: a) modelname: Name of the BERTopic() model used.
           b) topics: The list of topics generated by the model.

    Function: Takes in the input and returns a dictionary with topic names as␣
↪the keys and the keyword's index values from the df as the values.
```

```
    """

    grouped_topics = {topic: [] for topic in set(topics)}

    for index, topic in enumerate(topics):
        grouped_topics[topic].append(index)

    return grouped_topics
```

[9]:
```
def make_result_df(dictionary, topicsdict):
    ''''
    Input: a) Dictionary: The dictionary with results of dict of get_topic_val␣
↪function. It has all the topics as the keys with their respective keywords␣
↪as the index given by the model.
           b) topicsdict: The dictionary with the index of the corresponding␣
↪keyword row in the dataframe as the key and their string keyword as the␣
↪value.

    Function: Takes in the inputs, maps the index values of keywords with their␣
↪actual keyword names and returns a result dataframe.
    '''

    key = []
    re_keywords = []
    val = dictionary.items()

    for i, value in val:
        key.append(i)
        val = [*map(topicsdict.get, value)]
        re_keywords.append(val)

    new_dict = {k: v for k, v in zip(key, re_keywords)}
    result_df = pd.DataFrame(new_dict.items(), columns = ['Topic␣
↪Nr','Present_Input_Keywords'])
    result_df = result_df.rename_axis('Index').reset_index()


    return result_df
```

[10]:
```
def representativedocs(model, topics, docs, keywords):
    """
    Input: a) Model: Name of the model you want the results for.
           b) topics: topics extracted by the model
           c) docs: documents given as the input to the model. This is the␣
↪different topic names that the model suggests. (Top n)
           d) Keywords: the input keywords given to the model
```

```
    Function: Takes in all the inputs and extracts the representative documents␣
↪per topic.
    """
    model.get_topic_info()

    #extracting the topic names/numbers
    top_names = model.topic_names
    top_names = pd.DataFrame(top_names.items(), columns = [topics,docs])

    #extracting representative docs for all the topics
    rep_docs = model.representative_docs
    rep_docs = pd.DataFrame(rep_docs.items(), columns = [topics, keywords])

    #get topics with probability
    top_proba = model.get_topics()

    output = pd.merge(top_names,
                rep_docs,
                how='left',
                left_on='topic_num',
                right_on='topic_num')
    return output
```

[12]:
```
def make_final_dataframe(model, representdocsdf):
    """
    Inputs: a) Model: name of the model
            b) dataframe1: This is the dataframe formed including the topics␣
↪and their top n topic names for each
            c) representdocsdf: This is the resultant dataframe of the␣
↪representative docs function


    Function: Returns the resultant dataframe with topic number, their top n␣
↪names with c-tf-idf scores and all the keywords they contain.
    """
    dataframe1 = pd.DataFrame(model.topics.items(), columns = ['Topic Nr',␣
↪'Possible Topic Names'])
    finaldfname = pd.merge(dataframe1, representdocsdf)

    return finaldfname
```

[963]:
```
from sklearn.metrics.pairwise import cosine_similarity

def get_similarity_score(model):
    '''
```

```
    Parameters:
        Inputs: a) model: the model used to train your topic modelling
                b) topicnr: the topic for which you want to see the similarity␣
↪score. IMP: here the nr is the index of the row and not the topic nr so for␣
↪topic -1 = topicnr is 0
                c) resultdf: the resultant df to merge to get combined results
                d) threshold: the threshold above which you want to get similar␣
↪topics
        Ouput: A pandas dataframe with topicnr, topic names, keywords present␣
↪(input) and the distance score for each.
    '''

    topics = sorted(list(model.get_topics().keys()))

    # Extract topic words and their frequencies
    topic_list = sorted(topics)

    embeddings = model.c_tf_idf
    distance_matrix = cosine_similarity(embeddings)

    most_similar_ind = []
    most_similar_val = []

    for topic in topic_list:
        data = distance_matrix[topic] #topic -1
        i = np.argsort(data, axis=0)[-2]
        most_similar_ind.append(i)   #ensure length and order for the list
        most_similar_val.append(data[i])

    similar_df = pd.DataFrame()
    similar_df['Topic Nr'] = topic_list
    similar_df['most_similar'] =  most_similar_ind
    similar_df['similarity_score'] = most_similar_val
    return similar_df
```

0.0.4 Task 4: Exploratory Data Analysis

`Long-tail keywords` are unpopular keyword phrases with low search volume and high variation. In other words, these queries are only searched a few times per month because they are very specific keywords, or because people phrase their queries in many different ways.

[ ]:
```
##Looking for the long-tail keywords in our dataframe
tailnr = 17
df[df['Keyword'].apply(lambda x: len(x)>tailnr)]
```

[ ]:
```
wordcloud2 = WordCloud().generate(' '.join(df['Keyword']))
plt.figure(figsize = (10, 8), facecolor = None)
```

```
plt.imshow(wordcloud2)
plt.axis("off")
plt.show()
```

Analysis from the wordcloud of the keywords:

1) Top three keywords in our df are a,b, and c.
2) We can see different types of styles: type1, type2, type3, type 4 ....
3) We also see most people in this dataset search for branded things that is of name.
4) We have one location mentioned: location.
5) We see people searching for comparing keywords for different features related to topic and topic.
6) Some also want to learn how to use a specific functionality of topic, so name.

[ ]:
```
fig = px.histogram(df,x='Topic')
fig.show()
```

[ ]:
```
fig = px.histogram(df,x='col2')
fig.show()
```

Checking the topics with less count what keywords they contain

[ ]:
```
df[df['Topic']=='topicname']
```

[ ]:
```
df[df['Topic']=='topicname']
```

[ ]:
```
df[df['Topic']=='topicname']
```

[ ]:
```
df[df['Topic']=='topicname']
```

0.0.5 Task 5: Modelling

[21]:
```
docs = list(df.loc[:,'Keyword'].values)
```

[ ]:
```
docs[:5]
```

0.0.6 Embeddings

Sentence Transformers are the SOTA technique for sentence, text and image embeddings. * Useful for semantic similarity * Semantic Search * Paraphrasing Mining

We can select different sentence transformers embedding models from: https://www.sbert.net/docs/pretrained_models.html

[23]:
```
sent_topic_model =␣
 ↪BERTopic(embedding_model="xlm-r-bert-base-nli-stsb-mean-tokens",calculate_probabilities=Tru
 ↪3))
topics, probs = sent_topic_model.fit_transform(docs)
```

```
Batches:   0%|          | 0/60 [00:00<?, ?it/s]

2021-12-14 09:18:48,861 - BERTopic - Transformed documents to Embeddings
2021-12-14 09:19:03,347 - BERTopic - Reduced dimensionality with UMAP
2021-12-14 09:19:03,757 - BERTopic - Clustered UMAP embeddings with HDBSCAN
2021-12-14 09:20:48,974 - BERTopic - Reduced number of topics from 62 to 33
```

[ ]:
```
sent_topic_model.get_topic_info()
```

[25]:
```
topic_count = sent_topic_model.get_topic_freq()
```

[ ]:
```
topic_count.info()
```

[ ]:
```
fig = px.bar(topic_count,x='Topic',y='Count', title = 'Distribution of Topic␣
↪Generated')
fig.show()
```

0.0.7 Task 6: Number of Topic Analysis

[ ]:
```
most_similar_dict = dict(zip(df.Index, df.Keyword))
grouped_topics = get_topic_val(sent_topic_model, topics)
res_df = make_result_df(grouped_topics,most_similar_dict)
result_df = make_final_dataframe(sent_topic_model,res_df)
#result_df['count'] = result_df['Present_Input_Keywords']
result_df['count'] = result_df['Present_Input_Keywords'].apply(lambda x: len(x))
result_df
```

[ ]:
```
result_df['count'].sum()
```

[ ]:
```
output.shape
```

[43]:
```
pip install chart_studio
```

```
huggingface/tokenizers: The current process just got forked, after parallelism
has already been used. Disabling parallelism to avoid deadlocks…
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true |
false)
Collecting chart-studio
  Downloading https://files.pythonhosted.org/packages/ca/ce/330794a6b6ca4b9182c3
8fc69dd2a9cbff60fd49421cb8648ee5fee352dc/chart_studio-1.1.0-py3-none-any.whl
(64kB)
    100% |                 | 71kB 5.6MB/s ta 0:00:011
Requirement already satisfied (use --upgrade to upgrade): retrying>=1.3.3
in /opt/conda/lib/python3.7/site-packages (from chart-studio)
Requirement already satisfied (use --upgrade to upgrade): plotly in
/opt/conda/lib/python3.7/site-packages (from chart-studio)
```

```
Requirement already satisfied (use --upgrade to upgrade): requests in
/opt/conda/lib/python3.7/site-packages (from chart-studio)
Requirement already satisfied (use --upgrade to upgrade): six in
/opt/conda/lib/python3.7/site-packages (from chart-studio)
Requirement already satisfied (use --upgrade to upgrade): certifi>=2017.4.17 in
/opt/conda/lib/python3.7/site-packages (from requests->chart-studio)
Requirement already satisfied (use --upgrade to upgrade): urllib3<1.27,>=1.21.1
in /opt/conda/lib/python3.7/site-packages (from requests->chart-studio)
Requirement already satisfied (use --upgrade to upgrade): chardet<5,>=3.0.2 in
/opt/conda/lib/python3.7/site-packages (from requests->chart-studio)
Requirement already satisfied (use --upgrade to upgrade): idna<3,>=2.5 in
/opt/conda/lib/python3.7/site-packages (from requests->chart-studio)
Installing collected packages: chart-studio
Successfully installed chart-studio-1.1.0
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[44]:
```
import chart_studio

username = 'akshara.shukla'
api_key = 'apikey'

chart_studio.tools.set_credentials_file(username=username, api_key = api_key)

import chart_studio.plotly as py
import chart_studio.tools as tls
```

[ ]:
```
fig = sent_topic_model.visualize_topics()
fig
```

[45]:
```
py.plot(fig, filename = 'Intertopic Distance Map df', auto_open = True)
```

[45]:
```
'https://plotly.com/~akshara.shukla/1/'
```

get_topics() Return top n words for a specific topic and their c-TF-IDF scores

[ ]:
```
sent_topic_model.get_topic(16)
```

Having generated topic embeddings, through both c-TF-IDF and embeddings, we can create a similarity matrix by simply applying cosine similarities through those topic embeddings. The result will be a matrix indicating how similar certain topics are to each other.

[ ]:
```
sent_topic_model.visualize_heatmap(n_clusters=4)
```

Combining documents with similarity score higher than 0.70

1. Document -1 = 1(0.89), 2 (0.74), 6 (0.84), 11 (0.78), 13 (0.75),22 (0.74)

2. Document 1 = 22 (0.80), 13 (0.77), 11 (0.81), 6 (0.80), -1 (0.89)
3. Document 6
4. 11 and 14

```
[ ]: sent_topic_model.get_topic(-1)
```

We can visualize the selected terms for a few topics by creating bar charts out of the c-TF-IDF scores for each topic representation. Insights can be gained from the relative c-TF-IDF scores between and within topics.

```
[ ]: sent_topic_model.visualize_barchart(topics = [1,2,3,4,5,6,7])
```

0.0.8 Task 7: Finding Similar Topics

```
[ ]: get_similarity_score(sent_topic_model)
```

```
[ ]: sent_topic_model.get_topic(13)
```

```
[ ]: sent_topic_model.get_topic(13)
```

0.0.9 Task 8: Assigning new keywords to existing topics generated

```
[ ]: similar_topics, similarity = sent_topic_model.find_topics("cheap binoculars",␣
     ↪top_n=5);
     print(similar_topics)
     print(similarity)
```

```
[ ]: sent_topic_model.find_topics("topicnr")
```