CSE4077-Recommender Systems

J Component – Project Report

Article Recommender System

By

19MIA1001	Roshan Srinivaas
19MIA1003	Niranjan J
19MIA1093	Vignesh.N

M.Tech CSE with Specialization in Business Analytics

Submitted to

Dr. A.Bhuvaneshwari,
Assistant Professor Senior,
SCOPE, VIT, Chennai

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

November 2022

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

FALL SEM 22-23

Worklet details

Programme	M.Tech Integrated Business Analytics	
Course Name / Code	Recommender Systems / CSE4077	
Slot	E1 + TE1	
Faculty Name	Dr.A.Bhuvaneshwari	
Component	J – Component	
J Component Title	Article Recommender System	
Team Members Name | Reg. No	Roshan Srinivaas	19MIA1001
	Niranjan J	19MIA1003
	Vignesh. N	19MIA1093

Team Members(s) Contributions – Tentatively planned for implementation:

Worklet Tasks	*Contributor's Names*
Preprocessing	Niranjan, Vignesh, Roshan
Model building	Niranjan, Vignesh, Roshan
Visualization	Niranjan, Vignesh, Roshan
Technical Report writing	Niranjan, Vignesh, Roshan
Presentation preparation	Niranjan, Vignesh, Roshan

Acknowledgement

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr.Bhuvaneshwari** mam Assistant Professor, School of Computer Science Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We also take this opportunity to thank all the faculty of the school for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Roshan Srinivaas (19MIA1001)
Niranjan J (19MIA1003)
Vignesh.N (19MIA1093)

Abstract

In the last decade we have observed a mass increase of information, in particular information that is shared through smartphones. Consequently, the amount of information that is available does not allow the average user to be aware of all his options. In this context, recommender systems use a number of techniques to help a user find the desired product. Hence, nowadays recommender systems play an important role. Recommender Systems aim to identify products that best fits user preferences. These techniques are advantageous to both users and vendors, as it enables the user to rapidly find what he/she needs and the vendors to promote their products and sales. As the industry became aware of the gains that could be accomplished by using these algorithms, also a very interesting problem for many researchers, recommender systems became a very active area. Having in mind that this is an ongoing problem the present project intends to observe the value of using a recommender algorithm to find users likes by observing his/her domain preferences in news article. In this project we will show how news topics can be used to recommend news articles. In this project, we use content-based filtering technique to determine the user ratings for an article.

Introduction

It is highly difficult and time-consuming to find the desired item of choice fast due to the easy availability of enormous things (services) on our favorite online platforms like e-commerce, e-articles, e-newspapers such as ‘Amazon Audible’. Content based recommendation using **News category** dataset, the goal is to recommend news articles which are similar to the already read article by using attributes like article headline, category, author and publishing date. The number of news releases has grown rapidly and for one individual, it is cumbersome to browse through all online news resources for relevant news articles. Search engines help users up-to some extent in searching through the vast information collections available online and the recommendation systems have emerged to address different challenges and provide users with the information which matches their needs either by their preferences or by content similarity. Each online news publisher tries to handle its news and use some mechanisms to recommend similar news to their readers. The recommendation of news articles is a hard task because of a highly dynamic environment, which leads several challenges, e.g. frequent changes in the set of news articles, set of users, rapid changes in user’s preferences, etc. Therefore, recommendation algorithms must be able to process continuous incoming news streams in real-time. The complex requirements of news recommendations based on some relevancy among news articles that best fullfill the user’s requirements lead to many rich research scenarios and make it more interesting.

Literature Survey

Sl No	Title	Author / Journal name / Year	Technique	Result
1	News Recommendation with Recurrent Neural Networks	Gabriel de souza p. moreira, Dietmar jannach, adilson marques da cunha,2019	RNN, LSTM	Proposed hybrid approach leads to higher prediction accuracy.
2	News Recommendation Systems	Chong feng, Muzammil Khan, Journal of recommendation system,2020	Collaborative Filtering Approach	News recommender systems (NRS) are developed to relieve the information overload problem and suggest news items that might be of interest for the news readers.
3	News Recommendation system	Jing Qin Journal of recommendation model, 2020	CF method, userbased collaborative filtering (UserCF) method, ItemCF, CNN	CF methods are used to calculate the similarity of users or news or MFbased modelling CF method have better accuracy than CB
4	Recommender system for news articles using supervised learning	Akshay kumar chaturvedi	Linear regression, Naïve bayes classifier, Logistic regression	Linear regression turned out to be the best one. Number of classes and performance of model are inversely proportional to each other.

5	Article	Nayana k	Content based	Collaborative
	recommendation using python		and collaborative based filtering	filtering automatically builds a recommendation system that evolves and decides which feature needs to be used. Content based filtering at the start will help avoid problems like cold start where users have no history.
6	News recommender system : a review of recent progress, challenges and opportunities	Shaina raza and chen ding	Deep neural networks	The general algorithms are insufficient to provide news recommendations since they need to be modified, varied and extended to large extent. Deep learning-based algorithms have addressed those limitations. Beyond accuracy other aspects like diversity, coverage, novelty, serendipity is important to provide better user experience in news recommender system.

Datasets And Tools Used

The dataset contains around 200k news headlines from the year 2012 to 2018. The model trained on this dataset could be used to identify tags for untracked news articles or to identify the type of language used in different news articles.

Various categories in our dataset include :

Politics, Entertainment, Wellness, Travel, Style and beauty, Parenting, Healthy living and Queer voices, Food and drinks, Business, Comedy, Sports, Crime, Media, Science, Education etc. Within each category our dataset contains the Headlines of the news and the authors of that particular news. The link of that specified news is also provided with respect to that particular author.

There also includes a column for short_description of each news and the particular date it was published. These specified categories and columns in our dataset help in segregating each news and they further enhance the results of our dataset. Our dataset is obtained from Huffpost.

The tool we are going to work on this project is ‘Python’, where use use libraries such as numpy, pandas, matplotlib, sklearn and nltk.

Algorithms / Techniques description

Collaborative filtering tackles the similarities between the users and items to perform recommendations. Meaning that the algorithm constantly finds the relationships between the users and in-turns does the recommendations. The algorithm learns the embeddings between the users without having to tune the features. The most common technique is by performing Matrix Factorization to find the embeddings or features that make up the interest of a particular user.

Algorithm used here is Content based filtering. The aim of this content-based recommender system is to estimate the utility of a set of objects belonging to a given domain, starting from the information available about users and objects. The basis for content-based recommendations is the similarity between users/items determined by their qualities. It makes use of extra data (meta data) about persons or objects, i.e., it depends on the kind of content that is already available. This meta data may include demographic information about the user, such as age, gender, occupation, location, and skill sets. Similar information may be found for products, including item name, details, category, registration date, etc. Therefore, the fundamental notion is to suggest items by identifying persons or items that are similar to the subject user or item based on those individual's traits.

Techniques description: (following are the procedures for this technique)

1) Importing necessary libraries

2) Loading Data

3) Data Preprocessing
- Fetching only the articles from 2018
- Removing all the short headline articles
- Checking and removing all the duplicates
- Checking for missing values

4) Basic Data Exploration
- Basic statistics - Number of articles, authors, categories
- Distribution of articles category-wise
- Number of articles per month
- PDF for length of headlines

5) Text Preprocessing
- Stopwords removal

- Lemmatization

6) Headline based similarity on new articles

- Using Bag of Words method
- Using TF-IDF method

- Using Word2Vec embedding

- Weighted similarity based on headline and categories

- Weighted similarity based on headline, category and author

Weighted similarity based on headline, category, author and publishing day

Implementation

Importing necessary libraries

```
[ ] import nltk
    #nltk.download('stopwords')
    #nltk.download('punkt')
    #nltk.download('wordnet')
    #nltk.download('omw-1.4')
```

```
import numpy as np
import pandas as pd

import os
import math
import time

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.figure_factory as ff
import plotly.graph_objects as go
import plotly.express as px

# Below libraries are for text processing using NLTK
from nltk.corpus import stopwords
from nltk.tokenize import punkt
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# Below libraries are for feature representation using sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

# Below libraries are for similarity matrices using sklearn
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import pairwise_distances
```

2. Loading Data

```
news_articles = pd.read_json("/content/Article Recommendation System.json", lines = True)
```

```
news_articles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10002 entries, 0 to 10001
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   category           10002 non-null  object
 1   headline           10002 non-null  object
 2   authors            10002 non-null  object
 3   link               10002 non-null  object
 4   short_description  10002 non-null  object
 5   date               10002 non-null  datetime64[ns]
dtypes: datetime64[ns](1), object(5)
memory usage: 469.0+ KB
```

The dataset contains about two million records of six different features.

```
[ ] news_articles.head()
```

	category	headline	authors	link	short_description	date
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...	2018-05-26
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...	Of course it has a song.	2018-05-26
2	ENTERTAINMENT	Hugh Grant Marries For The First Time At Age 57	Ron Dicker	https://www.huffingtonpost.com/entry/hugh-gran...	The actor and his longtime girlfriend Anna Ebe...	2018-05-26
3	ENTERTAINMENT	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	Ron Dicker	https://www.huffingtonpost.com/entry/jim-carre...	The actor gives Dems an ass-kicking for not fi...	2018-05-26
4	ENTERTAINMENT	Julianna Margulies Uses Donald Trump Poop Bags...	Ron Dicker	https://www.huffingtonpost.com/entry/julianna-...	The "Dietland" actress said using the bags is ...	2018-05-26

3. Data Preprocessing

3.a Fetching only the articles from 2018

```
[ ] news_articles = news_articles[news_articles['date'] >= pd.Timestamp(2018,1,1)]
```

```
[ ] news_articles.shape
```

```
(8583, 6)
```

Now, the number of news articles comes down to 8583.

3.b Removing all the short headline articles

```
[ ] news_articles = news_articles[news_articles['headline'].apply(lambda x: len(x.split())>5)]
    print("Total number of articles after removal of headlines with short title:", news_articles.shape[0])
```

```
Total number of articles after removal of headlines with short title: 8530
```

3.c Checking and removing all the duplicates

```
[ ] news_articles.sort_values('headline',inplace=True, ascending=False)
    duplicated_articles_series = news_articles.duplicated('headline', keep = False)
    news_articles = news_articles[~duplicated_articles_series]
    print("Total number of articles after removing duplicates:", news_articles.shape[0])
```

```
Total number of articles after removing duplicates: 8485
```

3.d Checking for missing values

```
[ ] news_articles.isna().sum()
```

```
category            0
headline            0
authors             0
link                0
short_description   0
date                0
dtype: int64
```

4. Basic Data Exploration

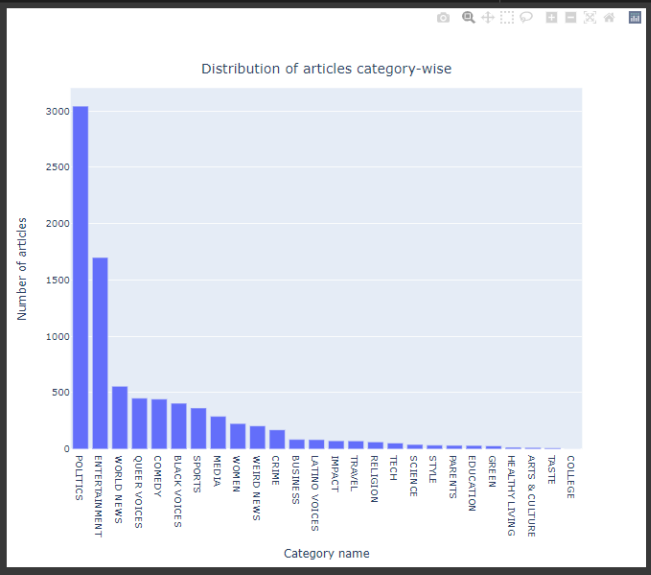
4.a Basic statistics - Number of articles,authors,categories

```
[ ] print("Total number of articles : ", news_articles.shape[0])
    print("Total number of authors : ", news_articles["authors"].nunique())
    print("Total number of unqiue categories : ", news_articles["category"].nunique())
```

```
Total number of articles :  8485
Total number of authors :  892
Total number of unqiue categories :  26
```

4.b Distribution of articles category-wise

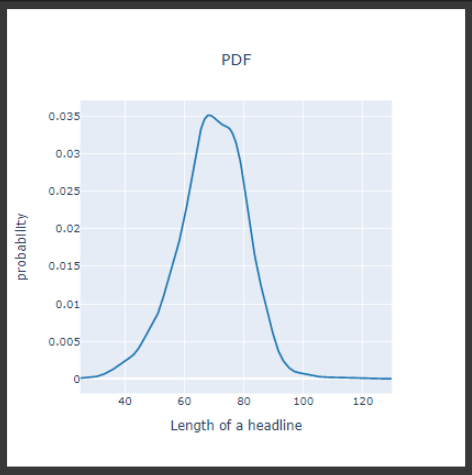
```
fig = go.Figure([go.Bar(x=news_articles["category"].value_counts().index, y=news_articles["category"].value_counts().values)])
fig['layout'].update(title={"text" : 'Distribution of articles category-wise','y':0.9,'x':0.5,'xanchor': 'center','yanchor': 'top'}, xaxis_title="Category name",yaxis_title="Number of articles")
fig.update_layout(width=800,height=700)
fig
```



From the bar chart, we can observe that **politics** category has **highest** number of articles then **entertainment** and so on.

4.d PDF for the length of headlines

```
fig = ff.create_distplot([news_articles['headline'].str.len()], ["ht"],show_hist=False,show_rug=False)
fig['layout'].update(title={'text':'PDF','y':0.9,'x':0.5,'xanchor': 'center','yanchor': 'top'}, xaxis_title="Length of a headline",yaxis_title="probability")
fig.update_layout(showlegend = False,width=500,height=500)
fig
```




```
[ ] news_articles.index = range(news_articles.shape[0])
```

```
[ ] # Adding a new column containing both day of the week and month, it will be required later while recommending based on day of the week and month
    news_articles["day and month"] = news_articles["date"].dt.strftime("%a") + "_" + news_articles["date"].dt.strftime("%b")
```

Since after text preprocessing the original headlines will be modified and it doesn't make sense to recommend articles by displaying modified headlines so let's copy the dataset into some other dataset and perform text preprocessing on the later.

5. Text Preprocessing

5.a Stopwords removal

```
stop_words = set(stopwords.words('english'))
```

```
[ ] for i in range(len(news_articles_temp["headline"])):
        string = ""
        for word in news_articles_temp["headline"][i].split():
            word = ("".join(e for e in word if e.isalnum()))
            word = word.lower()
            if not word in stop_words:
              string += word + " "
        if(i%1000==0):
          print(i)           # To track number of records processed
        news_articles_temp.at[i,"headline"] = string.strip()
```

```
0
1000
2000
3000
4000
5000
6000
7000
8000
```

5.b Lemmatization

```
[ ]  lemmatizer = WordNetLemmatizer()
```

```
for i in range(len(news_articles_temp["headline"])):
    string = ""
    for w in word_tokenize(news_articles_temp["headline"][i]):
        string += lemmatizer.lemmatize(w,pos = "v") + " "
    news_articles_temp.at[i, "headline"] = string.strip()
    if(i%1000==0):
        print(i)          # To track number of records processed
```

```
0
1000
2000
3000
4000
5000
6000
7000
8000
```

6. Headline based similarity on new articles

Generally, we assess **similarity** based on **distance**. If the **distance** is minimum then high **similarity** and if it is maximum then low **similarity**. To calculate the **distance**, we need to represent the headline as a **d-dimensional** vector. Then we can find out the **similarity** based on the **distance** between vectors.

There are multiple methods to represent a **text** as **d-dimensional** vector like **Bag of words**, **TF-IDF method**, **Word2Vec embedding** etc. Each method has its own advantages and disadvantages.

Let's see the feature representation of headline through all the methods one by one.

6.a Using Bag of Words method

A **Bag of Words(BoW)** method represents the occurence of words within a **document**. Here, each headline can be considered as a **document** and set of all headlines form a **corpus**.

Using **BoW** approach, each **document** is represented by a **d-dimensional** vector, where **d** is total number of **unique words** in the corpus. The set of such unique words forms the **Vocabulary**.

+ Code + Markdown

```
headline_vectorizer = CountVectorizer()
headline_features   = headline_vectorizer.fit_transform(news_articles_temp['headline'])
```

Python

```
headline_features.get_shape()
```

Python

```
(8485, 11122)
```

The output **BoW matrix**(headline_features) is a sparse matrix.

```
pd.set_option('display.max_colwidth', -1)  # To display a very long headline completely
```

Python

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:

Passing a negative integer is deprecated in version 1.0 and will not be supported in future version. Instead, use None to not limit the column width.
```

```
def bag_of_words_based_model(row_index, num_similar_items):
    couple_dist = pairwise_distances(headline_features,headline_features[row_index])
    indices = np.argsort(couple_dist.ravel())[0:num_similar_items]
    df = pd.DataFrame({'publish_date': news_articles['date'][indices].values,
                       'headline':news_articles['headline'][indices].values,
                       'Euclidean similarity with the queried article': couple_dist[indices].ravel()})
    print("="*30,"Queried article details","="*30)
    print('headline : ',news_articles['headline'][indices[0]])
    print("\n","="*25,"Recommended articles : ","="*23)
    #return df.iloc[1:,1]
    return df.iloc[1:,]

bag_of_words_based_model(133, 11) # Change the row index for any other queried article
```

Python

```
============================== Queried article details ==============================
headline :  Woman Fired After Flipping Off Trump's Motorcade Sues Former Employer

 ========================= Recommended articles :  =========================
```

6.b Using TF-IDF method

TF-IDF method is a weighted measure which gives more importance to less frequent words in a corpus. It assigns a weight to each term(word) in a document based on **Term frequency(TF)** and **inverse document frequency(IDF)**.

TF(i,j) = (# times word i appears in document j) / (# words in document j)

IDF(i,D) = log_e(#documents in the corpus D) / (#documents containing word i)

weight(i,j) = **TF(i,j)** x **IDF(i,D)**

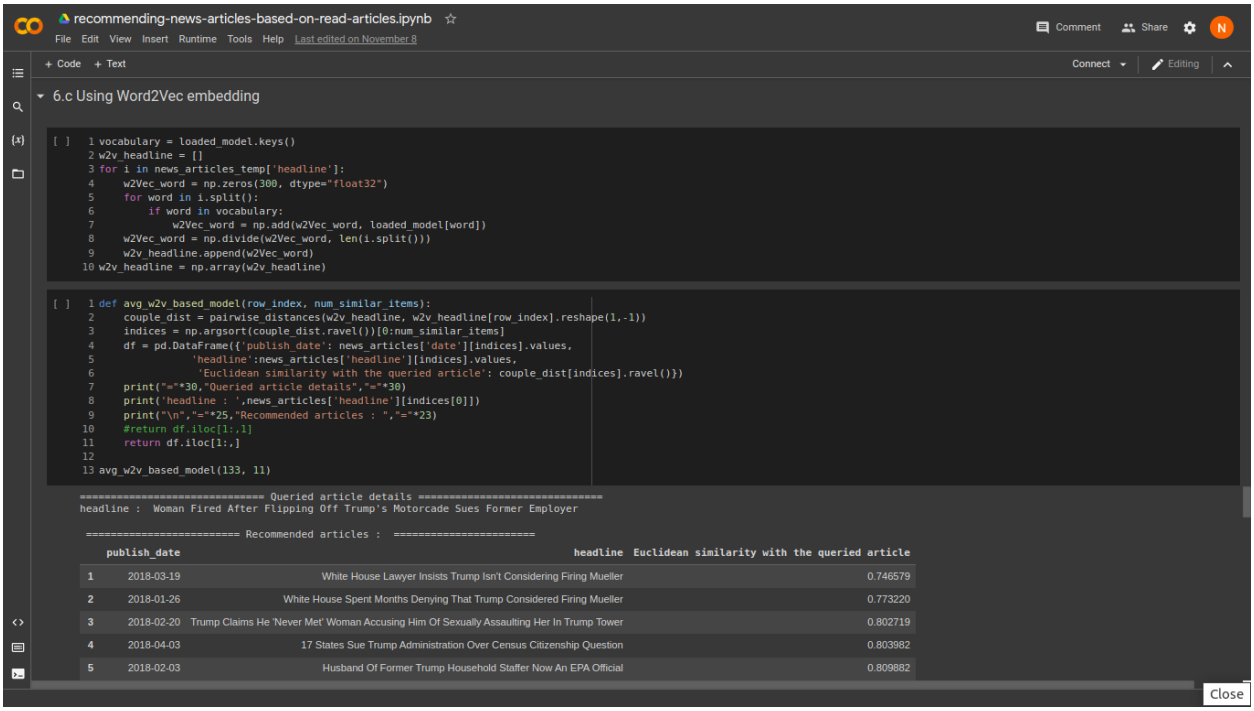
So if a word occurs more number of times in a document but less number of times in all other documents then its **TF-IDF** value will be high.

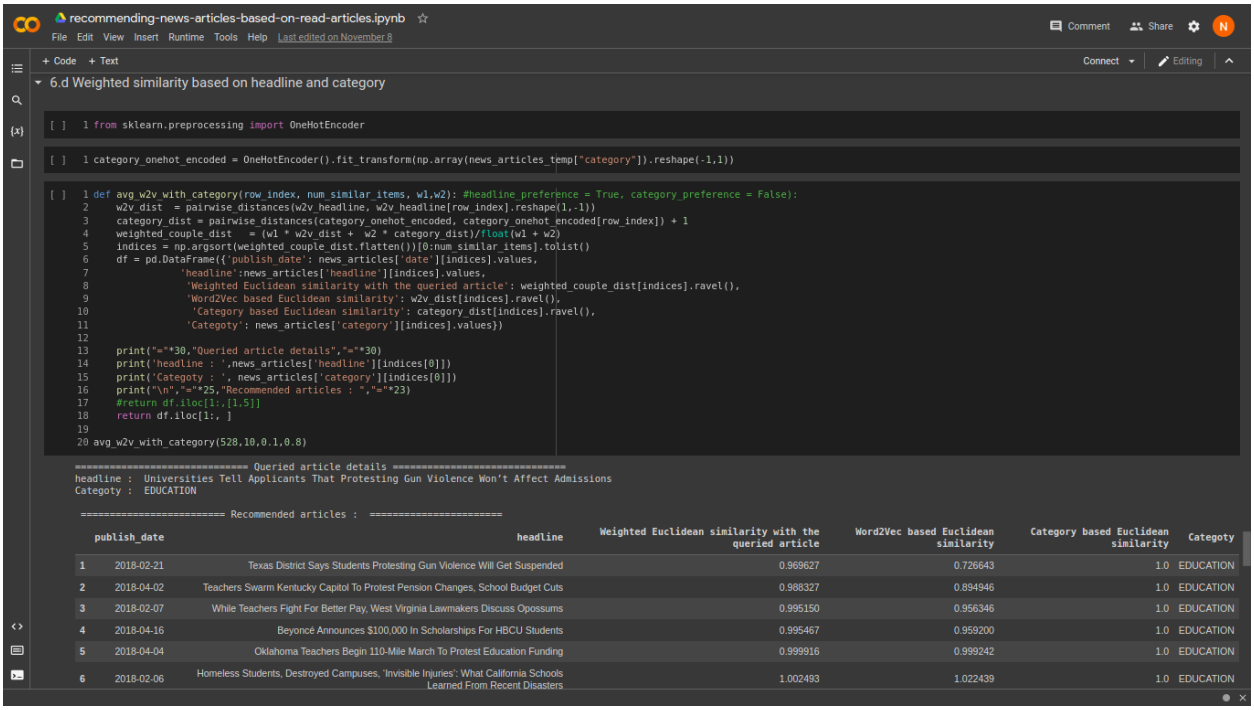
```
tfidf_headline_vectorizer = TfidfVectorizer(min_df = 0)
tfidf_headline_features = tfidf_headline_vectorizer.fit_transform(news_articles_temp['headline'])
```

Python

```
def tfidf_based_model(row_index, num_similar_items):
    couple_dist = pairwise_distances(tfidf_headline_features,tfidf_headline_features[row_index])
    indices = np.argsort(couple_dist.ravel())[0:num_similar_items]
    df = pd.DataFrame({'publish_date': news_articles['date'][indices].values,
                       'headline':news_articles['headline'][indices].values,
                       'Euclidean similarity with the queried article': couple_dist[indices].ravel()})
    print("="*30,"Queried article details","="*30)
    print('headline : ',news_articles['headline'][indices[0]])
    print("\n","="*25,"Recommended articles : ","="*23)

    #return df.iloc[1:,1]
    return df.iloc[1:,]
tfidf_based_model(133, 11)
```


recommending-news-articles-based-on-read-articles.ipynb
File Edit View Insert Runtime Tools Help Last edited on November 8
Comment Share
+ Code + Text
Connect Editing
6.c Using Word2Vec embedding
vocabulary = loaded_model.keys()
w2v_headline = []
for i in news_articles_temp['headline']:
w2Vec_word = np.zeros(300, dtype="float32")
for word in i.split():
if word in vocabulary:
w2Vec_word = np.add(w2Vec_word, loaded_model[word])
w2Vec_word = np.divide(w2Vec_word, len(i.split()))
w2v_headline.append(w2Vec_word)
w2v_headline = np.array(w2v_headline)
def avg_w2v_based_model(row_index, num_similar_items):
couple_dist = pairwise_distances(w2v_headline, w2v_headline[row_index].reshape(1,-1))
indices = np.argsort(couple_dist.ravel())[0:num_similar_items]
df = pd.DataFrame({'publish_date': news_articles['date'][indices].values,
'headline':news_articles['headline'][indices].values,
'Euclidean similarity with the queried article': couple_dist[indices].ravel()})
print("="*30,"Queried article details","="*30)
print('headline : ',news_articles['headline'][indices[0]])
print("\n","="*25,"Recommended articles : ","="*23)
#return df.iloc[1:,1]
return df.iloc[1:,]
avg_w2v_based_model(133, 11)
========== Queried article details ==========
headline : Woman Fired After Flipping Off Trump's Motorcade Sues Former Employer
========== Recommended articles : ==========
publish_date headline Euclidean similarity with the queried article
1 2018-03-19 White House Lawyer Insists Trump Isn't Considering Firing Mueller 0.746579
2 2018-01-26 White House Spent Months Denying That Trump Considered Firing Mueller 0.773220
3 2018-02-20 Trump Claims He 'Never Met' Woman Accusing Him Of Sexually Assaulting Her In Trump Tower 0.802719
4 2018-04-03 17 States Sue Trump Administration Over Census Citizenship Question 0.803982
5 2018-02-03 Husband Of Former Trump Household Staffer Now An EPA Official 0.809882
Close


recommending-news-articles-based-on-read-articles.ipynb
File Edit View Insert Runtime Tools Help Last edited on November 8
Comment Share
+ Code + Text
Connect Editing
6.d Weighted similarity based on headline and category
from sklearn.preprocessing import OneHotEncoder
category_onehot_encoded = OneHotEncoder().fit_transform(np.array(news_articles_temp["category"]).reshape(-1,1))
def avg_w2v_with_category(row_index, num_similar_items, w1,w2): #headline preference = True, category_preference = False):
w2v_dist = pairwise_distances(w2v_headline, w2v_headline[row_index].reshape(1,-1))
category_dist = pairwise_distances(category_onehot_encoded, category_onehot_encoded[row_index]) + 1
weighted_couple_dist = (w1 * w2v_dist + w2 * category_dist)/float(w1 + w2)
indices = np.argsort(weighted_couple_dist.flatten())[0:num_similar_items].tolist()
df = pd.DataFrame({'publish_date': news_articles['date'][indices].values,
'headline':news_articles['headline'][indices].values,
'Weighted Euclidean similarity with the queried article': weighted_couple_dist[indices].ravel(),
'Word2Vec based Euclidean similarity': w2v_dist[indices].ravel(),
'Category based Euclidean similarity': category_dist[indices].ravel(),
'Categoty': news_articles['category'][indices].values})
print("="*30,"Queried article details","="*30)
print('headline : ',news_articles['headline'][indices[0]])
print('Categoty : ', news_articles['category'][indices[0]])
print("\n","="*25,"Recommended articles : ","="*23)
#return df.iloc[1:,[1,5]]
return df.iloc[1:,]
avg_w2v_with_category(528,10,0.1,0.8)
========== Queried article details ==========
headline : Universities Tell Applicants That Protesting Gun Violence Won't Affect Admissions
Categoty : EDUCATION
========== Recommended articles : ==========
publish_date headline Weighted Euclidean similarity with the queried article Word2Vec based Euclidean similarity Category based Euclidean similarity Categoty
1 2018-02-21 Texas District Says Students Protesting Gun Violence Will Get Suspended 0.969627 0.726643 1.0 EDUCATION
2 2018-04-02 Teachers Swarm Kentucky Capitol To Protest Pension Changes, School Budget Cuts 0.988327 0.894946 1.0 EDUCATION
3 2018-02-07 While Teachers Fight For Better Pay, West Virginia Lawmakers Discuss Opossums 0.995150 0.956346 1.0 EDUCATION
4 2018-04-16 Beyoncé Announces $100,000 In Scholarships For HBCU Students 0.995467 0.959200 1.0 EDUCATION
5 2018-04-04 Oklahoma Teachers Begin 110-Mile March To Protest Education Funding 0.999916 0.999242 1.0 EDUCATION
6 2018-02-06 Homeless Students, Destroyed Campuses, 'Invisible Injuries': What California Schools Learned From Recent Disasters 1.002493 1.022439 1.0 EDUCATION

Discussion on results

Generally, we assess similarity based on distance. If the distance is minimum then high similarity and if it is maximum then low similarity.

To calculate the distance, we need to represent the headline as a d-dimensional vector. Then we can find out the similarity based on the distance between vectors.

A Bag of Words(BoW) method represents the occurrence of words within a document. Here, each headline can be considered as a document and set of all headlines form a corpus.

Using BoW approach, each document is represented by a d-dimensional vector, where d is total number of unique words in the corpus. The set of such unique words forms the Vocabulary.

TF-IDF method is a weighted measure which gives more importance to less frequent words in a corpus. It assigns a weight to each term(word) in a document based on Term frequency(TF) and inverse document frequency(IDF).

TF(i,j) = (# times word i appears in document j) / (# words in document j)

IDF(i,D) = log_e(#documents in the corpus D) / (#documents containing word i)

weight(i,j) = TF(i,j) x IDF(i,D)

So, if a word occurs a greater number of times in a document but a smaller number of times in all other documents then its TF-IDF value will be high.

Conclusion

Article Recommendation was done with the help of content-based filtering. Compared to BoW method, here TF-IDF method recommends the articles with headline containing words like "employer", "fire", "flip" in top 5 recommendations and these words occur less frequently in the corpus. Disadvantages :-

Bow and TF-IDF method do not capture semantic and syntactic similarity of a given word with other words but this can be captured using Word embeddings.

Github Repository Link

https://github.com/Niranjan1112/Recommender-Systems

References

[1]https://medium.com/web-mining-is688-spring-2021/article-recommendation-system-using-python-8b0fec6e6de8

[2]https://link.springer.com/article/10.1007/s10462-021-10043-x

[3]https://www.google.com/url?sa=t&source=web&rct=j&url=https://arxiv.org/pdf/1707.00506&ved=2ahUKEwiKyPfmpLz5AhXoRmwGHbCzASoQFnoECDsQAQ&usg=AOvVaw3F_oBvH9GLpt5tIEG_-2Vh

[4]https://arxiv.org/pdf/1904.10367v2.pdf

[5]https://www.researchgate.net/publication/346669020_Research_Progress_of_News_Recommendation_Methodsods

[6]https://www.researchgate.net/publication/338724860_News_Recommendation_Systems_-_Accomplishments_Challenges_Future_Directio