Recommendation System for Anonymous Microsoft Web Data

Project Group 8 -

Busi Pallavi Reddy (013852800)
Maunil Swadas (013850122)
Sarthak Sugandhi (013848497)

Introduction

- Recommendation System for Anonymous Microsoft Web Data

- Problem Statement -

  Build a recommendation engine which recommends users with URLs that a user might be interested in.

Dataset Description

Number of Instances - 37711

Number of Attributes - 294

Attribute Characteristics - Categorical

Missing Values - N/A

Total Values - 11,087,034

Link to Dataset - https://archive.ics.uci.edu/ml/datasets/Anonymous+Microsoft+Web+Data

Dataset Description

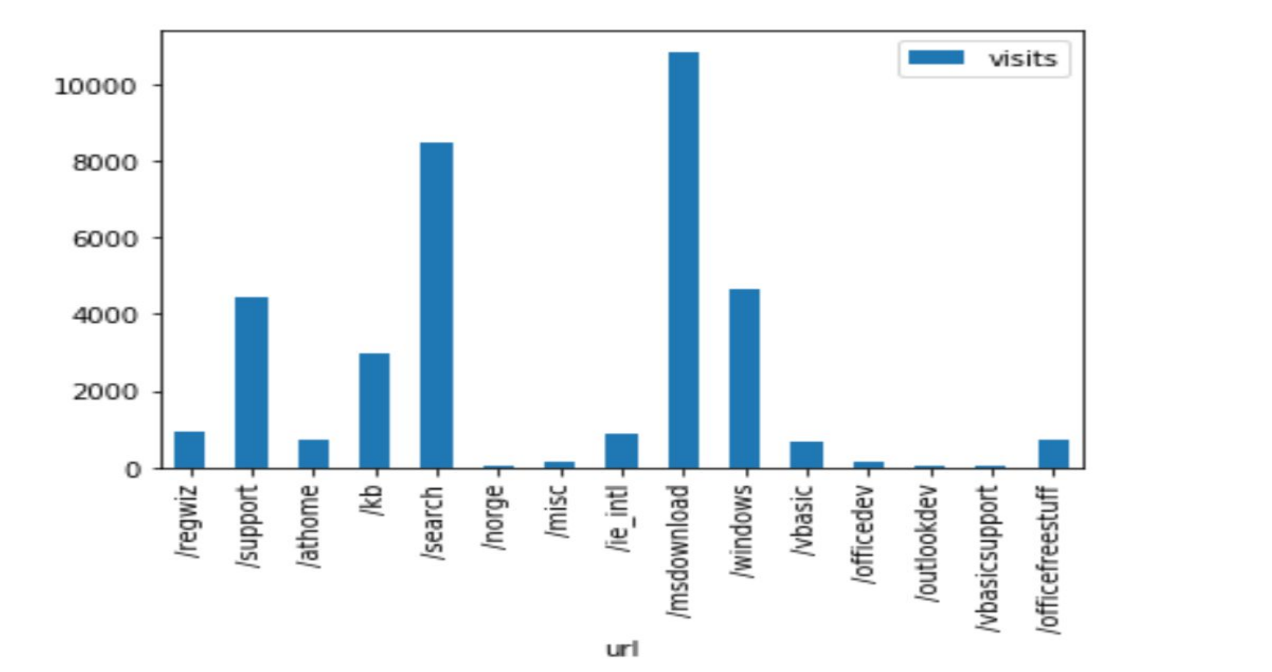
Attributes -

‘A’ - URL information

‘C’ - User Information, followed by

‘V’ - URLs visited by the above user defined in ‘C’ line

Dataset Description



Bar Graph showing URLs vs Number of Visits

Data Pre-Processing

- Utility Matrix of Users-Items

- Where each entry is either a
  - 1 (if a user has visited a URL) or
  - 0 (if a user hasn't visited the URL).

Models Used

- Item Based Collaborative Filtering
- User Based Collaborative Filtering
- Singular Value Decomposition(SVD)

Item Based Collaborative Filtering

In [11]:
```python
# Forming the item utility matrix
item_utility_matrix = data.pivot(index='user', columns='url', values='values').fillna(value=0)
item_utility_matrix.head()
```

Out[11]:

url	/access	/accessdev	/activeplatform	/activex	/athome	/australia	/automap	/backoffice	/brasil	/canada	/catalog	/(
user												
10001	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	C
10002	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	C
10003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	C
10005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	C
10006	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	C

5 rows × 104 columns

Item Based Collaborative Filtering

```
In [12]: # Recommending similar items to URL '/athome'
         item = item_utility_matrix['/athome']
         similarItems = item_utility_matrix.corrwith(item)
         similarItems.sort_values(ascending=False).head()

Out[12]: url
         /athome             1.000000
         /support            0.076860
         /windowssupport     0.067837
         /moneyzone          0.056557
         /windows            0.050309
         dtype: float64
```

User Based Collaborative Filtering

In [13]:
```
# Forming the user utility matrix
user_utility_matrix = item_utility_matrix.T
user_utility_matrix.head()
```

Out[13]:

user	10001	10002	10003	10005	10006	10007	10008	10009	10010	10011	10012	10013	10014	10015	10016	1
url																
/access	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
/accessdev	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
/activeplatform	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
/activex	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
/athome	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

5 rows × 32301 columns

User Based Collaborative Filtering

```
In [14]: # Similar users to user 10011
         u = 10011
         user = user_utility_matrix[u]
         similarUsers = user_utility_matrix.corrwith(user, axis='index').dropna().sort_values(ascending=F
         alse)
         similarUsers.drop(u, inplace=True)
         similarUsers.head()

Out[14]: user
         21000    0.74000
         16662    0.70014
         17563    0.70014
         37130    0.70014
         21724    0.70014
         dtype: float64
```

User Based Collaborative Filtering

In [15]:

```
# Similar users recommended URLs
df = pd.DataFrame(similarUsers)
users = pd.DataFrame(user_utility_matrix[df.iloc[0].name])
users.columns=['visited']
users = users[users['visited'] == 1]
recommended_websites = pd.merge(users, popular_websites, left_index=True, right_on="url")
recommended_websites = recommended_websites.sort_values(by='visits', ascending=False)
recommended_websites.head()
```

Out[15]:

	visited	visits	ID	title	url
287	1.0	5330	1018	isapi	/isapi
212	1.0	5108	1017	Products	/products
78	1.0	4451	1001	Support Desktop	/support
138	1.0	287	1016	MS Excel	/excel

User Based Collaborative Filtering

```
In [16]: # Recommending the website to user which he or she has not visited
website_urls = recommended_websites['url']
visited_sites = pd.DataFrame(user[user == 1])
print('Similar users urls for recommendation:', list(website_urls))
print('URLs visited by user:', list(visited_sites.index.values))
print('Recommended websites:', list(set(website_urls).difference(set(visited_sites.index.value
s))))

Similar users urls for recommendation: ['/isapi', '/products', '/support', '/excel']
URLs visited by user: ['/excel', '/isapi', '/mspowerpoint', '/products']
Recommended websites: ['/support']
```

User Based Collaborative Filtering

For new users -

```
In [17]: # Recommend a user without history with the top visited websites
         new_user_recommendations = popular_websites.sort_values(by='visits', ascending=False).head()
         list(new_user_recommendations['url'])

Out[17]: ['/msdownload', '/ie', '/search', '/isapi', '/products']
```

SVD

In [10]:
```
U,sigma,VT = c_recommend.processSVD()
index = c_recommend.processSingularValue(sigma)
U,sigma,VT = c_recommend.processNewSVD(U,sigma,VT,index)
diag = c_recommend.getDiagonalMatrix(sigma)
```

In [11]:
```
# Get unrated items for the user
```

In [12]:
```
itemsToRate = c_recommend.getUnRatedItems()
```

In [13]:
```
# Construct matrix with reduced data
```

In [14]:
```
transforMatrix = c_recommend.getTransformedItems(U,diag)
```

In [15]:
```
# Predict recommendations using COSINE SIMILARITY
```

In [16]:
```
recommendations = c_recommend.predictRecommendation(itemsToRate,transforMatrix)
```

SVD

```
In [17]: ids = []
         table = c_recommend.processOriginalData("./datasets/anonymous-msweb.csv")
         print("Top 5 Recommended URLs for userID : ", userID)
         for item, score in recommendations:
             page_id = c_recommend.ratings_matrix.columns[item]
             ids.append(page_id)
             print(table.loc[page_id][1])

         Top 5 Recommended URLs for userID :  200
         /support
         /athome
         /kb
         /search
         /norge
```

SVD

```
In [18]: # Predict recommendations using Pearson Correlation SIMILARITY
```

```
In [19]: userID = 200
predictionNumber = 5
p_recommend = Recommend("pearson",userID,predictionNumber)
p_recommend.loadData("./datasets/MS_ratings_matrix.csv")
U,sigma,VT = p_recommend.processSVD()
index = p_recommend.processSingularValue(sigma)
U,sigma,VT = p_recommend.processNewSVD(U,sigma,VT,index)
diag = p_recommend.getDiagonalMatrix(sigma)
itemsToRate = p_recommend.getUnRatedItems()
transforMatrix = p_recommend.getTransformedItems(U,diag)
recommendations = p_recommend.predictRecommendation(itemsToRate,transforMatrix)
ids = []
table = p_recommend.processOriginalData("./datasets/anonymous-msweb.csv")
print("Top 5 Recommended URLs for userID : ", userID)
for item, score in recommendations:
    page_id = p_recommend.ratings_matrix.columns[item]
    ids.append(page_id)
    print(table.loc[page_id][1])
```

```
Top 5 Recommended URLs for userID :  200
/support
/athome
/kb
/search
/norge
```

Evaluation Metrics

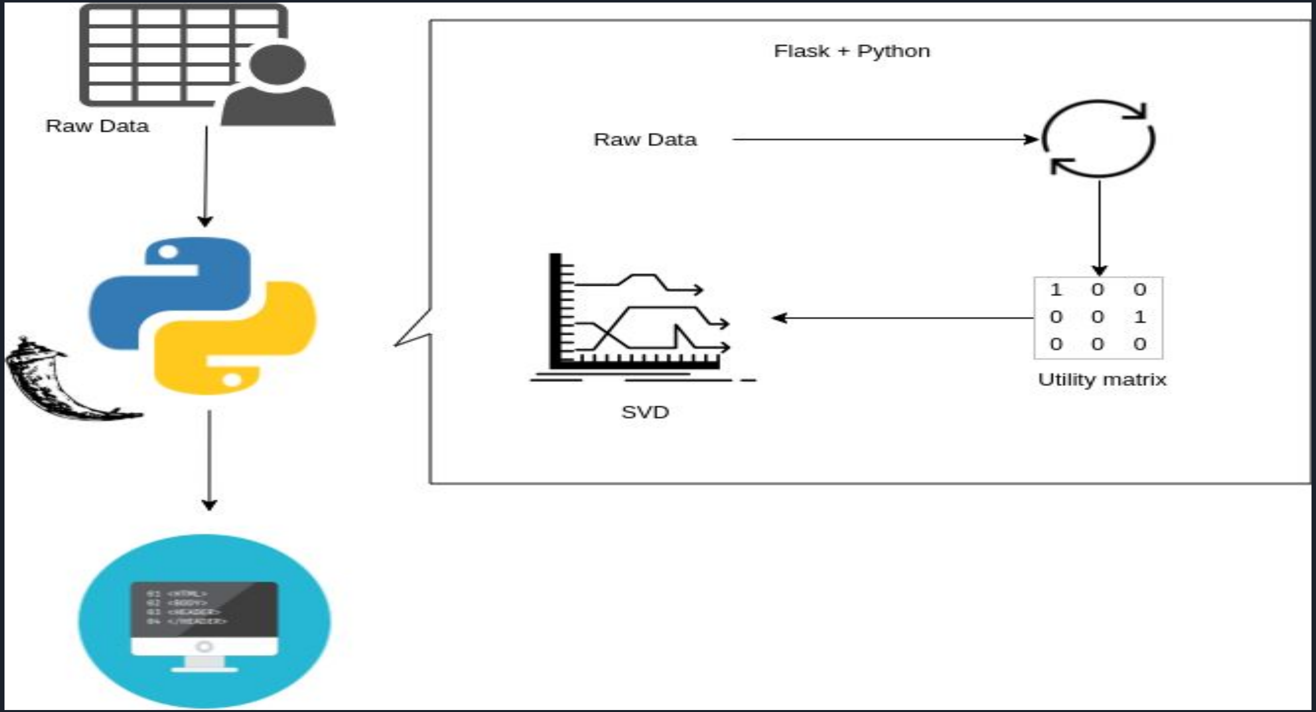
- RMSE for Evaluation (0.9-0.1 split on dataset)

- SVD with Pearson Correlation: 0.8745362102

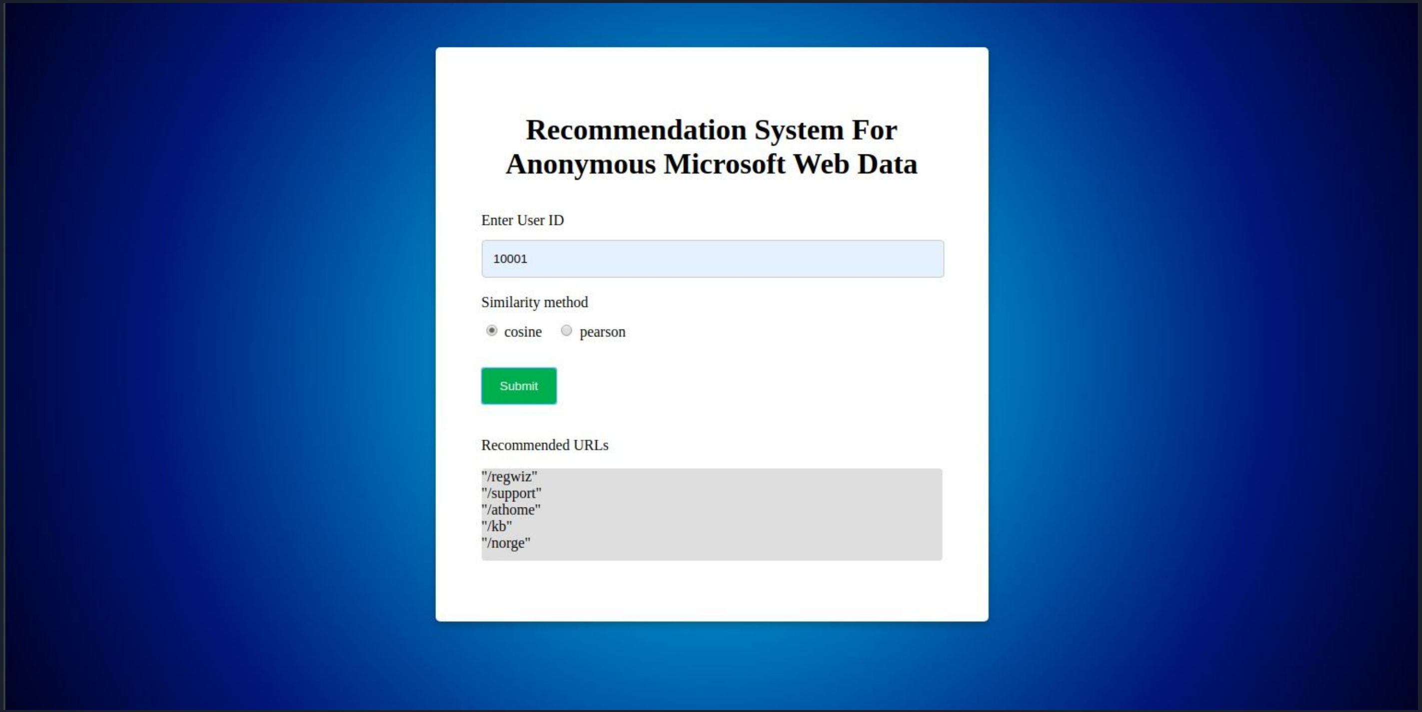
- User Based Collaborative Filtering: 0.9112323498

- Item Based Collaborative Filtering: 0.9111432323

Framework of the Deployed Application using SVD



UI Application



Conclusion

- Obtained better results using SVD
- Built a UI to view recommendations per user on the above model
- Explored the various recommendation models

THANKS!