Clustering Algorithms in Python

Trend in India, Pakistan and Philippine from Google Search Keywords

Hyerim Hwang

To whom might be interested in both countries to promote their products or services

Table of Contents

Introduction

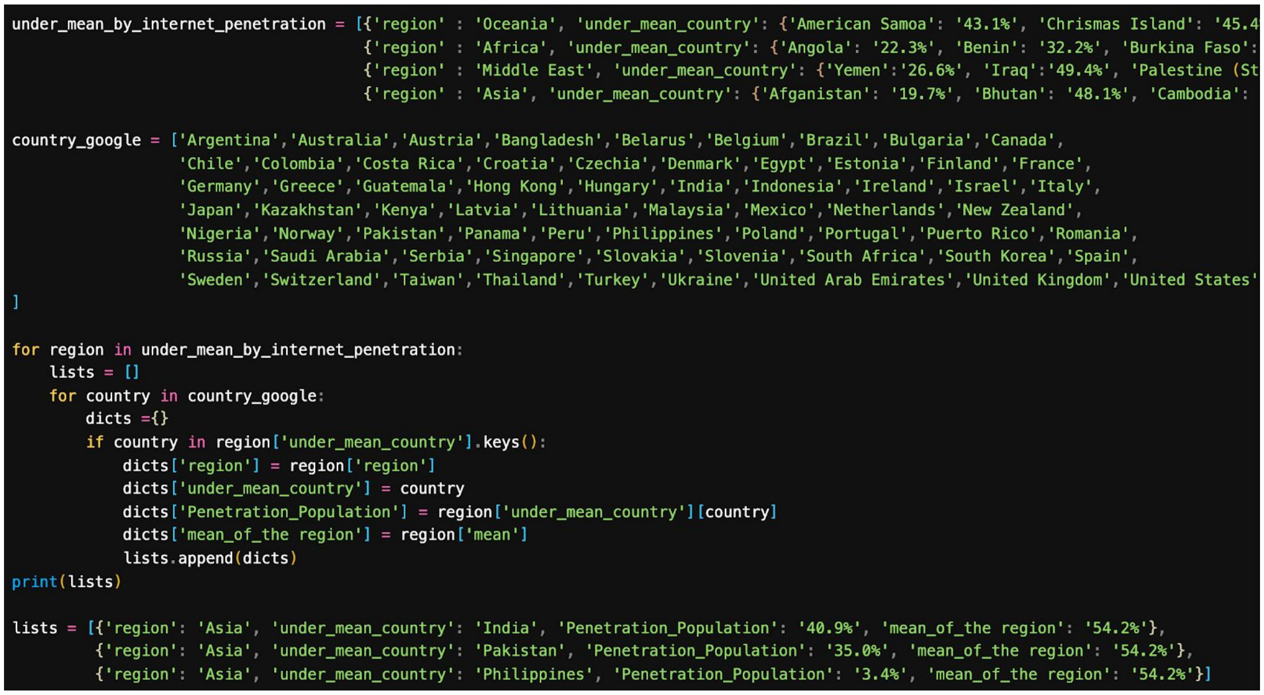
Google is the most commonly used search engine in the world across all platforms, “with more than 5.4 billion searches each day and 78,881 searches in 1 second" (Google Search Statistics). Google Trend, also, offers an well-structured information within the limit of 1 - 30 Categories. There are top 10 popular keywords per each category, and lists on each year by each country from 2001 until now. The initial trends per each country are expected to see and capture each historical interests or big events over time.

The report tried to capture the global trends, however, Global dataset in Google trends does not support China and three Higher internet usage countries in Asia don’t use Google as a primary search engine (Internet Stats & Facts for 2019). Due to these reasons, the report decided to rather look up the regions that have a lower broadband penetration rate. The lower broadband penetration rate will be a great indicator to find out a less compatible market for the search engine, so it will lead to get more prominent results by using the Google search dataset to see those region’s overall trends.

Methodology

Choosing data - India[1], Pakistan[2], Philippine[3]

To capture the regions' broadband penetration rates, the internet world statistics dataset was available based on the research of 4,536,248,808 internet users on Jun 30, 2019. The dataset below (Figure. 1) contained three different variables. At first, there's some country's broadband penetration rates that were lower than the average penetration rates in each region. Also, the country naming list of Google Trends were added to compare the regions' data that were lower penetration rates with it. Then, the ideal target countries came out with the name of India, Pakistan and the Philippines containing with each rate of internet penetration rate per each country.



```
under_mean_by_internet_penetration = [{'region' : 'Oceania', 'under_mean_country': {'American Samoa': '43.1%', 'Chrismas Island': '45.4
                                      {'region' : 'Africa', 'under_mean_country': {'Angola': '22.3%', 'Benin': '32.2%', 'Burkina Faso':
                                      {'region' : 'Middle East', 'under_mean_country': {'Yemen':'26.6%', 'Iraq':'49.4%', 'Palestine (St
                                      {'region' : 'Asia', 'under_mean_country': {'Afganistan': '19.7%', 'Bhutan': '48.1%', 'Cambodia':

country_google = ['Argentina','Australia','Austria','Bangladesh','Belarus','Belgium','Brazil','Bulgaria','Canada',
                  'Chile','Colombia','Costa Rica','Croatia','Czechia','Denmark','Egypt','Estonia','Finland','France',
                  'Germany','Greece','Guatemala','Hong Kong','Hungary','India','Indonesia','Ireland','Israel','Italy',
                  'Japan','Kazakhstan','Kenya','Latvia','Lithuania','Malaysia','Mexico','Netherlands','New Zealand',
                  'Nigeria','Norway','Pakistan','Panama','Peru','Philippines','Poland','Portugal','Puerto Rico','Romania',
                  'Russia','Saudi Arabia','Serbia','Singapore','Slovakia','Slovenia','South Africa','South Korea','Spain',
                  'Sweden','Switzerland','Taiwan','Thailand','Turkey','Ukraine','United Arab Emirates','United Kingdom','United States'
]

for region in under_mean_by_internet_penetration:
    lists = []
    for country in country_google:
        dicts ={}
        if country in region['under_mean_country'].keys():
            dicts['region'] = region['region']
            dicts['under_mean_country'] = country
            dicts['Penetration_Population'] = region['under_mean_country'][country]
            dicts['mean_of_the region'] = region['mean']
            lists.append(dicts)
print(lists)

lists = [{'region': 'Asia', 'under_mean_country': 'India', 'Penetration_Population': '40.9%', 'mean_of_the region': '54.2%'},
         {'region': 'Asia', 'under_mean_country': 'Pakistan', 'Penetration_Population': '35.0%', 'mean_of_the region': '54.2%'},
         {'region': 'Asia', 'under_mean_country': 'Philippines', 'Penetration_Population': '3.4%', 'mean_of_the region': '54.2%'}]
```

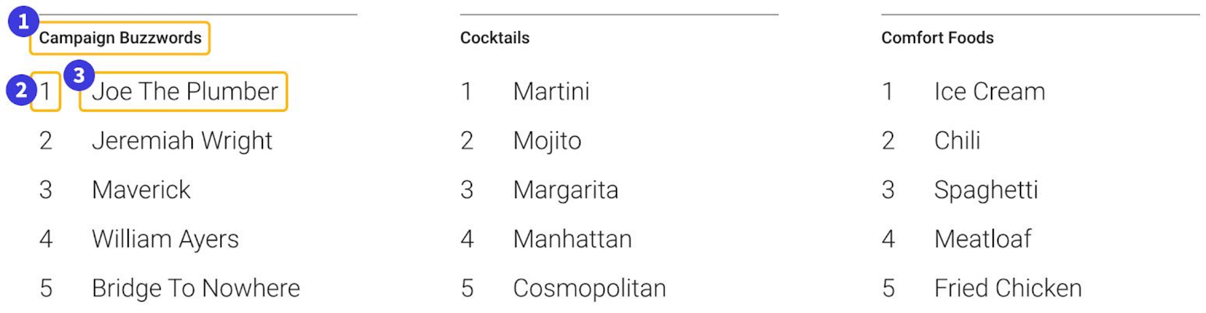
Figure. 1: Combining two dataset to get the target countries with lower broadband penetration rates.

[1] https://trends.google.com/trends/yis/2018/IN/
[2] https://trends.google.com/trends/yis/2018/PK/
[3] https://trends.google.com/trends/yis/2018/PH/

Since the datasets are India, Pakistan and the Philippines, the report tried to narrow down to see the correlation and difference between India and Pakistan throughout their each keyword. So the following step was building hypotheses that are

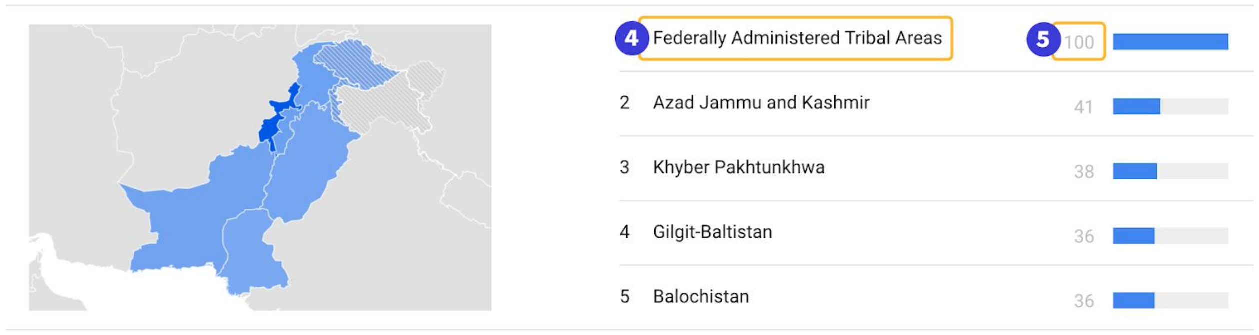
- Expect to see any changes in trends in each region over time
- Expect to see correlation or difference between the two countries over time

I began to collect the data in Google Trend with each year's search keywords and ranking (Figure. 2). It also required to gather the keyword's href attribute as well to see all of the search volumes by subregion that was located on each keyword's content page (Figure. 3).



Campaign Buzzwords		Cocktails		Comfort Foods	
1	Joe The Plumber	1	Martini	1	Ice Cream
2	Jeremiah Wright	2	Mojito	2	Chili
3	Maverick	3	Margarita	3	Spaghetti
4	William Ayers	4	Manhattan	4	Meatloaf
5	Bridge To Nowhere	5	Cosmopolitan	5	Fried Chicken

Figure. 2: The homepage contains a list that has 1: category | 2: ranking | 3: keyword.



	Federally Administered Tribal Areas	100
2	Azad Jammu and Kashmir	41
3	Khyber Pakhtunkhwa	38
4	Gilgit-Baltistan	36
5	Balochistan	36

Figure. 3: The content page contains 4: a name of subregion | 5: search volumes.

Collecting data - Selenium[4]

The datasets with multiple pages were needed to use web scraping method, so the Selenium library in Python was selected which allows to click on “show 5 more” buttons by an automatic bot based on my code (Appendix 1). To analyze the data based on the robust basement, I built the MongoDB cluster and database to store each year’s keyword and other information on each collection. Inside of the collection, I stored the data of each keyword including the data of ranking in each year, search keyword grouping by category, search volumes by region.

Cleaning data - Pandas[5] and Regular Expression[6]

The dataset are huge so applying pandas library was a primary tool to clean and recreate data set for fitting to analyze the data, and also utilized the regular expression library for potential overlapping keyword in each country - Split the keywords into words and lowercase the words and remove punctuation (Figure. 6).

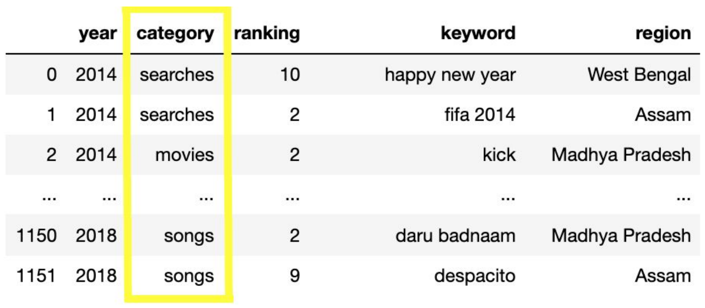
```
import re

# Remove punctuation
table['keyword'] = table['keyword'].map(lambda x: re.sub('[,\.!?]', '', x))
table['category'] = table['category'].map(lambda x: re.sub('[,\.!?]', '', x))
# Convert the titles to lowercase
table['keyword'] = table['keyword'].map(lambda x: x.lower())
table['category'] = table['category'].map(lambda x: x.lower())
```

Figure. 6: Remove punctuation and convert to lower case by Regular Expression

Preparing data for Clustering - CountVectorizer[7]

Each keyword already has each classified label, using the existing category label (Figure. 7) was selected for finding the trend over time. However, Each year and each country has its own criteria so it would better to create a new label for clustering.

[4] https://selenium-python.readthedocs.io/
[5] https://pandas.pydata.org/pandas-docs/stable/
[6] https://docs.python.org/3/library/re.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html



	year	category	ranking	keyword	region
0	2014	searches	10	happy new year	West Bengal
1	2014	searches	2	fifa 2014	Assam
2	2014	movies	2	kick	Madhya Pradesh
...	...	...	...	...	...
1150	2018	songs	2	daru badnaam	Madhya Pradesh
1151	2018	songs	9	despacito	Assam

Figure. 7: The existing category labels indicate classifications of each keyword.

Thus, I applied the CountVectorizer (Appendix. 2) and matplotlib that provides a tokenized collection of text documents and returned a chart (Figure. 8) with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document (How to Prepare Text Data).

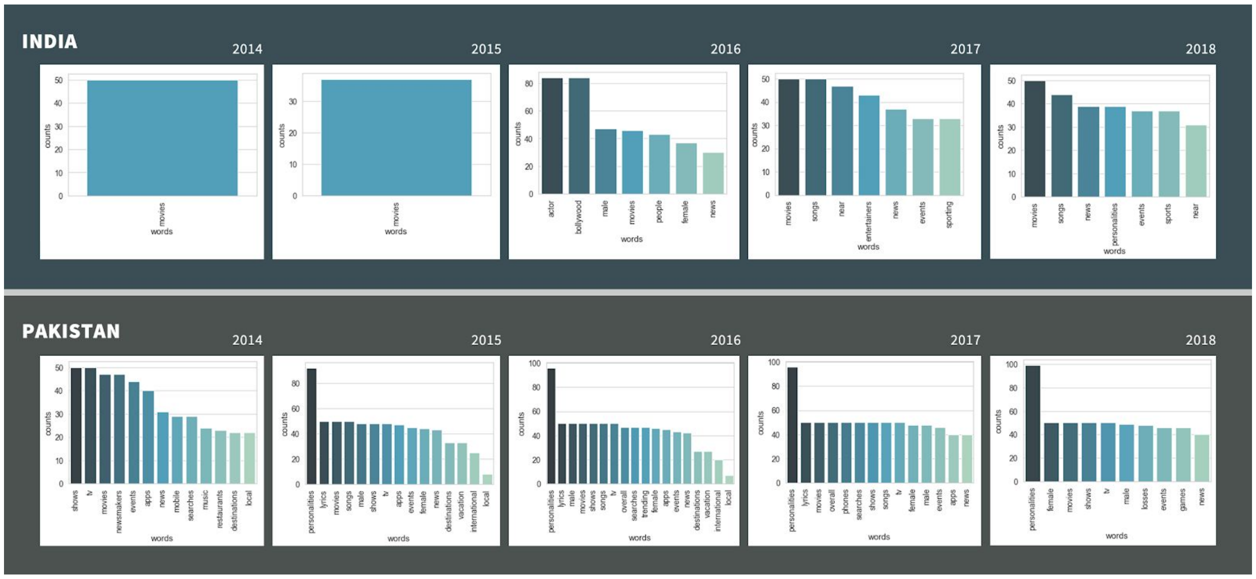


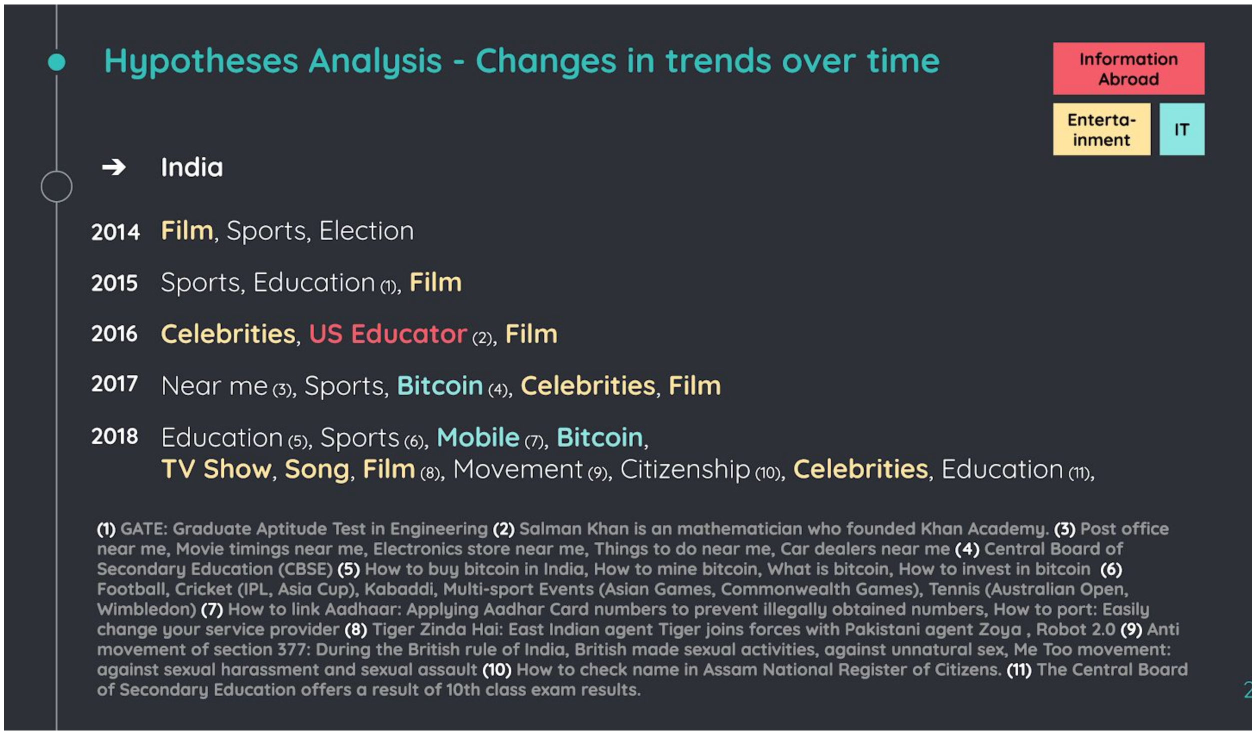
Figure. 8: Most common words from each category label over time

LDA model training and results - LatentDirichletAllocation[8]

To cluster into N number of topics on overall category labels, I needed to employ training LDA model (Appendix 3) Using "Latent Dirichlet Allocation with online variational Bayes algorithm" to create a training model for keywords by topics. I started by converting the

[8] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html

documents into a simple vector representation, and shifted a list of titles into lists of vectors, all with length equal to the vocabulary. Lastly, I classified these most common words per each country and year in manually to help with a better understanding and interpreting individual topics, and the relationships between the topics (Figure. 10).



Hypotheses Analysis - Changes in trends over time

Information Abroad

Enterta-inment

IT

➔ India

2014 Film, Sports, Election

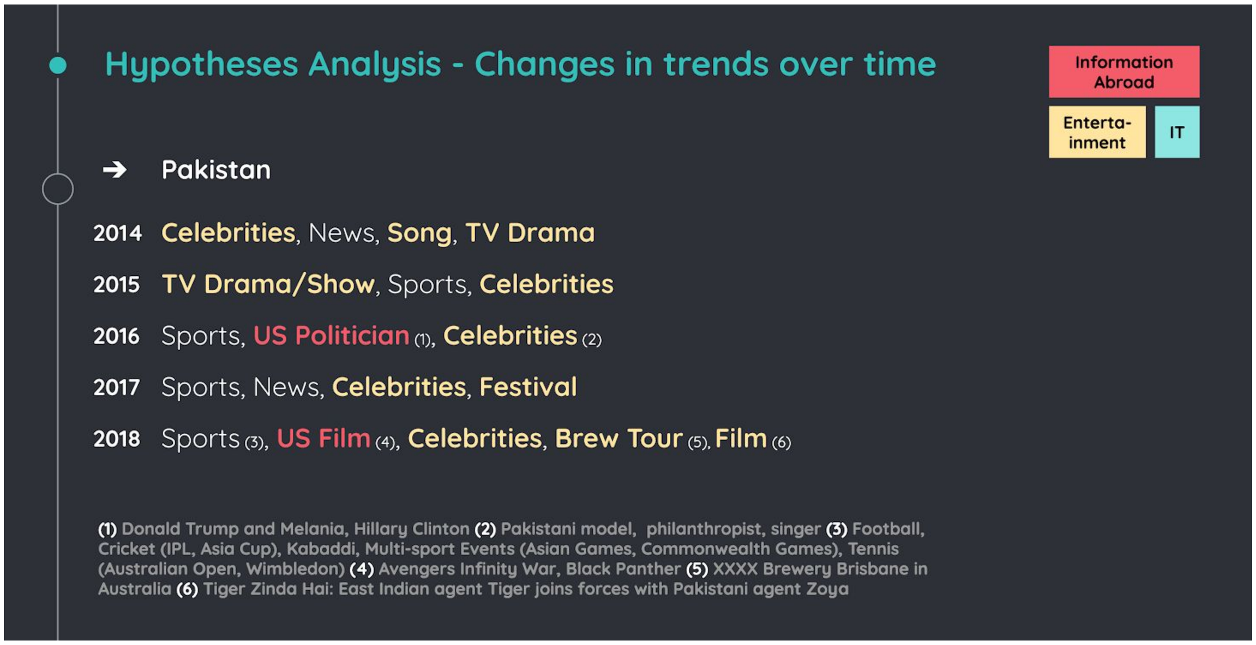
2015 Sports, Education (1), Film

2016 Celebrities, US Educator (2), Film

2017 Near me (3), Sports, Bitcoin (4), Celebrities, Film

2018 Education (5), Sports (6), Mobile (7), Bitcoin, TV Show, Song, Film (8), Movement (9), Citizenship (10), Celebrities, Education (11),

(1) GATE: Graduate Aptitude Test in Engineering **(2)** Salman Khan is an mathematician who founded Khan Academy. **(3)** Post office near me, Movie timings near me, Electronics store near me, Things to do near me, Car dealers near me **(4)** Central Board of Secondary Education (CBSE) **(5)** How to buy bitcoin in India, How to mine bitcoin, What is bitcoin, How to invest in bitcoin **(6)** Football, Cricket (IPL, Asia Cup), Kabaddi, Multi-sport Events (Asian Games, Commonwealth Games), Tennis (Australian Open, Wimbledon) **(7)** How to link Aadhaar: Applying Aadhar Card numbers to prevent illegally obtained numbers, How to port: Easily change your service provider **(8)** Tiger Zinda Hai: East Indian agent Tiger joins forces with Pakistani agent Zoya , Robot 2.0 **(9)** Anti movement of section 377: During the British rule of India, British made sexual activities, against unnatural sex, Me Too movement: against sexual harassment and sexual assault **(10)** How to check name in Assam National Register of Citizens. **(11)** The Central Board of Secondary Education offers a result of 10th class exam results.



Hypotheses Analysis - Changes in trends over time

Information Abroad

Enterta-inment

IT

➔ Pakistan

2014 Celebrities, News, Song, TV Drama

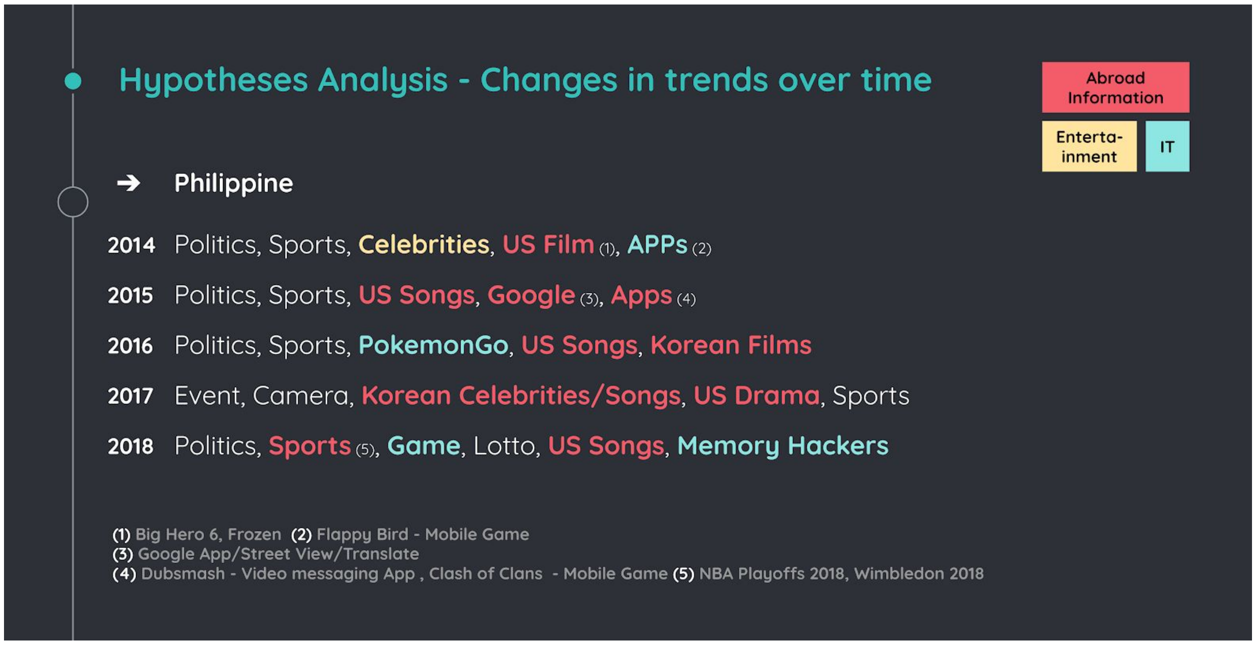
2015 TV Drama/Show, Sports, Celebrities

2016 Sports, US Politician (1), Celebrities (2)

2017 Sports, News, Celebrities, Festival

2018 Sports (3), US Film (4), Celebrities, Brew Tour (5), Film (6)

(1) Donald Trump and Melania, Hillary Clinton **(2)** Pakistani model, philanthropist, singer **(3)** Football, Cricket (IPL, Asia Cup), Kabaddi, Multi-sport Events (Asian Games, Commonwealth Games), Tennis (Australian Open, Wimbledon) **(4)** Avengers Infinity War, Black Panther **(5)** XXXX Brewery Brisbane in Australia **(6)** Tiger Zinda Hai: East Indian agent Tiger joins forces with Pakistani agent Zoya



Hypotheses Analysis - Changes in trends over time

Abroad Information

Enterta-inment

IT

➔ Philippine

2014 Politics, Sports, Celebrities, US Film (1), APPs (2)

2015 Politics, Sports, US Songs, Google (3), Apps (4)

2016 Politics, Sports, PokemonGo, US Songs, Korean Films

2017 Event, Camera, Korean Celebrities/Songs, US Drama, Sports

2018 Politics, Sports (5), Game, Lotto, US Songs, Memory Hackers

(1) Big Hero 6, Frozen **(2)** Flappy Bird - Mobile Game
(3) Google App/Street View/Translate
(4) Dubsmash - Video messaging App , Clash of Clans - Mobile Game **(5)** NBA Playoffs 2018, Wimbledon 2018

Figure. 10: Create each category over time and the classifications with color codings.

Findings

In overall, the keywords that go through India was Film, Pakistan got celebrities, and Pilippine had US culture, all of them can be classified by entertainment.
In the early stage of searching in Google, India got the film and sports on their most of keywords, Pakistan had the celebrities the most, and Philippine got US films; songs; and dramas. These keywords are quite similar topics that might be under the entertainment classification. Since India and Pakistan both have the same film in 2018 (Tiger Zinda Hai: East Indian agent Tiger joins forces with Pakistani agent Zoya), it also can tell that they were gotten to each other with the common background and interest. Interestingly, Philippine has been favored in lots of American culture but not two other countries.

Sports keyword is a key to describe these three countries' hobbies and interests that have been played the same major sport event for a long time.
Spatially, India and Pakistan have a lot of similarities in enjoying sports and holding sports events annually which are Football(FIFA World Cup), Cricket (IPL, Asia Cup), Kabaddi (Pro Kabaddi League) and Multi-sport Events (Asian Games, Commonwealth Games). These might cause of sharing the same culture in the past for a long time. On the other hand, Philippine have been only held in Football events (FIFA World Cup, Asian Games), but three countries all interested in watching the tennis games abroad such as Australian Open, Wimbledon.

Pilippine had already been interested in lots of different types of cultures, India has been gradually increased with a variety of topics in their interests, however, Pakistan still stick with their existing topics.
India became to pay closer attention to education, politics and News. Pakistan, however, kept focusing on entertainment topics. Philippine is the most various types of interests but mostly focusing on the entertainment topics.

Conclusion

The hypothesis of the report was finding any changes in trends in each region over time and capturing similarities or differences between three countries over time. Yes, India and Pakistan have a common understanding and culturally bound - both love playing or watching sports games, and like various movies, actors, and actresses in each other’s film. The only difference between these two countries were India has more diversity interests with education or news, Pakistan kept their interests in sports and celebrities. So if anyone interests to promote their services on three countries, India and Pakistan would be an almost the same target audience with these their interests. Philippine, however, has been already got interests abroad since 2014, It might have multiple sources to track the international trends so be aware of their market industries first.

Appendix

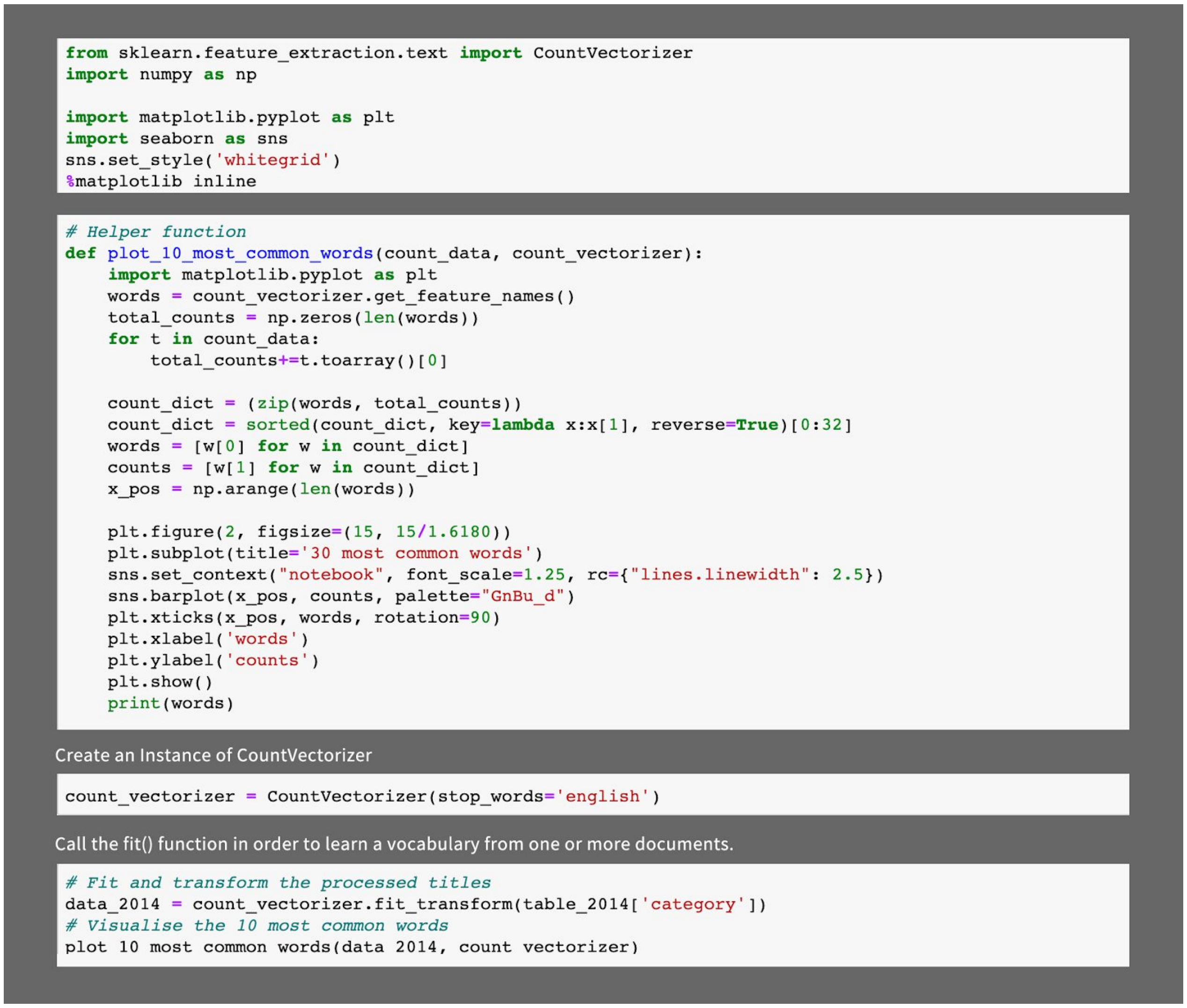
Appendix. 1 Scrape data from Google Trends

```
from selenium import webdriver
# 1st import: Allows you to launch/initiate a browser
from selenium.webdriver.common.by import By
# 2nd import: Allows you to search for things using specific parameters.
from selenium.webdriver.support.ui import WebDriverWait
# 3rd import: Allows you to wait for a page to load.
from selenium.webdriver.support import expected_conditions as EC
# 4th import: Specify what you are looking for on a specific page in order to
determine that the webpage has loaded.
from selenium.common.exceptions import TimeoutException
# 5th import: Handling a timeout situation
from selenium.common.exceptions import WebDriverException
# from selenium.webdriver.common.proxy import Proxy, ProxyType
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.proxy import Proxy, ProxyType

import re, pymongo, time, pdb, itertools, requests, json
from pymongo import MongoClient

def init_browser():
   # driver =
webdriver.Chrome('/Users/hh/Documents/ECT/rimhoho.github.io/chromedriver')
   prox = Proxy()
   prox.proxy_type = ProxyType.MANUAL
   prox.http_proxy = "http://localhost:8118"
   prox.ssl_proxy = "http://localhost:8118"

   capabilities = webdriver.DesiredCapabilities.CHROME
   prox.add_to_capabilities(capabilities)

   options = webdriver.ChromeOptions()
   options.binary_location = '/Applications/Google Chrome.app/Contents/MacOS/Google
Chrome'
   # options.add_argument('headless')
   # set the window size
   options.add_argument('window-size=1881x1280')

   # initialize the driver
   driver = webdriver.Chrome(options=options, desired_capabilities=capabilities)
   return driver

def searches_lists(driver, collections):
   base_url = 'https://trends.google.com/trends/yis/'
   years = ['2014', '2015', '2016', '2017', '2018']
   search_trend_urls = [base_url + year + '/' + country + '/' for country in
collections for year in years]
   # search_trend_urls = ['https://trends.google.com/trends/yis/2018/PK/']

   for each_country in search_trend_urls:
       time.sleep(2)
       driver.get(each_country)
```

```
        WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH,
'//*[@id="anchorName"]/div/div')))
        buttons =
driver.find_elements_by_xpath('//*[@id="anchorName"]/div/div/div[2]/div[1]')

        for b in buttons:
            driver.execute_script("arguments[0].click();", b)

        collection = each_country[-3:-1]
        year = re.sub('(https:\/\/trends.google.com\/trends\/yis\/)', '',
each_country[:-4])
        each_lists = driver.find_elements_by_class_name('grid-cell')
        container = []

        for each in each_lists:
            each_category = {}
            each_list = []
            ranking =
each.find_elements_by_class_name('fe-expandable-list-question-index')
            keyword = each.find_elements_by_class_name('fe-expandable-item-text')

            each_category['country'] = collection
            each_category['category'] = each.find_element_by_tag_name('span').text

            for i in range(len(ranking)):
                each_keyword = {}
                each_keyword['ranking'] = ranking[i].text
                each_keyword['keyword'] = keyword[i].text
                each_keyword['href'] = keyword[i].get_attribute('href')
                each_list.append(each_keyword)

            each_category['rank_keyword'] = each_list
            container.append(each_category)

        result = get_higher_search_by_region(driver, container)

        collections[collection].insert_many([
            { 'country' : each_key['country'],
              'year' : year,
              'category' : each_key['category'],
              'ranking' : each_infos['ranking'],
              'keyword' : each_infos['keyword'],
              'href' : each_infos['href'],
              'region' : each_searches['region'],
              'search_volume' : each_searches['search_volume']
        } for each_key in result for each_infos in each_key['rank_keyword'] for
each_searches in each_infos['higher_search_volumes']])
        print('result: ', result)
    return

def get_higher_search_by_region(driver, container):
    for each_category in container:
            for each_keyword in each_category['rank_keyword']:
                driver.get(each_keyword['href'])
                each_list = []
                time.sleep(3)
                try:
                    interest_lists = WebDriverWait(driver,
10).until(EC.presence_of_element_located((By.XPATH,
'/html/body/div[2]/div[2]/div/md-content/div/div/div[2]/trends-widget/ng-include/widge
t/div/div/ng-include/div/div/div[2]/widget/div')))
```

```
                   for item in
interest_lists.find_elements_by_class_name('progress-label'):
                       search_volume =
item.find_element_by_class_name('progress-value').text
                       if int(search_volume) > 49:
                           each_dict = {}
                           region =
item.find_element_by_class_name('label-text').find_element_by_tag_name('span').text
                           each_dict['region'] = region
                           each_dict['search_volume'] = search_volume
                           each_list.append(each_dict)
                   each_keyword['higher_search_volumes'] = each_list
               except Exception as e:
                   print('error', each_keyword['keyword'])
                   each_keyword['higher_search_volumes'] = ''
                   pass
   return container

def access_db(dbname, collectionnames):
   cluster =
MongoClient('mongodb+srv://rimho:0000@cluster0-yehww.mongodb.net/test?retryWrites=true
&w=majority')
   db = cluster[dbname]
   collections = {c:db[c] for c in collectionnames}
   return db, collections

def main():
   db, collections = access_db('3_Google_search_trends_db', ['IN'])
   driver = init_browser()
   time.sleep(3)
   searches_lists(driver, collections)
   print('Go To MongoDB')
   return

main()
```

Appendix. 2 Making a function to build vocabulary collection



```
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline
```

```
# Helper function
def plot_10_most_common_words(count_data, count_vectorizer):
    import matplotlib.pyplot as plt
    words = count_vectorizer.get_feature_names()
    total_counts = np.zeros(len(words))
    for t in count_data:
        total_counts+=t.toarray()[0]

    count_dict = (zip(words, total_counts))
    count_dict = sorted(count_dict, key=lambda x:x[1], reverse=True)[0:32]
    words = [w[0] for w in count_dict]
    counts = [w[1] for w in count_dict]
    x_pos = np.arange(len(words))

    plt.figure(2, figsize=(15, 15/1.6180))
    plt.subplot(title='30 most common words')
    sns.set_context("notebook", font_scale=1.25, rc={"lines.linewidth": 2.5})
    sns.barplot(x_pos, counts, palette="GnBu_d")
    plt.xticks(x_pos, words, rotation=90)
    plt.xlabel('words')
    plt.ylabel('counts')
    plt.show()
    print(words)
```

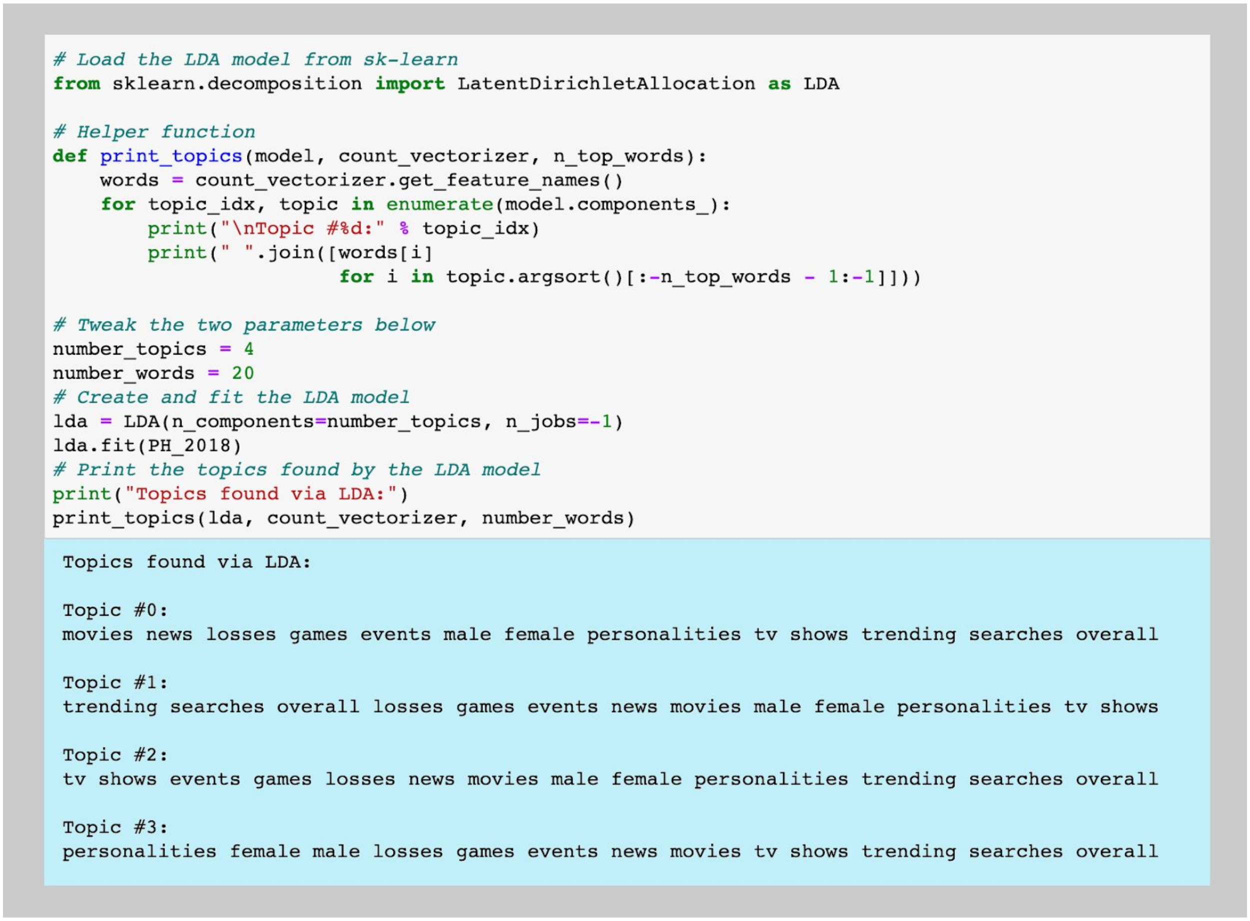
Create an Instance of CountVectorizer

```
count_vectorizer = CountVectorizer(stop_words='english')
```

Call the fit() function in order to learn a vocabulary from one or more documents.

```
# Fit and transform the processed titles
data_2014 = count_vectorizer.fit_transform(table_2014['category'])
# Visualise the 10 most common words
plot_10_most_common_words(data_2014, count_vectorizer)
```

Appendix. 3 Final topics for clustering category labels



```
# Load the LDA model from sk-learn
from sklearn.decomposition import LatentDirichletAllocation as LDA

# Helper function
def print_topics(model, count_vectorizer, n_top_words):
    words = count_vectorizer.get_feature_names()
    for topic_idx, topic in enumerate(model.components_):
        print("\nTopic #%d:" % topic_idx)
        print(" ".join([words[i]
                        for i in topic.argsort()[:-n_top_words - 1:-1]]))

# Tweak the two parameters below
number_topics = 4
number_words = 20
# Create and fit the LDA model
lda = LDA(n_components=number_topics, n_jobs=-1)
lda.fit(PH_2018)
# Print the topics found by the LDA model
print("Topics found via LDA:")
print_topics(lda, count_vectorizer, number_words)
```

```
Topics found via LDA:

Topic #0:
movies news losses games events male female personalities tv shows trending searches overall

Topic #1:
trending searches overall losses games events news movies male female personalities tv shows

Topic #2:
tv shows events games losses news movies male female personalities trending searches overall

Topic #3:
personalities female male losses games events news movies tv shows trending searches overall
```

References

Google searches in 1 second, Retrieved from:
https://www.internetlivestats.com/google-search-statistics/#trend

Internet Usage Statistics, Retrieved from:
https://www.internetworldstats.com/stats.htm

HostingFacts Team. Dec 17, 2018, Internet Stats & Facts for 2019. Retrieved from:
https://hostingfacts.com/internet-facts-stats/

wordcloud.ImageColorGenerator. Retrieved from:
https://amueller.github.io/word_cloud/generated/wordcloud.ImageColorGenerator.html

sklearn.feature_extraction.text.CountVectorizer. Retrieved from:
https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

How to Prepare Text Data. Retrieved from:
https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

Topic Modeling in Python. Retrieved from:
https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0

India–Pakistan relations. wikipedia. Retrieved from:
https://en.wikipedia.org/wiki/India%E2%80%93Pakistan_relations