Worksheet_4a

PES University, Bangalore¶

Established under Karnataka Act No. 16 of 2013

UE20CS312 - Data Analytics - Worksheet 4a¶

Course instructor: Gowri Srinivasa, Professor Dept. of CSE, PES University

Designed by Harshith Mohan Kumar, Dept. of CSE - harshithmohankumar@pesu.pes.edu

Collaborative & Content based filtering¶

The **Collaborative filtering method** for recommender systems is a method that is solely based on the past interactions that have been recorded between users and items, in order to produce new recommendations.

The **Content-based** approach uses additional information about users and/or items. The Content-based approach requires a good amount of information about items' features, rather than using the user's interactions and feedback.

Prerequisites¶

- Revise the following concepts
  - TF-IDF
  - Content-based filtering
  - Cosine Similarity
- Install the following software
  - pandas
  - numpy
  - sklearn

Task¶

After the disastrous pitfall of Game of Thrones season 8), George R. R. Martin set out to fix mindless mistakes caused by the producers David and Daniel.

A few years down the line, we now are witnessing George R. R. Martin's latest work: House of the Dragon. This series is a story of the Targaryen civil war that took place about 200 years before events portrayed in Game of Thrones.

In this notebook you will be exploring and analying tweets related to The House of Dragon TV series. First we shall tokenize the textual data using TF-IDF. Then we will proceed to find the tok-k most similar tweets using cosine similarity between the transformed vectors.

The dataset has been extracted using the Twitter API by utilizing a specific search query. The data has been extensively preprocessed and a small subset has been stored within the `twitter_HOTD_DA_WORKSHEET4A.csv`

Note: This notebook may contain spoilers to the show.

Data Dictionary¶

author_id: A unique identifier assigned to the twitter user.

tweet_id: A unique identifier assigned to the tweet.

text: The text associated with the tweet.

retweet_count: The number of retweets for this particular tweet.

reply_count: The number of replies for this particular tweet.

like_count: The number of likes for this particular tweet.

quote_count: The number of quotes for this particular tweet.

tokens: List of word tokens extracted from `text`.

hashtags: List of hashtags extracted from `text`.

Points¶

The problems in this worksheet are for a total of 10 points with each problem having a different weightage.

- Problem 1: 4 points
- Problem 2: 4 points
- Problem 3: 2 point

Loading the dataset¶

In [1]:

```
# Import pandas
import pandas as pd
# Use pandas read_csv function to load csv as DataFrame
df = pd.read_csv('./twitter_HOTD_DA_WORKSHEET4A.csv')
df
```

Out[1]:

	author_id	tweet_id	text
0	1576486223442583554	1577813753902555136	rt i would perform my duty if mother had only ...
1	732912167846973441	1577813747846254592	rt viserys look at me!! aemond #houseofthedragon
2	891426095928672257	1577813743794257920	rt house of the dragon is a show about a king ...
3	352012951	1577813724366249986	man just thinking of when viserys finally croa...
4	1090278774925611008	1577813708818059264	rt jajaja. #houseofthedragon
...	...	...	...
8056	2916627870	1570663468055040000	rt may i go on record that if matt smith looke...
8057	2916627870	1570663455606312961	rt good morning to matt smith who is now respo...
8058	2916627870	1570663389684453378	rt the entire world paused during this #houseo...
8059	212524158	1570663558563921922	days left #hotd
8060	1497796183279996929	1570663337658298371	the #houseofthedragon cast is too small. due t...

8061 rows × 9 columns

Problem 1 (4 points)¶

Tokenize the string representations provided in the **tokens** column of the DataFrame using TF-IDF from sklearn. Then print out the TF-IDF of the first row of the DataFrame.

Solution Steps:

1. Initialize the `TfidfVectorizer()`
2. Use the `.fit_transform()` method on the entire text
3. `.transform()` the Text
4. Print number of samples and features using `.shape`
5. Print the TF-IDF of the first row

For futher reference: https://www.analyticsvidhya.com/blog/2021/09/creating-a-movie-reviews-classifier-using-tf-idf-in-python/

In [2]:

```
# Imports
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [3]:

```
# Convert string representation of a list into a list of strings
import ast
text = []
for r in df['tokens']:
    res = ast.literal_eval(r)
    if(' '.join(res).lower() == ''):
```

```
        print(r)
    text.append(' '.join(res).lower())
# Print the end result
text
```

Out[3]:

```
['perform duty mother betrothed',
 'viserys look aemond',
 'house dragon king bunch questionable decisions based idea dead',
 'man thinking viserys finally croaks fucking races gon na fucking hate damn cliffhanger mak
 'jajaja',
 'funny   ',
 'way lucerys threw blade sending  ',
 'house dragon alright kinda dog',
 'showmax watch house dragon nigeria',
 'poor guy   ',
 'trying catchup house dragon rhaenyra better written galadriel rop.',
 'daemon targaryen criston cole owner  ',
 'daemyra',
 'love scene fucking',
 'house dragon bollywood',
 'opening targaryen family tree bloodline represented king viserys valyria model diorama ama
 'baela rhaena  younger older',
 'green like alicent',
 'watching ep',
 'scenes photos episode',
 'viserys raven reaches king landing informs rhaenyra daemon got married',
 'year ago today got official teaser',
 'aemon holding life    happy biggest dragon',
 'imagine scene song \x01f90c  ',
 'living sons remained strength consolation queen called',
 'boy right',
 'spot difference  ',
 'viserys reading westeros daily paper finding rhaenyra daemon got married',
 'started going',
 'arty froushan john macmillan scenes episode',
 'fire blood',
 'dragons riders far',
 'echo de menos este principe canalla',
 'moment knew coming',
 'leo ashton ty tennant scenes episode   ',
 'spot difference  ',
 'fans defended hbo max decision writing   full-moon scenes house',
 'dragon matters sudden benchmark closest comp success',
 'maesters episode',
 'parallels viserys targaryen children',
```

Problem 2 (4 points)¶

Find the top-5 most similar tweets to the tweet with index `7558` using cosine similarity between the TF-IDF vectors.

Solution Steps:

1. Import `cosine_similarity` from sklearn.metrics.pairwise
2. Compute `cosine_similarity` using text_tf with index `7558` and all other rows
3. Use `argsort` to sort the cosine_similarity results
4. Print indices of top-5 most similar results from sorted array (hint: argsort sorts in ascending order)
5. Display text of top-5 most similar results using `df.iloc[index]`

In [4]:

```
# Print out the tokens from index `7654`
print(text[7558])
# Print out the text from index `7654`
print(df.iloc[7558][2])
```

```
viserys wanna build lego set mind business let man live peace
rt viserys just wanna build his lego set and mind his business . let that man live in peace
```

Problem 3 (2 point)¶

A great disadvantage in using TF-IDF is that it can not capture semantics. If you had classify tweets into positive/negative, what technique would you use to map words to vectors? In short words, provide the sequence of solution steps to solve this task. Note: Assume sentiment labels have been provided.

(Hint: take a look at how I've provided solution steps in previous problems)