Aravind Patnam Jeremy Tan Timothy Le

CSE-184 Final Project - Trending Here Trending There An analysis on trending and nontrending Youtube videos.

In [1]:

```
import scripts.scripts as script
%run "scripts/imports.py"
import plotly.io as pio
pio.renderers.default = 'iframe'
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

```
/usr/local/lib/python3.7/site-packages/botocore/awsrequest.py:624: DeprecationWarning: Using or importing the ABCs from 'collections' i
ons.abc' is deprecated, and in 3.8 it will stop working
  class HeadersDict(collections.MutableMapping):
/usr/local/lib/python3.7/site-packages/gensim/corpora/dictionary.py:11: DeprecationWarning: Using or importing the ABCs from 'collectio
lections.abc' is deprecated, and in 3.8 it will stop working
  from collections import Mapping, defaultdict
/usr/local/lib/python3.7/site-packages/scipy/sparse/sparsetools.py:21: DeprecationWarning: `scipy.sparse.sparsetools` is deprecated!
scipy.sparse.sparsetools is a private module for scipy.sparse, and should not be used.
  _deprecated()
/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:
```

```
inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

/usr/local/lib/python3.7/site-packages/plotly/express/_doc.py:451: DeprecationWarning:

inspect.getargspec() is deprecated since Python 3.0, use inspect.signature() or inspect.getfullargspec()

[nltk_data] Downloading package wordnet to /Users/aravind/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/aravind/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/aravind/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [2]:

```
## written by Aravind Patnam and Jeremy Tan
#### get all csv dataframes for trending. These files should be in the same directory.
US_trending_df = pd.read_csv('data/USvideos.csv') #USA
CA_trending_df = pd.read_csv('data/CAvideos.csv') #CANADA
DE_trending_df = pd.read_csv('data/DEvideos.csv') #GERMANY
FR_trending_df = pd.read_csv('data/FRvideos.csv') #FRANCE
GB_trending_df = pd.read_csv('data/GBvideos.csv') #GREAT BRITAIN
IN_trending_df = pd.read_csv('data/INvideos.csv') #INDIA
JP_trending_df = pd.read_csv('data/JPvideos.csv', encoding='ISO-8859-1') #JAPAN
KR_trending_df = pd.read_csv('data/KRvideos.csv' , encoding='ISO-8859-1') #SOUTH KOREA
MX_trending_df = pd.read_csv('data/MXvideos.csv', encoding='ISO-8859-1') #MEXICO
RU_trending_df = pd.read_csv('data/RUvideos.csv', encoding='ISO-8859-1') #RUSSIA

list_of_all_trending_dfs = [US_trending_df, CA_trending_df, DE_trending_df, FR_trending_df, GB_trending_df, IN_trending_df,
                            JP_trending_df, KR_trending_df, MX_trending_df, RU_trending_df]
list_of_csvs = ['data/USvideos.csv','data/CAvideos.csv', 'data/DEvideos.csv', 'data/FRvideos.csv', 'data/GBvideos.csv', 'data/INvideos.
sv','data/KRvideos.csv', 'data/MXvideos.csv', 'data/RUvideos.csv' ]

big_df = list()
for csv in list_of_csvs:
    # use encoding to bypass utf error
    df = pd.read_csv(csv, index_col='video_id', encoding='ISO-8859-1')
    # add new column called "country" to indentify which videos the csv are coming from
    # depending on your path name, this will break as it looks at the path name
    df['country'] = csv[5:7]
    big_df.append(df)

full_trending_df = pd.concat(big_df)
full_trending_df.tail()
```

Out[2]:

| | trending_date | title | channel_title | category_id | publish_time | tags | views | likes | dislikes | comment_count | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| video_id | | | | | | | | | | | |
| OMmR9THjVKM | 18.14.06 | Ð£ Ð¼ÐµÐ½Ñ Ð¡ÐÐÐ! (Story booth Ð½Ð° ÑÑÑ... | Pastime Time | 22 | 2018-06-13T13:47:01.000Z | story booth Ð½Ð° ÑÑÑÑÐºÐ¾Ð¼\|"story booth Ð... | 129488 | 5893 | 164 | 990 | https:/ |
| tX7p7NtNVDE | 18.14.06 | ÐÐ¾Ð¸ ÐÐµÑÑÐºÐ¸Ðµ Ð¢ÑÐ°Ð²Ð¼Ñ 2 (Ð°Ð½Ð¸Ð¼... | CaGArt | 23 | 2018-06-12T09:38:38.000Z | ÐºÐ°Ð³Ð°ÑÑ\|"ÐºÐ°Ð³Ð¸"\|"ÑÐ°Ð³Ð°ÑÑ"\|"Ð³Ð°Ð³... | 99163 | 4659 | 337 | 692 | http |
| KAyj5Xm1C64 | 18.14.06 | [ENG SUB] BTS PROM PARTY 2018 Intro + 2nd Gran... | DaisyxBTS 07 | 24 | 2018-06-13T12:51:23.000Z | [none] | 449611 | 24808 | 93 | 974 | http |
| 4PiSLIrsSiY | 18.14.06 | ÐÐÐÐÐ ÐÐÐÐÐÐ ÐÐÐÐÐÐ¢Ð«ð/ Ð¢Ð... | ÐÑÑÑÐ° ÐÐµÐ±ÐµÐ ´ÐµÐ²Ð° | 22 | 2018-06-13T00:23:33.000Z | ÑÑÐ¼ ÑÑÑ Ð²Ð°Ð½Ð½Ð¾Ð¹ ÐºÐ¾Ð¼Ð½Ð°ÑÑ\|"roo... | 14225 | 793 | 39 | 209 | h |
| Ehy5foVfKOE | 18.14.06 | ÐÐ»Ð¾ÑÐ¾Ð¹ ÑÐ¸Ð³Ð½Ð°Ð». Ð¡ÑÐ°Ð½Ñ ÑÐµÐ»Ð¾... | Tubus Show | 29 | 2018-06-13T09:50:09.000Z | Ð¿Ð»Ð¾ÑÐ¾Ð¹ ÑÐ¸Ð³Ð½Ð°Ð»\|"ÑÑÐ±ÑÑ ÑÐ¾Ñ"\|... | 52340 | 7708 | 133 | 1819 | htt |

In [3]:

```
#### reformatting and detecting nans

# reformat trending_date
full_trending_df['trending_date'] = pd.to_datetime(full_trending_df['trending_date'],errors='coerce', format='%y.%d.%m')
full_trending_df['publish_time'] = pd.to_datetime(full_trending_df['publish_time'], errors='coerce', format='%Y-%m-%dT%H:%M:%S.%fZ')

# detects any nans
full_trending_df = full_trending_df[full_trending_df['trending_date'].notnull()]
full_trending_df = full_trending_df[full_trending_df['publish_time'].notnull()]

# drop all nans by trmeoving them
full_trending_df = full_trending_df.dropna(how='any',inplace=False, axis = 0)

# this is done already so don't run it twice
full_trending_df.insert(4, 'publish_date', full_trending_df['publish_time'].dt.date)
full_trending_df['publish_time'] = full_trending_df['publish_time'].dt.time

# set index by video id and sort by trending dates
full_trending_df_fill = full_trending_df.reset_index().sort_values('trending_date').set_index('video_id')
# set index by vide id and sort by trending dates, but make sure to drop duplicates
full_trending_df = full_trending_df.reset_index().sort_values('trending_date').drop_duplicates('video_id',keep='last').set_index('video
# prep data to by adding like rate and spliitng publish time into a hour, min, and sec column
full_trending_df['like_rate'] =  full_trending_df ['likes'] / full_trending_df['views'] * 100
full_trending_df[['hour','min','sec']] = full_trending_df['publish_time'].astype(str).str.split(':', expand=True).astype(int)
full_trending_df.head()
```

Out[3]:

| | trending_date | title | channel_title | category_id | publish_date | publish_time | tags | views | likes | dislikes | ... | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| video_id | | | | | | | | | | | | |
| GSid9wlRqBQ | 2017-11-14 | Julien Bam reagiert auf HATE Kommentare (zum a... | Julien Bam | 24 | 2017-11-11 | 11:00:02 | Julien\|"Bam"\|"Hate"\|"Kommentare"\|"Reagieren"\|"... | 1186759 | 134977 | 5704 | ... | ht |
| KNq8laLAqcc | 2017-11-14 | à¹à¸à¸à¸à¸·à¸- à¸à¸£à¸«à¸¡à¸¥à¸ ´à¸à¸´à¸ ... | one31 | 24 | 2017-11-13 | 14:51:22 | [none] | 363046 | 1375 | 70 | ... | h |
| j8h7KEGcswk | 2017-11-14 | Engel 11:11 Portal Orakel fÃ¼r die Woche vom 1... | DasEngelOrakel | 1 | 2017-11-12 | 09:33:38 | [none] | 13363 | 336 | 20 | ... | h |
| 5MYXzKS95XY | 2017-11-14 | Denunziantentum heiÃt jetzt Zivilcourage | Achgut.Pogo | 25 | 2017-11-13 | 14:03:52 | Gerald Hensel\|"Scholz & Friends"\|"Kein Geld fÃ... | 6051 | 526 | 8 | ... | htt |
| _UEk3WRixnc | 2017-11-14 | Bodybuilder bekommen Platzwunden - Paintball C... | HARDGAINER CREW | 17 | 2017-11-13 | 20:10:36 | hardcore bodybuilding\|"hardgainer crew"\|"hardg... | 31500 | 3122 | 28 | ... | ht |

5 rows × 21 columns

In [4]:

```
## written by Aravind Patnam

## run these to get the non trending datasets generated from the youtube api. These files should be in the same directory
not_trending_us_df = pd.read_csv('data/not_trending_us_df.csv')
not_trending_ca_df = pd.read_csv('data/not_trending_ca_df.csv')
not_trending_de_df = pd.read_csv('data/not_trending_de_df.csv')
not_trending_fr_df = pd.read_csv('data/not_trending_fr_df.csv')
not_trending_gb_df = pd.read_csv('data/not_trending_gb_df.csv')
not_trending_in_df = pd.read_csv('data/not_trending_in_df.csv')
not_trending_jp_df = pd.read_csv('data/not_trending_jp_df.csv')
not_trending_kr_df = pd.read_csv('data/not_trending_kr_df.csv')
not_trending_mx_df = pd.read_csv('data/not_trending_mx_df.csv')
not_trending_ru_df = pd.read_csv('data/not_trending_ru_df.csv')
```

In [ ]:

```
## written by Aravind Patnam
### do not run this unless you have a lot of time and 10 Youtube API keys!!!. Datasets are already generated for you for testing.
n = 50
US_trending_videoIds = US_trending_df.sample(n)['video_id'].tolist()
CA_trending_videoIds = CA_trending_df.sample(n)['video_id'].tolist()
DE_trending_videoIds = DE_trending_df.sample(n)['video_id'].tolist()
FR_trending_videoIds = FR_trending_df.sample(n)['video_id'].tolist()
GB_trending_videoIds = GB_trending_df.sample(n)['video_id'].tolist()
IN_trending_videoIds = IN_trending_df.sample(n)['video_id'].tolist()
JP_trending_videoIds = JP_trending_df.sample(n)['video_id'].tolist()
KR_trending_videoIds = KR_trending_df.sample(n)['video_id'].tolist()
MX_trending_videoIds = MX_trending_df.sample(n)['video_id'].tolist()
RU_trending_videoIds = RU_trending_df.sample(n)['video_id'].tolist()

## written by Aravind Patnam
### do not run this unless you have a lot of time and 10 Youtube API keys!!! Datasets are already generated for you for testing.
## do following requests separately with a new API Key and have the file called "apiKey"
not_trending_us_df = script.process_youtube_requests(US_trending_videoIds)
not_trending_ca_df = script.process_youtube_requests(CA_trending_videoIds)
not_trending_de_df = script.process_youtube_requests(DE_trending_videoIds)
not_trending_fr_df = script.process_youtube_requests(FR_trending_videoIds)
not_trending_gb_df = script.process_youtube_requests(GB_trending_videoIds)
not_trending_in_df = script.process_youtube_requests(IN_trending_videoIds)
not_trending_jp_df = script.process_youtube_requests(JP_trending_videoIds)
not_trending_kr_df = script.process_youtube_requests(KR_trending_videoIds)
not_trending_mx_df = script.process_youtube_requests(MX_trending_videoIds)
not_trending_ru_df = script.process_youtube_requests(RU_trending_videoIds)
```

In [5]:

```
## written by Aravind Patnam

## puts all nontrending and all dfs that we have together for one big df and multiple smaller ones
list_of_all_nontrending_dfs = [not_trending_us_df, not_trending_ca_df, not_trending_de_df,
                               not_trending_fr_df, not_trending_gb_df, not_trending_in_df, not_trending_jp_df,
                               not_trending_kr_df, not_trending_mx_df, not_trending_ru_df]


# aravind, the categories are already in the csv??????????????????
full_nontrending_df = pd.concat(list_of_all_nontrending_dfs)
allDfsList = list_of_all_trending_dfs + list_of_all_nontrending_dfs + [full_trending_df] + [full_nontrending_df]
allDfsDf = pd.concat(allDfsList)
allDfsList.append(allDfsDf)

## written by Jeremy Tan
## insert new category field into dataframes
for df in allDfsList:
    script.insert_category_field(df)


## written by Jeremy Tan
## converts all columns in dataframes to type for analysis
for df in list_of_all_nontrending_dfs:
    df['video_id'] = df['video_id'].astype(str)
    df['title'] = df['title'].astype(str)
    df['channel_title'] = df['channel_title'].astype(str)
    df['category_id'] = df['category_id'].astype(int)
    #df['category'] = df['category'].astype(str)
    df['tags'] = df['tags'].astype(str)
    df['views'] = df['views'].astype(int)
    df['likes'] = df['likes'].astype(int)
    df['dislikes'] = df['dislikes'].astype(int)
    df['comment_count'] = df['comment_count'].astype(int)
    df['description'] = df['description'].astype(str)
```

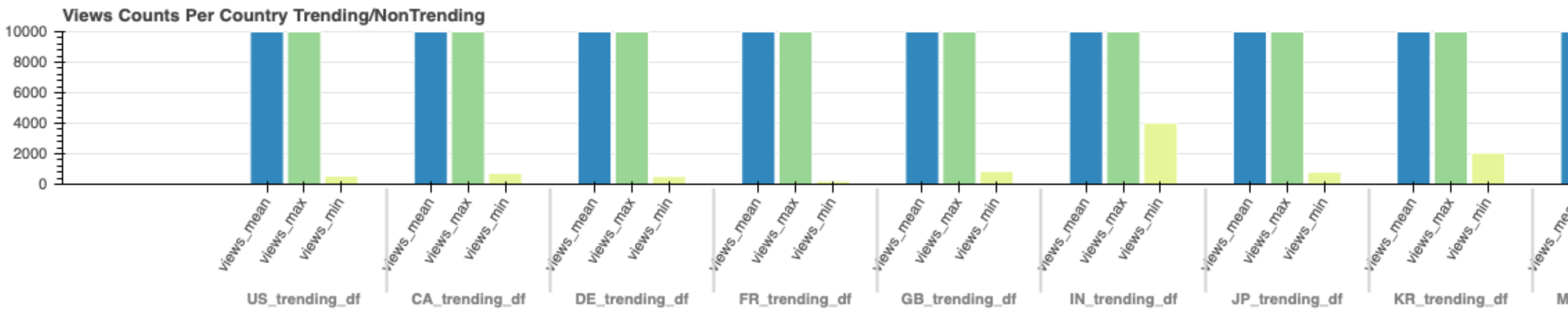
In [9]:

```
## written by Aravind Patnam
full_nontrending_df = pd.concat(list_of_all_nontrending_dfs)
allDfsList = list_of_all_trending_dfs + list_of_all_nontrending_dfs + [full_trending_df] + [full_nontrending_df]
allDfsDf = pd.concat(allDfsList)
allDfsList.append(allDfsDf)

## output is a map containing all the numeric data for each df
describes = []
for df in allDfsList:
    describes.append(script.find_stats(df))
describesKeys = ['US_trending_df', 'CA_trending_df', 'DE_trending_df', 'FR_trending_df', 'GB_trending_df',
                'IN_trending_df', 'JP_trending_df', 'KR_trending_df', 'MX_trending_df', 'RU_trending_df',
                 'not_trending_us_df', 'not_trending_ca_df', 'not_trending_de_df', 'not_trending_fr_df',
                'not_trending_gb_df', 'not_trending_in_df', 'not_trending_jp_df', 'not_trending_kr_df',
                 'not_trending_mx_df', 'not_trending_ru_df',
                'full_trending_df', 'full_nontrending_df', 'allDfsDf']
describeMap = dict(zip(describesKeys, describes))
## make sense of each data point we have from the describeMap and clean it for visualization
countries = list(describeMap.keys())
## written by Aravind Patnam
likes_count, dislikes_count = script.do_describe_analysis(describeMap, countries, describesKeys)
```

(https://bokeh.org) Loading BokehJS ...

Views Analysis | Likes Analysis | Dislikes Analysis | Comments Analysis | Tags Analysis

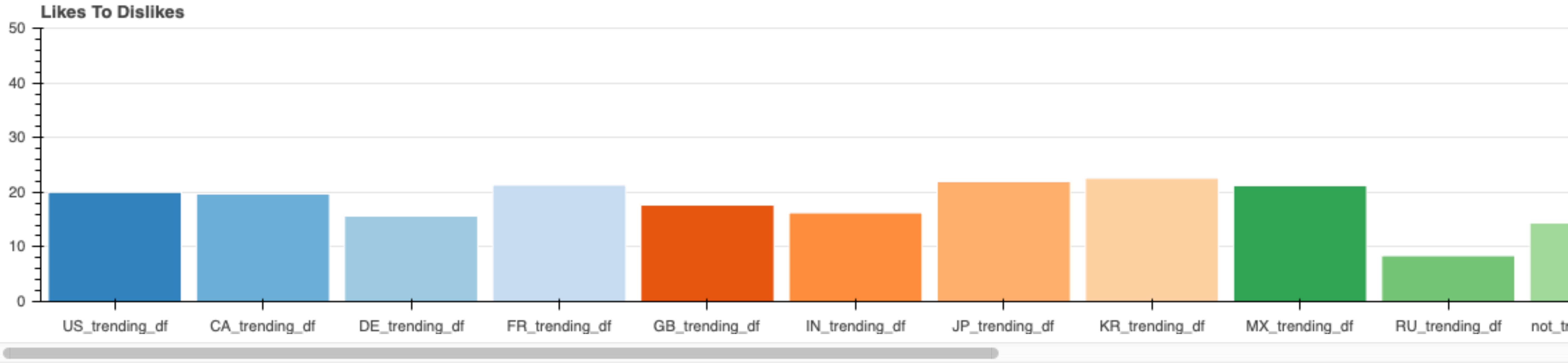
Views Analysis | Likes Analysis | Dislikes Analysis | Comments Analysis | Tags Analysis


Views Counts Per Country Trending/NonTrending

In [11]:

```
script.do_likes_to_dislikes_analysis(likes_count, dislikes_count, countries)
```

(https://bokeh.org) BokehJS 1.4.0 successfully loaded.



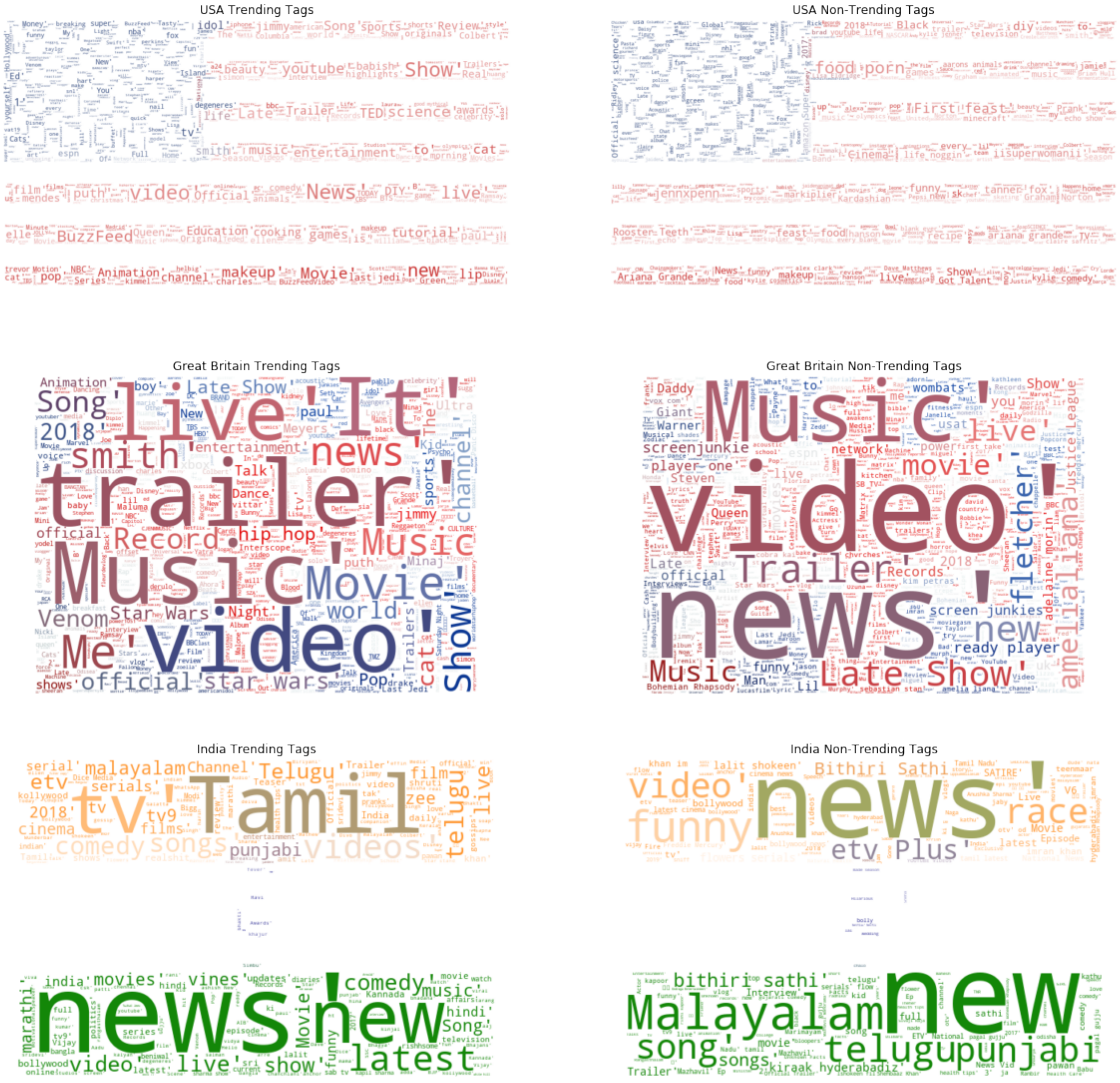
In [10]:

```
## written by Aravind Patnam

## a bunch of maps containing the most popular tags for each country that are both trending and not trending and their frequency
us_trending_most_common_tags = script.get_most_common_tags(US_trending_df)
ca_trending_most_common_tags = script.get_most_common_tags(CA_trending_df)
de_trending_most_common_tags = script.get_most_common_tags(DE_trending_df)
fr_trending_most_common_tags = script.get_most_common_tags(FR_trending_df)
gb_trending_most_common_tags = script.get_most_common_tags(GB_trending_df)
in_trending_most_common_tags = script.get_most_common_tags(IN_trending_df)
jp_trending_most_common_tags = script.get_most_common_tags(JP_trending_df)
kr_trending_most_common_tags = script.get_most_common_tags(KR_trending_df)
mx_trending_most_common_tags = script.get_most_common_tags(MX_trending_df)
ru_trending_most_common_tags = script.get_most_common_tags(RU_trending_df)

us_nontrending_most_common_tags = script.get_most_common_tags(not_trending_us_df)
ca_nontrending_most_common_tags = script.get_most_common_tags(not_trending_ca_df)
de_nontrending_most_common_tags = script.get_most_common_tags(not_trending_de_df)
fr_nontrending_most_common_tags = script.get_most_common_tags(not_trending_fr_df)
gb_nontrending_most_common_tags = script.get_most_common_tags(not_trending_gb_df)
in_nontrending_most_common_tags = script.get_most_common_tags(not_trending_in_df)
jp_nontrending_most_common_tags = script.get_most_common_tags(not_trending_jp_df)
kr_nontrending_most_common_tags = script.get_most_common_tags(not_trending_kr_df)
mx_nontrending_most_common_tags = script.get_most_common_tags(not_trending_mx_df)
ru_nontrending_most_common_tags = script.get_most_common_tags(not_trending_ru_df)
```

Below is a wordcloud visualization of some of the countries and a comparison of them between trending and non-trending videos. Since there is not much space to do all 20 different
the most popular tags were displayed.

In [12]:

```
## written by Aravind Patnam
script.do_wordcloud(us_trending_most_common_tags, gb_trending_most_common_tags, in_trending_most_common_tags, us_nontrending_most_commo
_most_common_tags, in_nontrending_most_common_tags)
```



```
<Figure size 1440x1440 with 0 Axes>
```

Below represents a sentiment analysis conducted on the youtube tags. A datset of positive words and negative words is provided for the model to work.

In [13]:

```
## written by Aravind Patnam

## gets the classifications from the sentiment analysis and prints out accuracies of the model
classifications_us_trending = script.execute_model(list(us_trending_most_common_tags.keys()))
classifications_ca_trending = script.execute_model(list(ca_trending_most_common_tags.keys()))
classifications_de_trending = script.execute_model(list(de_trending_most_common_tags.keys()))
classifications_fr_trending = script.execute_model(list(fr_trending_most_common_tags.keys()))
classifications_gb_trending = script.execute_model(list(gb_trending_most_common_tags.keys()))
classifications_in_trending = script.execute_model(list(in_trending_most_common_tags.keys()))
classifications_jp_trending = script.execute_model(list(jp_trending_most_common_tags.keys()))
classifications_kr_trending = script.execute_model(list(kr_trending_most_common_tags.keys()))
classifications_mx_trending = script.execute_model(list(mx_trending_most_common_tags.keys()))
classifications_ru_trending = script.execute_model(list(ru_trending_most_common_tags.keys()))

classifications_us_nontrending = script.execute_model(list(us_nontrending_most_common_tags.keys()))
classifications_ca_nontrending = script.execute_model(list(ca_nontrending_most_common_tags.keys()))
classifications_de_nontrending = script.execute_model(list(de_nontrending_most_common_tags.keys()))
classifications_fr_nontrending = script.execute_model(list(fr_nontrending_most_common_tags.keys()))
classifications_gb_nontrending = script.execute_model(list(gb_nontrending_most_common_tags.keys()))
classifications_in_nontrending = script.execute_model(list(in_nontrending_most_common_tags.keys()))
classifications_jp_nontrending = script.execute_model(list(jp_nontrending_most_common_tags.keys()))
classifications_kr_nontrending = script.execute_model(list(kr_nontrending_most_common_tags.keys()))
classifications_mx_nontrending = script.execute_model(list(mx_nontrending_most_common_tags.keys()))
classifications_ru_nontrending = script.execute_model(list(ru_nontrending_most_common_tags.keys()))
```

```
Accuracy is: 0.7950162513542796
Accuracy is: 0.8043336944745395
Accuracy is: 0.7917659804983749
Accuracy is: 0.7943661971830986
Accuracy is: 0.7917659804983749
Accuracy is: 0.7919826652221018
Accuracy is: 0.7919826652221018
Accuracy is: 0.7965330444203683
Accuracy is: 0.7963163596966414
Accuracy is: 0.7926327193932827
Accuracy is: 0.7965330444203683
Accuracy is: 0.7863488624052004
Accuracy is: 0.7900325027085591
Accuracy is: 0.7917659804983749
Accuracy is: 0.79068255687974
Accuracy is: 0.7965330444203683
Accuracy is: 0.7967497291440954
Accuracy is: 0.7878656554712893
Accuracy is: 0.7859154929577464
Accuracy is: 0.7785482123510292
```

In [14]:

```
## written by Aravind Patnam

## calls above method for stats for each country for visualization

country_us_trending, us_trending_pos, us_trending_neg = script.get_sentiment_stats(classifications_us_trending, "USA_Trending")
country_ca_trending, ca_trending_pos, ca_trending_neg = script.get_sentiment_stats(classifications_ca_trending, "Canada_Trending")
country_de_trending, de_trending_pos, de_trending_neg = script.get_sentiment_stats(classifications_de_trending, "Denmark_Trending")
country_fr_trending, fr_trending_pos, fr_trending_neg = script.get_sentiment_stats(classifications_fr_trending, "France_Trending")
country_gb_trending, gb_trending_pos, gb_trending_neg = script.get_sentiment_stats(classifications_gb_trending, "GreatBritain_Trending"
country_in_trending, in_trending_pos, in_trending_neg = script.get_sentiment_stats(classifications_in_trending, "India_Trending")
country_jp_trending, jp_trending_pos, jp_trending_neg = script.get_sentiment_stats(classifications_jp_trending, "Japan_Trending")
country_kr_trending, kr_trending_pos, kr_trending_neg = script.get_sentiment_stats(classifications_kr_trending, "SouthKorea_Trending")
country_mx_trending, mx_trending_pos, mx_trending_neg = script.get_sentiment_stats(classifications_mx_trending, "Mexico_Trending")
country_ru_trending, ru_trending_pos, ru_trending_neg = script.get_sentiment_stats(classifications_ru_trending, "Russia_Trending")


country_us_nontrending, us_nontrending_pos, us_nontrending_neg = script.get_sentiment_stats(classifications_us_nontrending, "USA_NonTre
country_ca_nontrending, ca_nontrending_pos, ca_nontrending_neg = script.get_sentiment_stats(classifications_ca_nontrending, "Canada_Non
country_de_nontrending, de_nontrending_pos, de_nontrending_neg = script.get_sentiment_stats(classifications_de_nontrending, "Denmark_No
country_fr_nontrending, fr_nontrending_pos, fr_nontrending_neg = script.get_sentiment_stats(classifications_fr_nontrending, "France_Non
country_gb_nontrending, gb_nontrending_pos, gb_nontrending_neg = script.get_sentiment_stats(classifications_gb_nontrending, "GreatBrita
country_in_nontrending, in_nontrending_pos, in_nontrending_neg = script.get_sentiment_stats(classifications_in_nontrending, "India_NonT
country_jp_nontrending, jp_nontrending_pos, jp_nontrending_neg = script.get_sentiment_stats(classifications_jp_nontrending, "Japan_NonT
country_kr_nontrending, kr_nontrending_pos, kr_nontrending_neg = script.get_sentiment_stats(classifications_kr_nontrending, "SouthKorea
country_mx_nontrending, mx_nontrending_pos, mx_nontrending_neg = script.get_sentiment_stats(classifications_mx_nontrending, "Mexico_Non
country_ru_nontrending, ru_nontrending_pos, ru_nontrending_neg = script.get_sentiment_stats(classifications_ru_nontrending, "Russia_Non

countries = [country_us_trending, country_ca_trending, country_de_trending, country_fr_trending, country_gb_trending,
            country_in_trending, country_jp_trending, country_kr_trending, country_mx_trending, country_ru_trending,
            country_us_nontrending, country_ca_nontrending, country_de_nontrending, country_fr_nontrending,
            country_gb_nontrending, country_in_nontrending, country_jp_nontrending, country_kr_nontrending,
            country_mx_nontrending, country_ru_nontrending]
positivePercenteages = [us_trending_pos, ca_trending_pos, de_trending_pos, fr_trending_pos, gb_trending_pos, in_trending_pos, jp_trendi
            kr_trending_pos, mx_trending_pos, ru_trending_pos, us_nontrending_pos, ca_nontrending_pos, de_nontrending_pos, fr_nontrendi
            gb_nontrending_pos, in_nontrending_pos, jp_nontrending_pos, kr_nontrending_pos, mx_nontrending_pos, ru_nontrending_pos]
negativePercentages = [us_trending_neg, ca_trending_neg, de_trending_neg, fr_trending_neg, gb_trending_neg, in_trending_neg, jp_trendin
                      kr_trending_neg, mx_trending_neg, ru_trending_neg, us_nontrending_neg, ca_nontrending_neg, de_nontrending_neg,
                      fr_nontrending_neg, gb_nontrending_neg, in_nontrending_neg, jp_nontrending_neg, kr_nontrending_neg, mx_nontrendin
                      ru_nontrending_neg]
posToNegRatios = [i / j for i, j in zip(positivePercenteages, negativePercentages)]
data_dict = {"Country" : countries, "Positives": positivePercenteages, "Negatives": negativePercentages, "PositiveNegativeRatio": posTo
data = pd.DataFrame(data_dict, columns = ['Country', 'Positives' , 'Negatives', 'PositiveNegativeRatio'])

data
```

Out[14]:

| | Country | Positives | Negatives | PositiveNegativeRatio |
|---|---|---|---|---|
| 0 | USA_Trending | 195 | 805 | 0.242236 |
| 1 | Canada_Trending | 232 | 768 | 0.302083 |
| 2 | Denmark_Trending | 171 | 829 | 0.206273 |
| 3 | France_Trending | 138 | 862 | 0.160093 |
| 4 | GreatBritain_Trending | 211 | 789 | 0.267427 |
| 5 | India_Trending | 312 | 688 | 0.453488 |
| 6 | Japan_Trending | 95 | 905 | 0.104972 |
| 7 | SouthKorea_Trending | 85 | 915 | 0.092896 |
| 8 | Mexico_Trending | 135 | 865 | 0.156069 |
| 9 | Russia_Trending | 123 | 877 | 0.140251 |
| 10 | USA_NonTrending | 703 | 297 | 2.367003 |
| 11 | Canada_NonTrending | 564 | 436 | 1.293578 |
| 12 | Denmark_NonTrending | 613 | 387 | 1.583979 |
| 13 | France_NonTrending | 546 | 454 | 1.202643 |
| 14 | GreatBritain_NonTrending | 601 | 399 | 1.506266 |
| 15 | India_NonTrending | 852 | 148 | 5.756757 |
| 16 | Japan_NonTrending | 93 | 907 | 0.102536 |
| 17 | SouthKorea_NonTrending | 63 | 937 | 0.067236 |
| 18 | Mexico_NonTrending | 246 | 754 | 0.326260 |
| 19 | Russia_NonTrending | 184 | 816 | 0.225490 |

Hover over the plots and switch table to play with them and see what each one represents!! Use the toolbar to crop out some plots for better analysis. The first plot visualizes the ratio
negative tags while the second one shows the actual values that were classified as positive and negative by the sentiment analysis classifier.

In [15]:

```
## written by Aravind Patnam
script.do_sentiment_analysis_visualization(data)
```

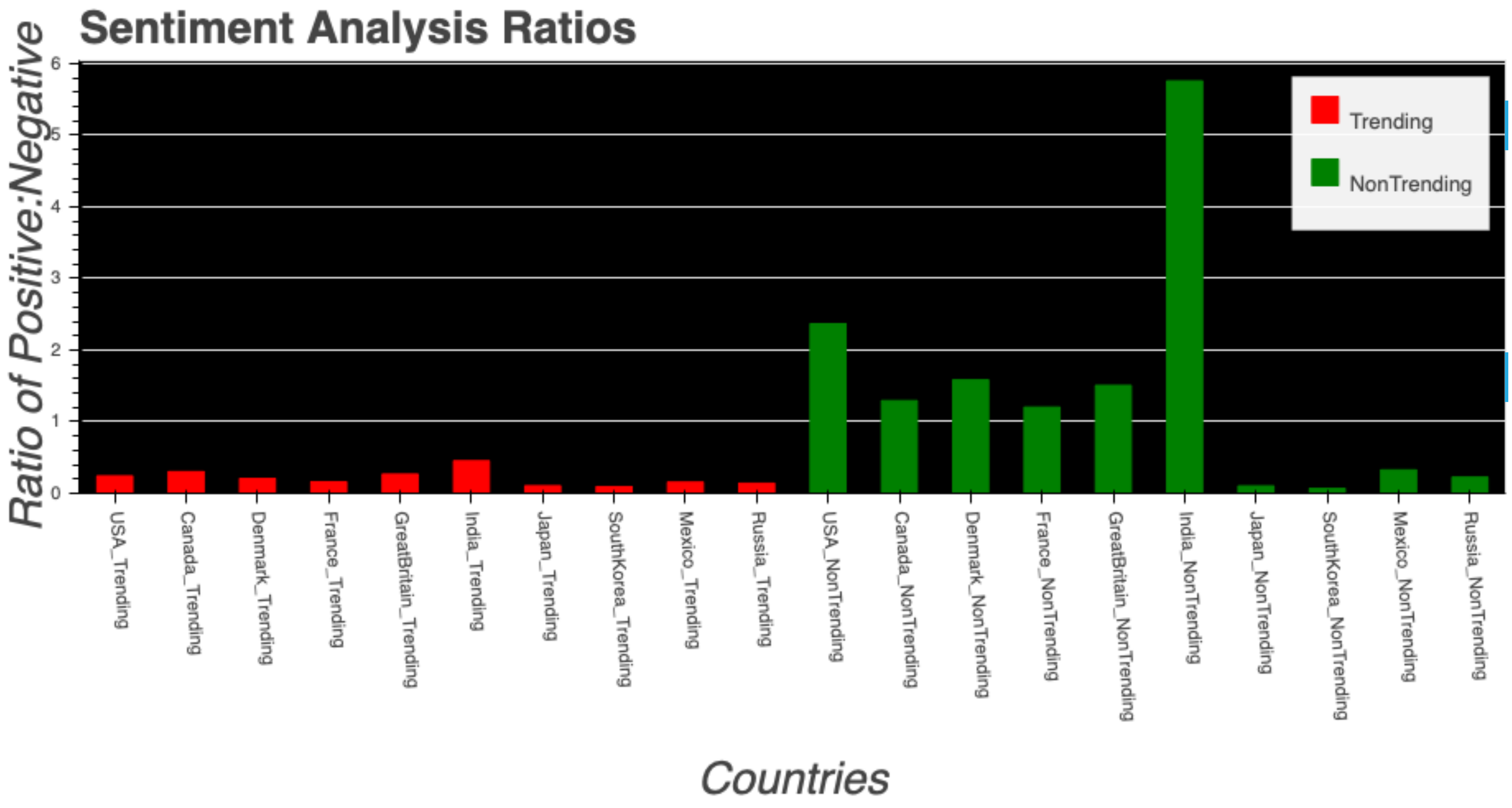
(https://bokeh.org) Loading BokehJS ...

(https://bokeh.org) BokehJS 1.4.0 successfully loaded.

Sentiment Analysis Ratios | Positives vs. Negatives

(http

Sentiment Analysis Ratios | Positives vs. Negatives



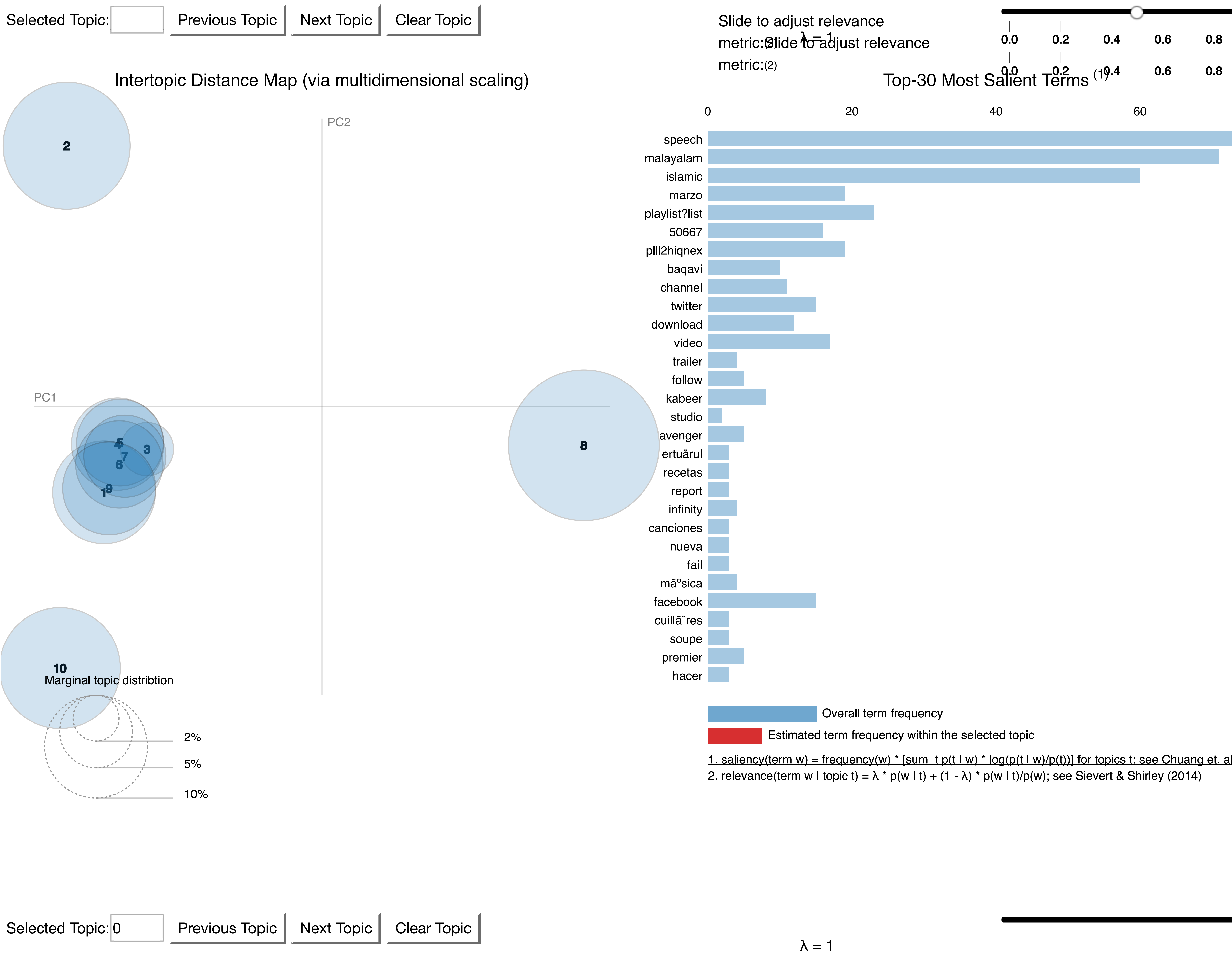
(http

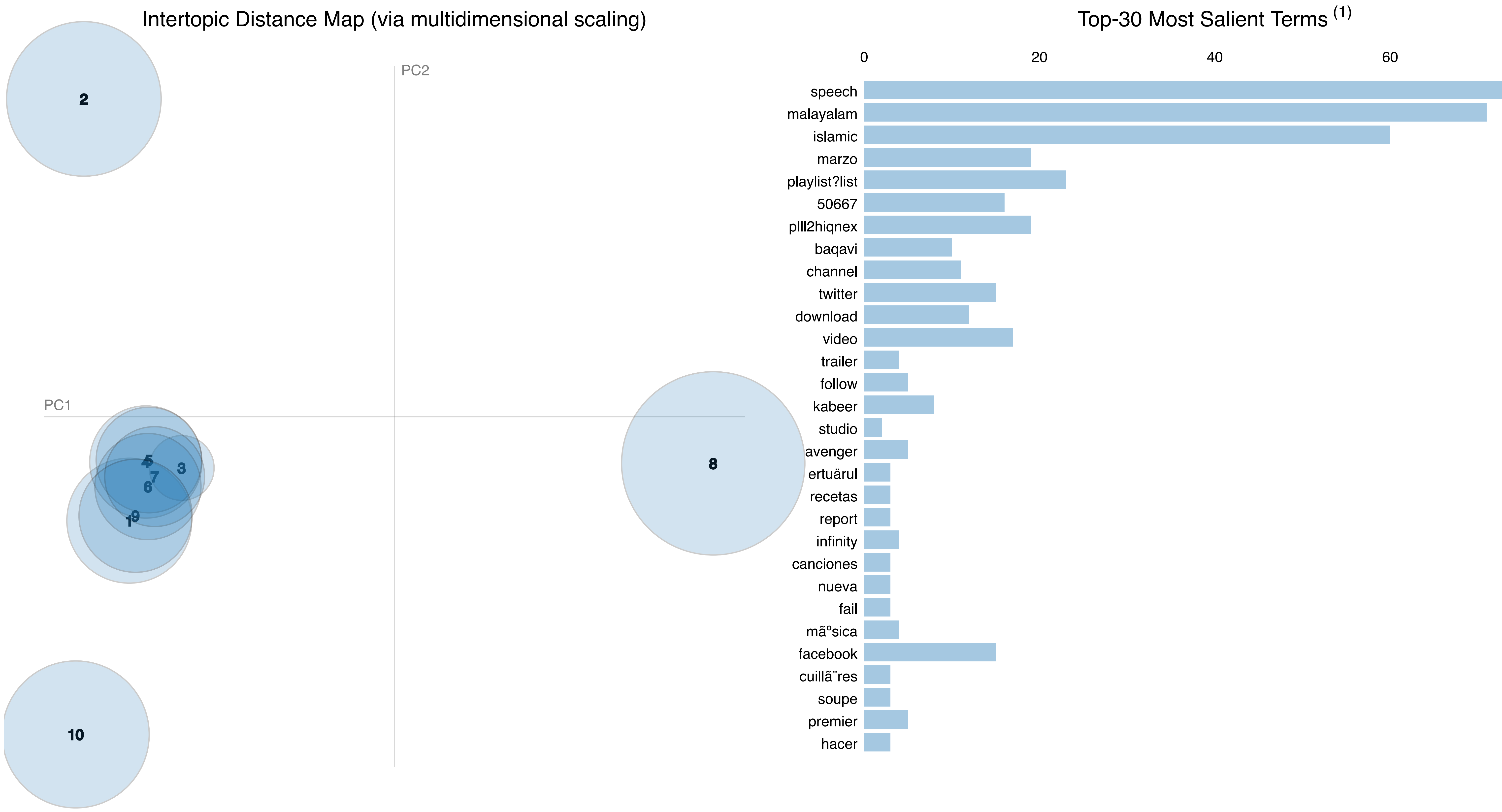
Hover and play around with the principle component analysis presented below. LDA topic model was used on youtube tags and descriptions to show these findings. The relevancy m
submission is set to around 0.30 since it showed the most favorable results that were not too specific and not too generic.

In [82]:

```
## written by Aravind Patnam

## this will take a long time to run!

## visualize trending videos using pyLDAvis -> this might be only visible on nbviewer depending on your notebook viewing settings

full_trending_lda_input = list(script.get_most_common_tags(full_trending_df).keys()) + list (full_trending_df.sample(16901)['descriptio
topics_Full_Trending, corpus, dictionary = script.do_LDA(full_trending_lda_input)
lda_display = script.visualize_LDA(True, corpus, dictionary)
pyLDAvis.display(lda_display)
```

Out[82]:

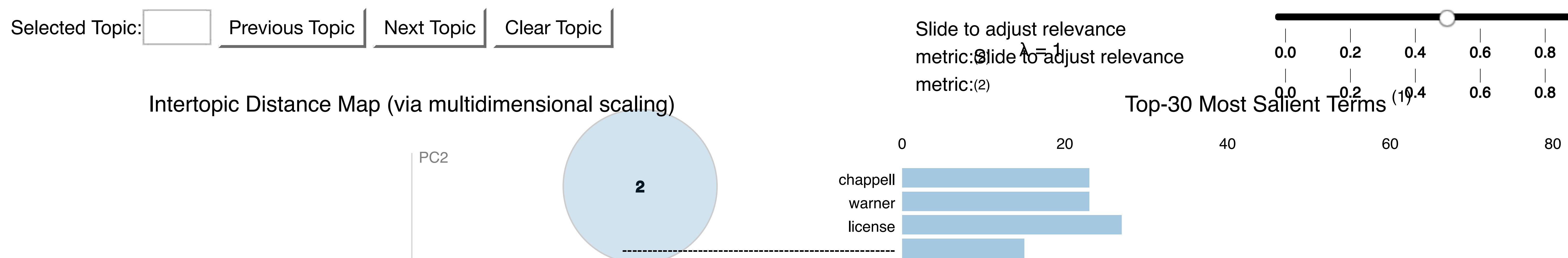


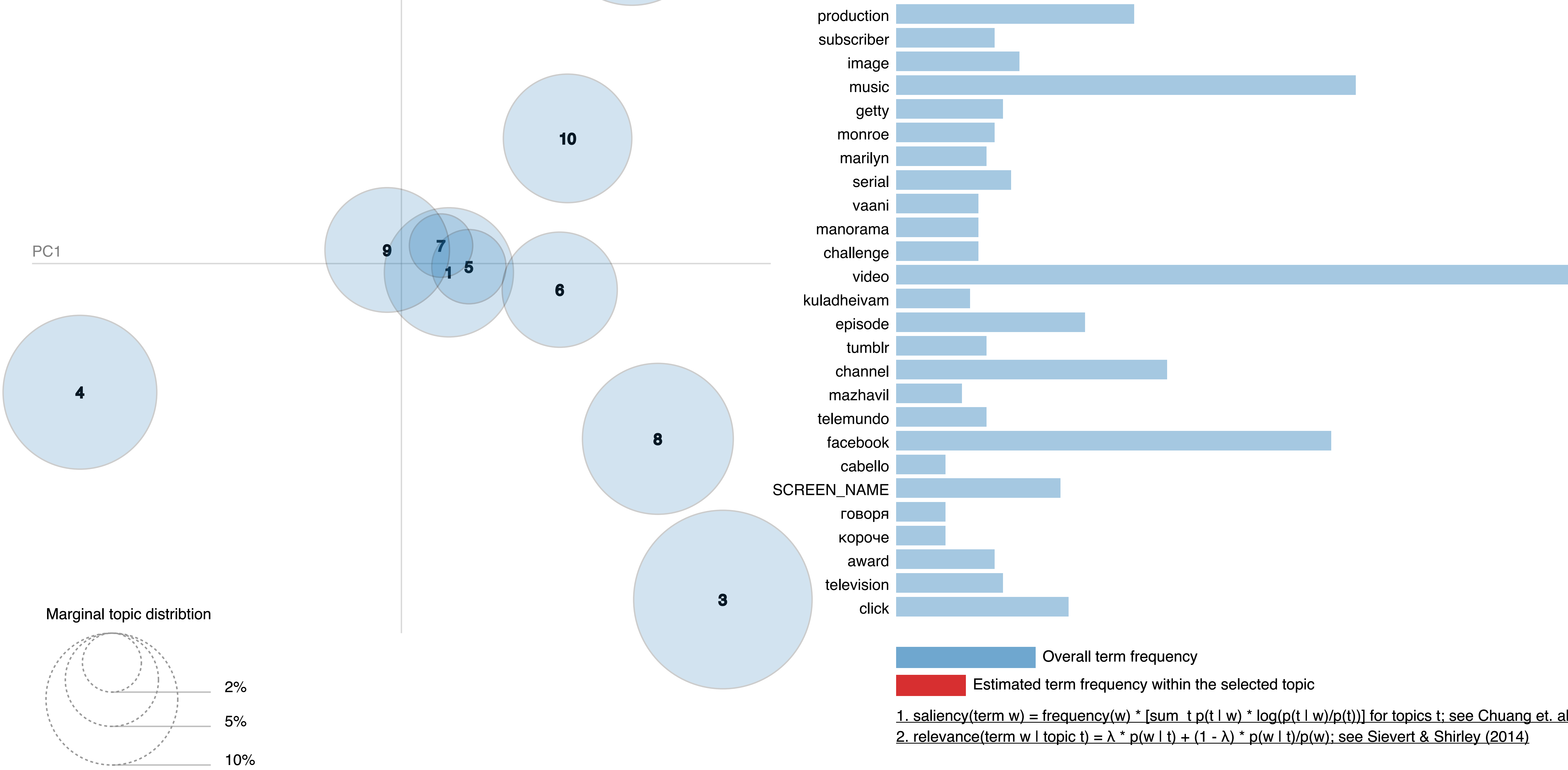


In [83]:

```
## written by Aravind Patnam

## this will take a long time to run!

## visualize nontrending videos using pyLDAvis -> this might be only visible on nbviewer depending on your notebook viewing settings


full_nontrending_lda_input = list(script.get_most_common_tags(full_nontrending_df).keys()) + list (full_nontrending_df['description'])
topics_Full_Nontrending, corpus, dictionary = script.do_LDA(full_nontrending_lda_input)
lda_display = script.visualize_LDA(True, corpus, dictionary)
pyLDAvis.display(lda_display)
```

Out[83]:


Selected Topic:
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance
metric:(3)lide to adjust relevance
metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
0.0
0.2
0.4
0.6
0.8
Intertopic Distance Map (via multidimensional scaling)
PC2
2
Top-30 Most Salient Terms (1)
0
20
40
60
80
chappell
warner
license

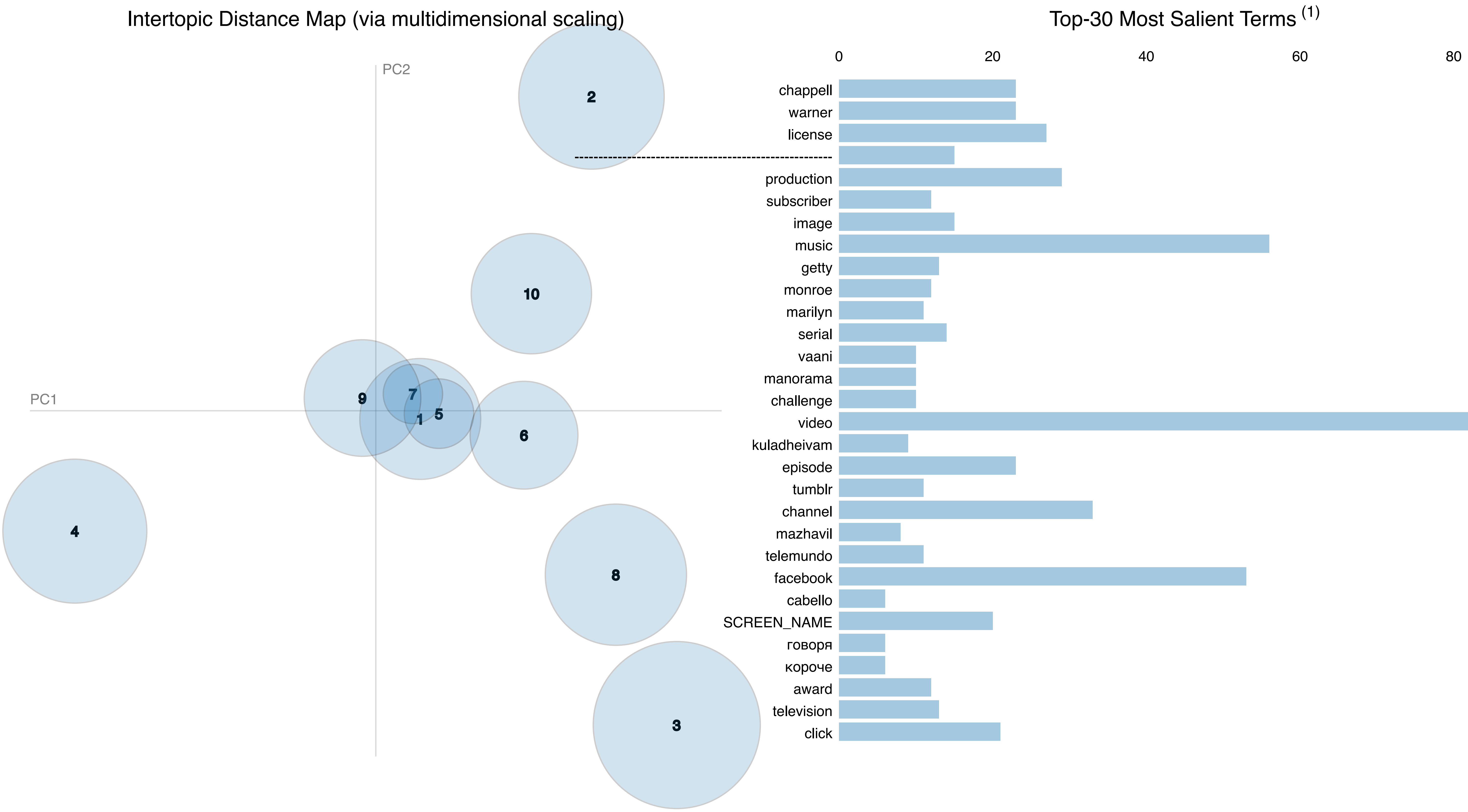


1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al

2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Selected Topic: 0 Previous Topic Next Topic Clear Topic

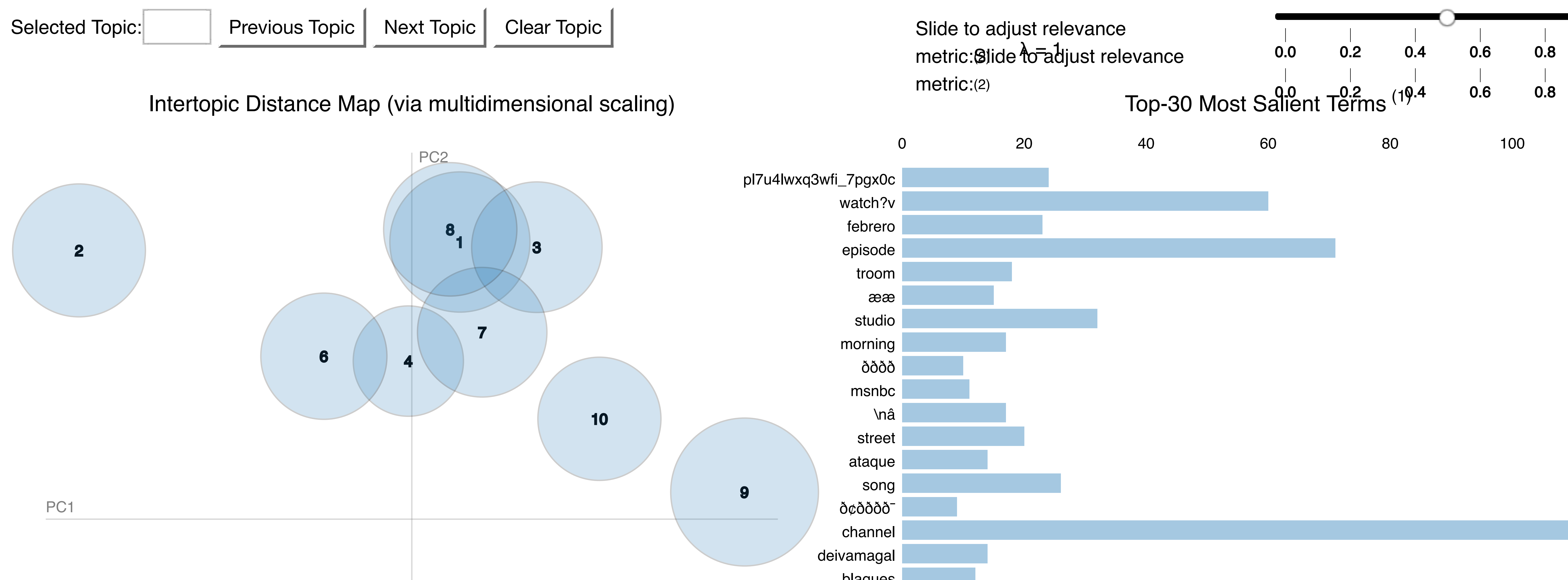
λ = 1

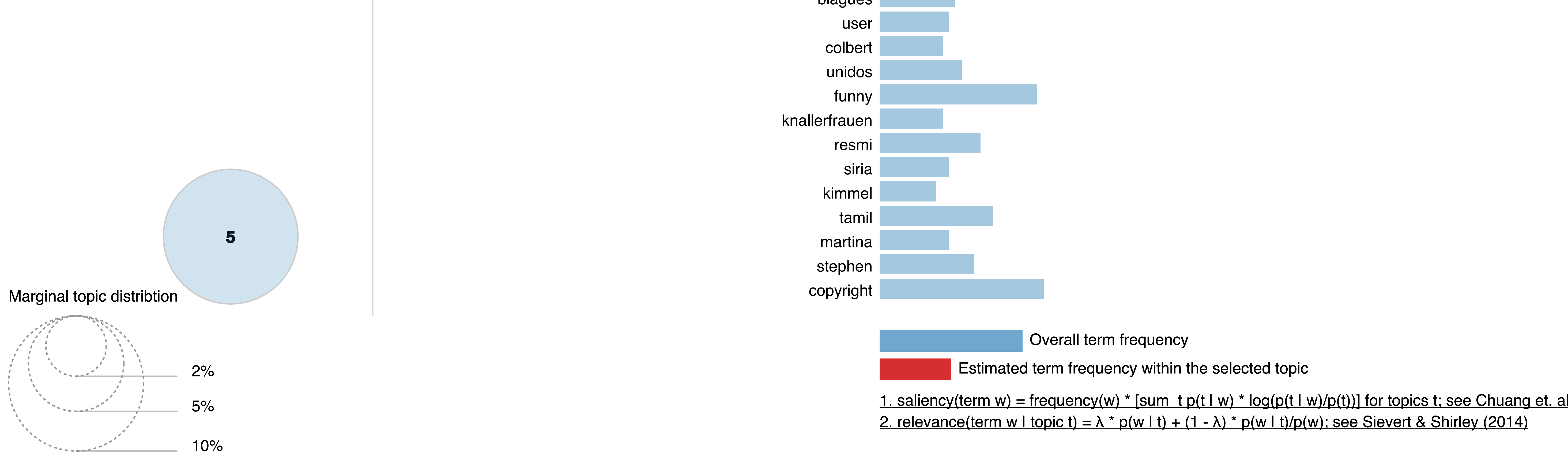


In [147]:

```
## written by Aravind Patnam

## this will take a long time to run!

## visualize all videos using pyLDAvis -> this might be only visible on nbviewer depending on your notebook viewing settings

allDfsDf_lda_input = list(script.get_most_common_tags(allDfsDf).keys()) + list (allDfsDf.sample(100000)['description'])
topics_all_dfs, corpus, dictionary = script.do_LDA(allDfsDf_lda_input)
lda_display = script.visualize_LDA(True, corpus, dictionary)
pyLDAvis.display(lda_display)
```

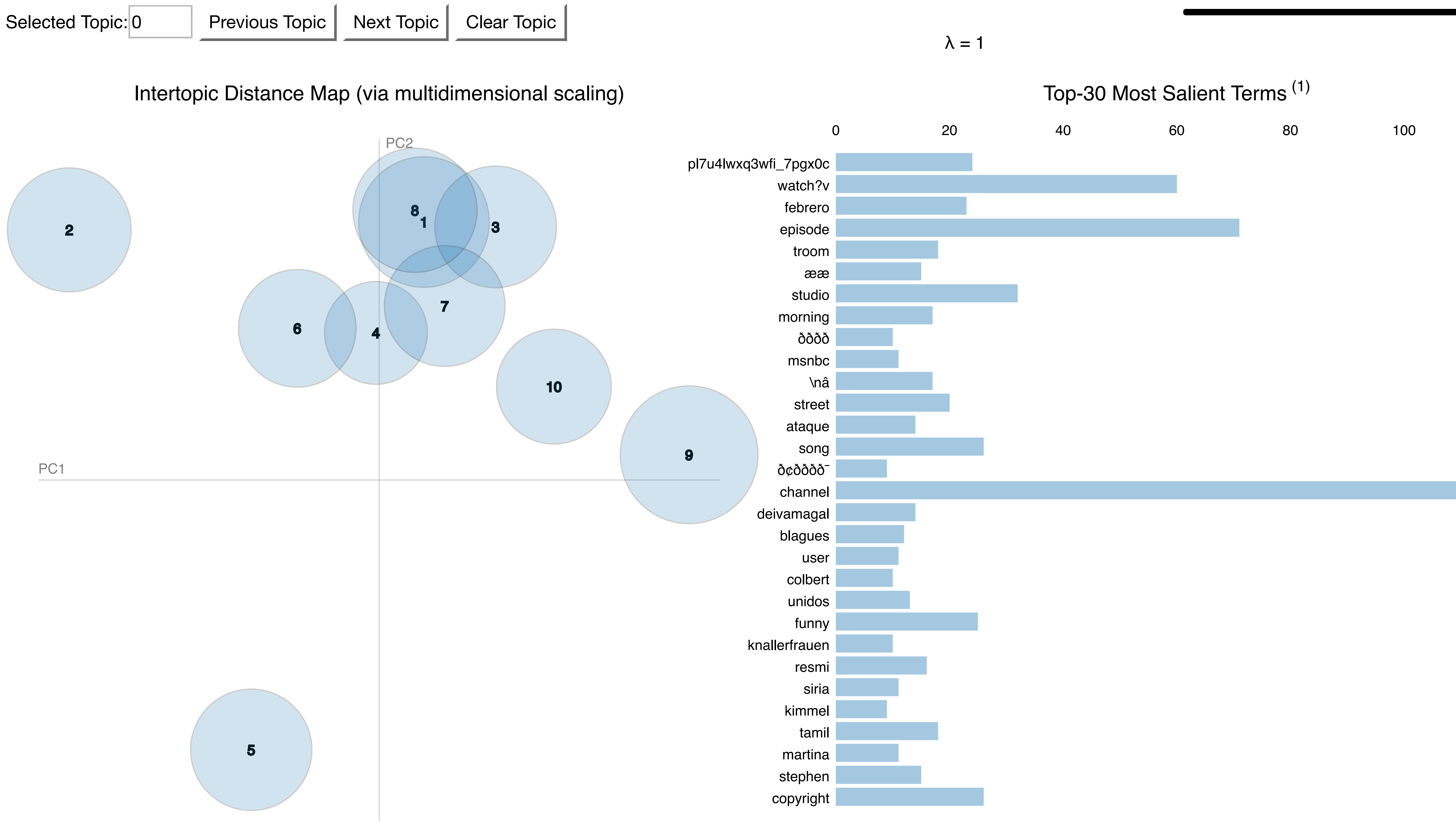
Out[147]:





1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al

2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

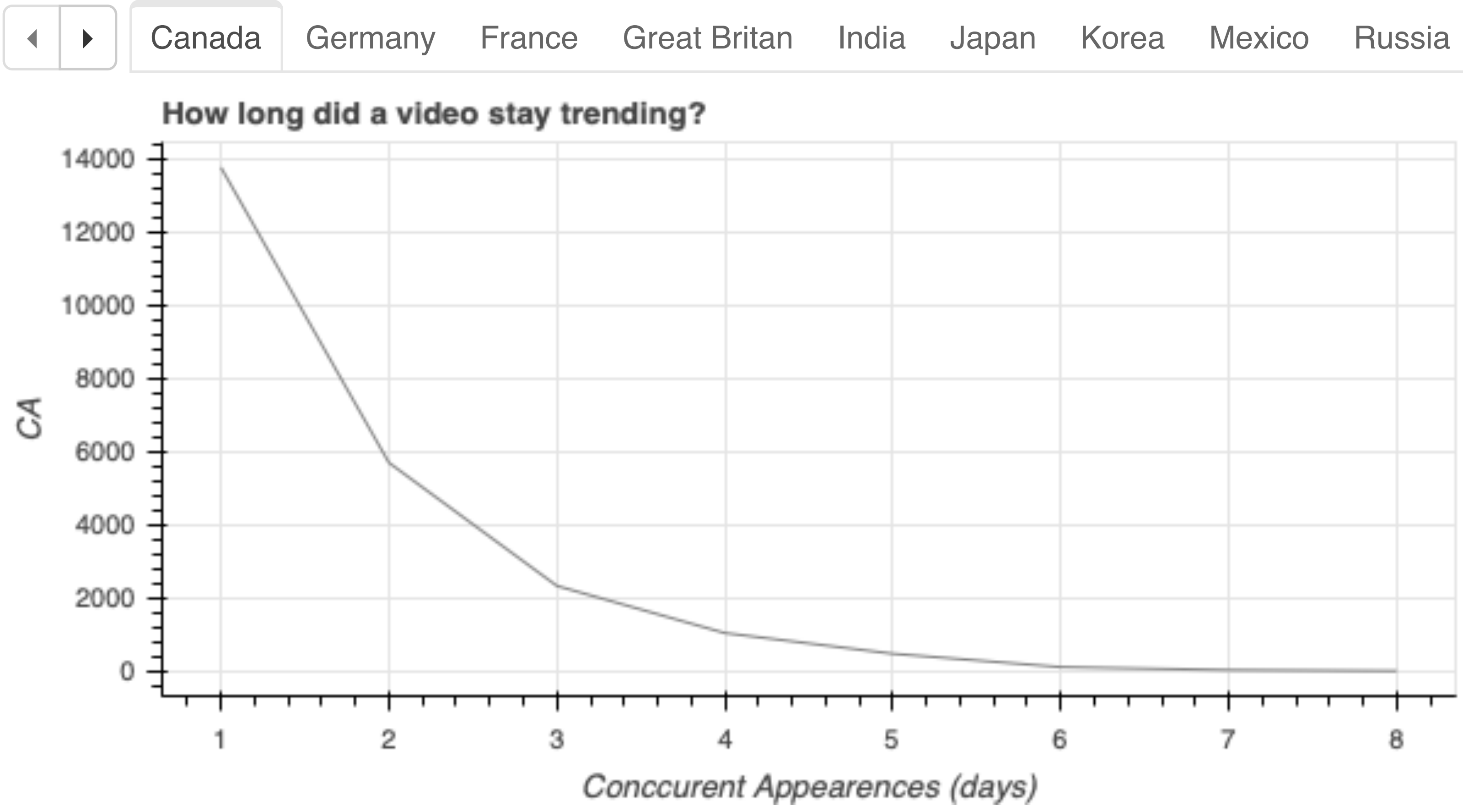


In [16]:

```
# Written by Jeremy Tan
script.makeVideoTrending(full_trending_df, full_trending_df_fill)
```

(https://bokeh.org) BokehJS 1.4.0 successfully loaded.

◀ ▶ Canada Germany France Great Britan India Japan Korea Mexico Russia



As you can see from the visualization above, most trending videos only stay trending for two days or less. Some notable outliers are videos from Great Britan, which has videos that v
days! For countries in the far East, videos only seem to stay tredning for one day so the turnover rate is pretty high.

In [17]:

```
# Written by Jeremy Tan
# Makes two new columns that count tilte length and description length ands changes comment_count to comments
full_trending_df['title_length'] = full_trending_df['title'].str.len()
full_trending_df['description_length'] = full_trending_df['description'].str.len()
full_trending_df = full_trending_df.rename(columns={'comment_count':'comments'})
```

In [18]:

```
# Written by Jeremy Tan
# What is the correlation between views, likes, disklies, and comment count?
script.makeScatter(full_trending_df)
```

As seen from the scatter matrix, there is a somewhat strong correlation between views and likes. Another correlation is between likes and comments, views and comments, and dislik
somewhat weak!). Another interesting point is the length of the description and the length of the title seem to be tied together.

In [19]:

```
# What is the correlation between views, likes, disklies, and comment count?
# Another interactive visualization
corr = full_trending_df.loc[:, ['views', 'likes', 'dislikes', 'comments', 'description_length', 'title_length']].corr()
script.makeHeatMap(corr)
```

Another way to visualzie the correlations. Here, I can see additional correlations of dislikes and views, dislikes and likes, and dislikes and comments.

In [20]:

```
# Written by Jeremy Tan
# What is the correlation between views, likes, disklies, and comment count in categories?
categories = full_trending_df['category'].unique()
corr_list = [full_trending_df[full_trending_df['category'] == cat].loc[:, ['views', 'likes', 'dislikes', 'comments', 'description_lengt
orr() for cat in categories]
script.makeCategoryHeatMap(corr_list, categories)
```

Another way to visualize correlations but now based on categories. For the popular categories, the correlations I stated earlier hold true. However, if you were to go to a unpopular ca correlations appear in place of the previous, strong correlations. Most suprisingly, for categories that elict human emotion, such as "Pet & Animals" and "Nonprofits & Activism" there between likes, views, comments, and dislikes.

In [21]:

```
# Written by Jeremy Tan
# What category of videos trend the most in which countries?
# What category of videos fail to hit trending in which countries?
```

In [22]:

```
script.trendingCategories(US_trending_df, "United States")
```

In [23]:

```
script.nontrendingCategories(not_trending_us_df, "United States")
```

Based on the two plot above, one can see "Entertainment" videos are the majority of videos made. They, however, have the highest chance to both fail and succeed. The pattern illun the category, the more videos that are being put out.

```
In [24]:
```

```python
script.trendingCategories(CA_trending_df, "Canada")
```

```
In [25]:
```

```python
script.nontrendingCategories(not_trending_ca_df, "Canada")
```

The same pattern seems to appear as it did in the United States. However, "People and Blogs" and "Music" seem to fail more in this country.

In [26]:

```
script.trendingCategories(DE_trending_df, "Germany")
```

In [27]:

```
script.nontrendingCategories(not_trending_de_df, "Germany")
```

The same pattern seems to appear as it did in the United States. However, "People and Blogs" have more of a success rate but "Films & Animation" seem to fail more.

In [28]:

```
script.trendingCategories(FR_trending_df, "France")
```

In [29]:

```
script.nontrendingCategories(not_trending_fr_df, "France")
```

Same pattern occurs like in the Untied States. An interesting point is that "Peoples & Blog" have a higher probability of being trending. We can assume this is due to people valuing th

In [30]:

```
script.trendingCategories(GB_trending_df, "Great Britan")
```

In [31]:

```
script.nontrendingCategories(not_trending_gb_df, "Great Britan")
```

Same pattern occurs like in the Untied States. However, "Music" dominates this country with the majority of the videos being produced are of this category.

In [32]:

```
script.trendingCategories(IN_trending_df, "India")
```

In [33]:

```
script.nontrendingCategories(not_trending_in_df, "India")
```

Same pattern as the US. Nothing interesitng to note as it follows the pattern exactly where the more successful a category, the more likely the video will also fail.

In [34]:

```
script.trendingCategories(JP_trending_df, "Japan")
```

In [35]:

```
script.nontrendingCategories(not_trending_jp_df, "Japan")
```

Similar pattern as Germany, where "People and Blogs" videos will have a higher chance to be tredning and not fail.

In [36]:

```
script.trendingCategories(KR_trending_df, "Korea")
```

In [37]:

```
script.nontrendingCategories(not_trending_kr_df, "Korea")
```

Same pattern as the US.Interesting enough, this country cares alot about "News & Politics" and have lots of video that hit trending (only second behind "Entertainment").

In [38]:

```
script.trendingCategories(MX_trending_df, "Mexico")
```

In [39]:

```
script.nontrendingCategories(not_trending_mx_df, "Mexico")
```

Same pattern as the US. Similar stats as the US too.

In [40]:

```
script.trendingCategories(RU_trending_df, "Russia")
```

In [41]:

```
script.nontrendingCategories(not_trending_ru_df, "Russia")
```

Same pattern as the US. However, like Germany and Japan, "Peoples & Blogs" have a high succes rate to reach trending and fail less.

In [ ]:

```
# Written by Jeremy Tan
# Which categories are the most popular? Do highest average amongst likes and views ----> shows which type of videos people enjoy the m
# makes a box plot to visuzlaize distbution of likes and views
```

In [42]:

```
script.likes_to_categories(US_trending_df, "United States", 'likes_log')
```

In [43]:

```
script.likes_to_categories(US_trending_df, "United States", 'views_log')
```

Judging from the boxplot, it seems people in the US enjoy videos categorized Music more the most. Gaming is the second most popualr. Then, Entertainment.

In [44]:

```
script.likes_to_categories(CA_trending_df, "Canada", 'likes_log')
```

In [45]:

```
script.likes_to_categories(CA_trending_df, "Canada", 'views_log')
```

It isn't as clear, but there seems to be a tie between Movies and Music as some of the more popular videos in Canada. Comedy lags behind third.

In [46]:

```
script.likes_to_categories(DE_trending_df, "Germany", 'likes_log')
```

In [47]:

```
script.likes_to_categories(DE_trending_df, "Germany", 'views_log')
```

In Germany, Music is the most popular category. Movies trail right behind, and Comedy right after.

In [48]:

```
script.likes_to_categories(FR_trending_df, "France", 'likes_log')
```

In [49]:

```
script.likes_to_categories(FR_trending_df, "France", 'views_log')
```

In France, Music is the most popular category like Germany. However, Comedy and Entertainment trail right behind.

In [50]:

```
script.likes_to_categories(GB_trending_df, "Great Britan", 'likes_log')
```

In [51]:

```
script.likes_to_categories(GB_trending_df, "Great Britan", 'views_log')
```

Music is the most popular category, while Nonprfits & Activism then Entertainment trail behind.

In [52]:

```
script.likes_to_categories(IN_trending_df, "India", 'likes_log')
```

In [53]:

```
script.likes_to_categories(IN_trending_df, "India", 'views_log')
```

It isn't clear which videos come out on top, but based on likes, Pets & Animals, Gaming, and then Comedy are the top categories.

In [54]:

```
script.likes_to_categories(JP_trending_df, "Japan", 'likes_log')
```

In [55]:

```
script.likes_to_categories(JP_trending_df, "Japan", 'views_log')
```

Funny enough, Science & Technology are the most popular videos follwoed by Music and then Comedy.

In [56]:

```
script.likes_to_categories(KR_trending_df, "Korea", 'likes_log')
```

In [57]:

```
script.likes_to_categories(KR_trending_df, "Korea", 'views_log')
```

Similar to Japan, Science & Technology is first as Musis is second. However, Sports is third.

In [58]:

```
script.likes_to_categories(MX_trending_df, "Mexico", 'likes_log')
```

```
In [59]:
script.likes_to_categories(MX_trending_df, "Mexico", 'views_log')
```

Music is first, Gaming is second, and Comedy is third.

In [60]:

```
script.likes_to_categories(RU_trending_df, "Russia", 'likes_log')
```

In [61]:

```
script.likes_to_categories(RU_trending_df, "Russia", 'views_log')
```

Music is first Science and Technology is second. Comedy is third.

In [62]:

```
# Written by Jeremy Tan
# Which channels are the most successfucl at reaching trending?
# Grabs most reoccuring videos in trending
```

In [63]:

```
script.videos_top(US_trending_df, "United States")
```

In [64]:

```
script.videos_top(CA_trending_df, "Canada")
```

In [65]:

```
script.videos_top(DE_trending_df, "Germany")
```

In [66]:

```
script.videos_top(FR_trending_df, "France")
```

In [67]:

```
script.videos_top(GB_trending_df, "Great Britan")
```

In [68]:

```python
script.videos_top(IN_trending_df, "India")
```

In [69]:

```
script.videos_top(JP_trending_df, "Japan")
```

Original scrapper did not properly encode characters correctly. Hence the weird symbols.

In [70]:

```
script.videos_top(KR_trending_df, "Korea")
```

Original scrapper did not properly encode characters correctly. Hence the weird symbols.

In [71]:

```
script.videos_top(MX_trending_df, "Mexico")
```

In [72]:

```
script.videos_top(RU_trending_df, "Russia")
```

Original scrapper did not properly encode characters correctly. Hence the weird symbols.

In [73]:

```
# Written by Jeremy Tan
# When a video gets published, what is the intial like rate that got them to trending?
# Has two plots: one shows which hour a video is most commonly published and the other is a boxen plot that shows differnet quartiles o
iers
script.showHours(full_trending_df)
```

It seems most trending videos are published at 4 pm. However, more engagemnt in terms of like happens at 8pm.

In [74]:

```
full_nontrending_df.head()
```

Out[74]:

|  | Unnamed: 0 | Unnamed: 0.1 | video_id | title | category | channel_title | category_id | publish_time | tags | views | likes | dislikes | co |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | JlqbeidXvK0 | Ryan Reynolds Got High-Fived At The Worst Poss... | Entertainment | The Graham Norton Show | 24 | 2018-05-11T15:19:21.000Z | Joan Armatrading\|Graham Norton new\|the Graham ... | 4411805 | 53204 | 538 |  |
| 1 | 1 | 1 | T06oh88VhiE | Red Beans and Rice - Creole-Style Spicy Red Be... | Howto & Style | Food Wishes | 26 | 2015-02-17T00:58:32.000Z | Red Beans And Rice (Food)\|Louisiana Creole Cui... | 1035084 | 20271 | 442 |  |
| 2 | 2 | 2 | yMVA3RNiTE8 | United State of Pop \| DJ Earworm Mashup 2017 | People & Blogs | Mashup Songs | 22 | 2017-01-15T16:58:11.000Z | dj earworm 2016 2017\|mashup 2017\|United State ... | 180529 | 1232 | 215 |  |
| 3 | 3 | 3 | WiinVuzh4DA | i love you | Music | Billie Eilish - Topic | 10 | 2019-03-28T10:07:29.000Z | Billie Eilish ビリー・アイリッシュ ビリーアイリッシュ WHEN WE ALL... | 25833526 | 339090 | 9882 |  |
| 4 | 4 | 4 | u3wUZw9S2PM | The Black Eyed Peas - The APL Song (Official M... | Music | BlackEyedPeasVEVO | 10 | 2009-12-25T04:20:22.000Z | Black\|Eyed\|Peas\|Interscope\|The Black Eyed Peas... | 13002879 | 45974 | 1380 |  |

In [ ]:

```
# Written by Jeremy Tan
# Grabs the videos that have the most views, likes, or dislikes
# What videos have the most views, likes, and dislikes in the tredning dataset and nontrending dataset?
```

In [75]:

```
script.visualize_most(full_trending_df, "views")
```

In [76]:

```
script.visualize_most(full_nontrending_df, "views")
```

In [77]:

```
script.visualize_most(full_trending_df, "likes")
```

In [78]:

```
script.visualize_most(full_nontrending_df, "likes")
```

In [79]:

```
script.visualize_most(full_trending_df, "dislikes")
```

In [80]:

```
script.visualize_most(full_nontrending_df, "dislikes")
```

In [81]:

```
# Written by Jeremy Tan
# Which country has the most active participation and engagemet?
script.engagement(full_trending_df)
```

Overall, Great Britan has the most active audience with the US seriously lagging behind.