DATA ANALYTICS- WORKSHEET 4A

GAURAV MAHAJAN

PES1UG20CS150

Collaborative & Content based filtering

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

	author_id	tweet_id	text	retweet_count	rep
0	1576486223442583554	1577813753902555136	rt i would perform my duty if mother had only ...	327	
1	732912167846973441	1577813747846254592	rt viserys look at me!! aemond #houseofthedragon	2164	
2	891426095928672257	1577813743794257920	rt house of the dragon is a show about a king ...	905	
3	352012951	1577813724366249986	man just thinking of when viserys finally croa...	0	
4	1090278774925611008	1577813708818059264	rt jajaja. #houseofthedragon	29	
...	...	...	...	...	
8056	2916627870	1570663468055040000	rt may i go on record that if matt smith looke...	166	

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

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

```
# 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
```

```
['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 making wait year war start lol',
 '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',
 'hate ser criston anybody want stab repeatedly screen hotness ai anymore',
 'despite morally grey villain possibly main antagonist series fans believe
daemon targaryen',
 'throwback teaser got crazy end season https t',
 'pure gold \x01f979 ♥  😂',
 'continues disappoint week week love character progression costumes story pacing
way book written important moments history bog',
 'succession house dragon',
 'house dragon',
 'daemon volvio su hogar',
 'mi semanal matalos otto matalos todos adhiero un matalos aemond mata esos
bastardos granujas',
 'going live today stream actually help support community signing w',
 'meant burn draw scene 🖤',
 'aemond targaryen bastard jace',
 '😂 😂 😂',
```

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(text)
X_new = vectorizer.transform(text)
```

```
X_new.shape
```

```
(8061, 10950)
```

```
print(X_new[0])
```

```
  (0, 7080)     0.5632689245894086
  (0, 6277)     0.3774828944719565
  (0, 3021)     0.47218947633853087
  (0, 1054)     0.5632689245894086
```

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]`

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
# 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 ir
```

```
a=cosine_similarity(X_new[7558] , X_new)
```

```
b=a[0].argsort()
```

```
c=b[::-1]
```

```
print(c[0:5])
```

```
[7595 7558 3656 6548 7705]
```

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)

Technique used is word2vec

-> Word embeddings is a technique where individual words are transformed into a numerical representation of the word (a vector). Where each word is mapped to one vector, this vector is then learned in a way which resembles a neural network. -> The vectors try to capture various

characteristics of that word with regard to the overall text. -> These characteristics can include the semantic relationship of the word, definitions, context, etc.

Solution:

from gensim.models import Word2Vec as w2v

remove stopwords

w = w2v(filtered_lines,min_count=3, sg = 1,window=7)

emb_df = (pd.DataFrame([w.wv.get_vector(str(n)) for n in w.wv.key_to_index], index = w.wv.key_to_index))