

# Exercício de Programação 2

## Sistemas de Recomendação

Sheldon Goulart de Alcântara  
alcantarash92@gmail.com  
Vinícius Teodoro  
vtcpires@gmail.com

### Abstract

This report aims to present some variations of recommendation systems for scientific articles.

## 1 Sistemas de Recomendação

O sistema de recomendação é definido como uma estratégia de tomada de decisão para usuários em ambientes de informações complexas. Além disso, o sistema de recomendação foi definido a partir do ponto de vista do comércio eletrônico como uma ferramenta que ajuda os usuários a pesquisar através de registros de conhecimento relacionados ao interesse e preferência dos usuários. Definido como um meio de auxiliar e aumentar o processo social de usar recomendações de outros para fazer escolhas quando não há conhecimento pessoal suficiente ou experiência das alternativas. Os sistemas lidam com o problema da sobrecarga de informações que os usuários geralmente enfrentam, fornecendo-lhes recomendações de conteúdo e serviço personalizadas.

No trabalho em questão, foi adotado quatro estratégias para recomendação de artigos científicos:

- Aleatória.
- Mais popular.
- Baseada em Conteúdo.
- Filtragem colaborativa.

## 2 Aleatória

É recomendado um artigo independente da preferência do usuário.

```
1 import numpy as np
2 import pandas as pd
3 from random import randint
4
5 raw_data = pd.read_csv('raw-data.csv', encoding='iso-8859-1');
6 users = pd.read_csv('users.dat', header=None)
7 users[0] = users[0].str.split(' ')
8
9 def checkifRandom(vetor, index, y): #função para checar se j existe
10     no vetor
11     x = 0
12     for i in vetor[index]:
13         if int(i) == y:
14             x = 1
15             break
```

```

054 15     if x == 1:
055 16         checkifRandom(users[0], index, randint(0, len(vetor)))
056 17     else:
057 18         print('The recommendation for the user', index, 'is the article'
058 19         , y)
059 20         resultRandom.append(y) #coloca no result o resultado
060 21     return
061 22 def suggest_randomly(users): #função de sugestão aleatória
062 23     len_user = len(users[0])
063 24     for index, x in enumerate(users[0]):
064 25         y = randint(0, len_user)
065 26         checkifRandom(users[0], index, y)
066 27     return
067 28
068 29     Out (10 primeiros leitores):
069 30
070 31 ('The recommendation for the user', 0, 'is the article', 4747)
071 32 ('The recommendation for the user', 1, 'is the article', 3653)
072 33 ('The recommendation for the user', 2, 'is the article', 4201)
073 34 ('The recommendation for the user', 3, 'is the article', 2188)
074 35 ('The recommendation for the user', 4, 'is the article', 3072)
075 36 ('The recommendation for the user', 5, 'is the article', 678)
076 37 ('The recommendation for the user', 6, 'is the article', 3978)
077 38 ('The recommendation for the user', 7, 'is the article', 2915)
078 39 ('The recommendation for the user', 8, 'is the article', 2469)
079 40 ('The recommendation for the user', 9, 'is the article', 4884)

```

Listing 1: Recommender System Random

### 3 Mais Popular

Considerando o artigo mais lido no conjunto de usuários, é definido um ranking de popularidade entre os documentos lidos, e o mais popular é recomendado ao usuário.

```

085 1 import numpy as np
086 2 import pandas as pd
087 3
088 4 raw_data = pd.read_csv('raw-data.csv', encoding='iso-8859-1');
089 5 users = pd.read_csv('users.dat', header=None)
090 6 users[0] = users[0].str.split(' ')
091 7
092 8 articles = {}
093 9 for k in range(len(raw_data)):
094 10     articles[k+1] = 0
095 11
096 12 #Vetor para armazenar quantas vezes um artigo foi lido
097 13 for i in range(len(users)):
098 14     for j in users[0][i]:
099 15         if int(j) != 0:
100 16             articles[int(j)] = articles[int(j)] + 1
101 17
102 18 #Ordenando o vetor de artigos lidos
103 19 aux = []
104 20 articles_sorted = sorted(articles, key=articles.get, reverse=True)# "
105 21 ranking" dos artigos lidos
106 22 for r in articles_sorted:
107 23     print r, articles[r]
108 24     aux.append(r)
109 25
110 26 #Verifica se o usuário j leu o artigo e recomenda o próximo mais
111 27 popular
112 28 def checkReadMostPopular(vetor, index, position):

```

```

108 27 read = 0
109 28 for i in vetor[index]:
110 29     if int(i) == aux[position]:
111 30         read = 1
112 31         break
113 32 if read == 1:
114 33     position = position + 1
115 34     checkReadMostPopular(users[0], index, position)
116 35 else:
117 36     vetor[index].append(aux[position])
118 37     print('The recommendation for the user', index, 'is the article'
119 38         , aux[position])
120 39 #Fun o para recomendar o artigo mais popular
121 40 def suggest_mostPopular(users):
122 41     for index, x in enumerate(users[0]):
123 42         checkReadMostPopular(users[0], index, 0)
124 43     return
125 44
126 45 Out(10 primeiros leitores):
127 46
128 47 ('The recommendation for the user', 0, 'is the article', 10)
129 48 ('The recommendation for the user', 1, 'is the article', 10)
130 49 ('The recommendation for the user', 2, 'is the article', 10)
131 50 ('The recommendation for the user', 3, 'is the article', 10)
132 51 ('The recommendation for the user', 4, 'is the article', 10)
133 52 ('The recommendation for the user', 5, 'is the article', 10)
134 53 ('The recommendation for the user', 6, 'is the article', 10)
135 54 ('The recommendation for the user', 7, 'is the article', 14)
136 55 ('The recommendation for the user', 8, 'is the article', 10)
137 56 ('The recommendation for the user', 9, 'is the article', 10)

```

Listing 2: Most Popular

## 4 Baseada em Conteúdo

Um sistemas de recomendação baseado em conteúdo funciona com dados que o usuário fornece, nesse caso os artigos lidos pelo mesmo. Com base no abstract dos artigos lidos, foi usado o valor de TF-IDF associado ao método do vizinho mais próximo (kNN).

TF-IDF consiste numa medida estatística que tem o intuito de indicar a importância de uma palavra de um documento em relação a uma coleção de documentos.

Sendo assim, TF conta o número de vezes que um determinado termo ocorre em cada documento e soma esse valor. O número de vezes que um termo ocorre em um documento é a frequência do termo (Term Frequency - TF). Enquanto IDF é incorporado para diminuir o peso dos termos que ocorrem mais frequentemente no conjunto de textos selecionados, ao mesmo tempo que aumenta o peso daqueles que ocorrem raramente[1].

```

150 1 import numpy as np
151 2 import pandas as pd
152 3 from sklearn.feature_extraction.text import CountVectorizer,
153   TfIdfVectorizer
154 4 from sklearn.neighbors import NearestNeighbors
155 5 import numpy
156 6
157 7 raw_data = pd.read_csv('raw-data.csv', encoding='iso-8859-1', index_col='
158   doc.id');
159 8 users = pd.read_csv('users.dat', header=None)
160 9 users[0] = users[0].str.split(' ')
161 10
162 11 tfidf_vectorizer = TfIdfVectorizer()
163 12 raw_data['tfidf'] = list(tfidf_vectorizer.fit_transform(raw_data['raw.
164   abstract']))

```

```

162 13
163 14 tfidf_vectorizer = TfidfVectorizer(stop_words='english')
164 15 raw_data['tfidf'] = list(tfidf_vectorizer.fit_transform(raw_data['raw.
165 abstract']))
166 16
167 17 tfidf_matrix = tfidf_vectorizer.fit_transform(raw_data['raw.abstract'])
168 18 nbrs = NearestNeighbors(n_neighbors=2).fit(tfidf_matrix)
169 19
170 20 def get_similar_articles(doc_id):#Retorna o primeiro artigo mais similar
171 ao artigo lido
172 21 row = raw_data.index.get_loc(doc_id)
173 22 distances, indices = nbrs.kneighbors(tfidf_matrix.getrow(row))
174 23 names_similar = pd.Series(indices.flatten()).map(raw_data.
175 reset_index()['doc.id'])
176 24 return list(names_similar)[1:2]
177 25
178 26 def otherArticles(vetArtLidos):#Cria uma lista com os artigos similares
179 aos artigos lidos
180 27 listarticles = []
181 28 for i in vetArtLidos:
182 29     if (int(i) != 0):
183 30         similar_articles = get_similar_articles(int(i))
184 31         listarticles.append(similar_articles)
185 32 return list(set([y for x in listarticles for y in x]))
186 33
187 34 #Fun o para checar se o artigo recomendado j foi lido pelo usu rio
188 35 def checkReadBasedContent(vetor, index, articles, position):
189 36     read = 0
190 37     for i in vetor[index]:
191 38         if int(i) == articles[position]:
192 39             read = 1
193 40             break
194 41     if read == 1:
195 42         position = position + 1
196 43         checkReadBasedContent(users[0], index, articles, position)
197 44     else:
198 45         vetor[index].append(articles[position])
199 46         print('The recommendation for the user', index, 'is the article'
200 , articles[position])
201 47
202 48 return
203 49
204 50 #Fun o para recomendar o artigo baseado em conte do
205 51 def suggest_basedContent(users):
206 52     for index, x in enumerate(users[0]):
207 53         userArticles = otherArticles(x)
208 54         checkReadBasedContent(users[0], index, userArticles, 0)
209 55     return
210 56
211 57 Out(10 primeiros leitores):
212 58
213 59 ('The recommendation for the user', 0, 'is the article', 13703)
214 60 ('The recommendation for the user', 1, 'is the article', 9131)
215 61 ('The recommendation for the user', 2, 'is the article', 6337)
216 62 ('The recommendation for the user', 3, 'is the article', 803)
217 63 ('The recommendation for the user', 4, 'is the article', 12480)
218 64 ('The recommendation for the user', 5, 'is the article', 1920)
219 65 ('The recommendation for the user', 6, 'is the article', 13504)
220 66 ('The recommendation for the user', 7, 'is the article', 2209)
221 67 ('The recommendation for the user', 8, 'is the article', 1408)
222 68 ('The recommendation for the user', 9, 'is the article', 11265)

```

Listing 3: Content Based

## 5 Filtragem Colaborativa

A técnica de fatoração matricial é capaz de caracterizar tanto itens quanto usuários através de vetores de variáveis latentes inferidas de padrões de avaliação dos itens. Uma alta correspondência entre as variáveis latentes de um item e um usuário levam a uma recomendação[3].

Essa técnica está fortemente relacionada à decomposição de valores singulares (SVD, do inglês Singular Value Decomposition)

A idéia por trás de uma abordagem de SVD clássica é decompor uma matriz de avaliações  $P$  de dimensão em um produto de três matrizes,

$$P = A \Sigma B^T \quad (1)$$

onde  $A$  possui dimensão  $N \times N$ ,  $\Sigma$  possui dimensão  $N \times M$  e  $B$  possui dimensão  $M \times M$ . As matrizes  $A$  e  $B$  são ortogonais.  $\Sigma$  é uma matriz diagonal com  $k$  entradas diferentes de zero. Assim sendo, as dimensões efetivas das matrizes  $A$ ,  $\Sigma$  e  $B$  são,  $N \times K$ ,  $K \times K$  e  $K \times M$ , respectivamente. Estas  $k$  entradas diagonais da matriz  $\Sigma$  são todas positivas com  $\zeta_1 \geq \zeta_2 \geq \dots \zeta_k \geq 0$ . As colunas de  $A$  e  $B$  são, respectivamente, chamadas de autovetores a esquerda e a direita de  $P$ .

```
1 import pandas as pd
2 import numpy as np
3 from scipy.sparse.linalg import svds
4
5 users = pd.read_csv('users.dat', header=None)
6 raw_data = pd.read_csv('raw-data.csv', encoding='iso-8859-1');
7
8 article_df = pd.DataFrame(raw_data, columns = ['doc.id', 'raw.title', '
          raw.abstract'])
9
10 article_df = article_df.rename(index=str, columns={"doc.id": "ArticleID",
          "raw.title": "Title", "raw.abstract": "Abstract"})
11
12 users[0] = users[0].str.split(' ')
13 ratings = pd.read_csv('ratings.csv')
14
15 def fillRatings(ratings):
16     columns = ['userId', 'articleId', 'rating']
17     for index, user in enumerate(users[0]):
18         for article in user:
19             df = pd.DataFrame([[index, article, 5.0], [index, article
20             , 5.0]], columns=columns)
21             frames = [ratings, df]
22             ratings = pd.concat(frames, ignore_index=True)
23         return ratings
24
25 new_rate = fillRatings(ratings)
26 ax = new_rate.drop_duplicates()
27 ratings_df = pd.read_csv('ratings.csv')
28
29 ratings_df = ratings_df.append(ax, ignore_index=True)
30 ratings_df=ratings_df.rename(columns = {'userId': 'UserID', 'articleId': '
          ArticleID', 'rating': 'Rating' })
31
32 article_df['ArticleID'] = article_df['ArticleID'].apply(pd.to_numeric)
33
34 R_df = ratings_df.pivot(index = 'UserID', columns = 'ArticleID', values =
          'Rating').fillna(0)
35
36 R = R_df.as_matrix()
37 user_ratings_mean = np.mean(R, axis = 1)
38 R_demeaned = R - user_ratings_mean.reshape(-1, 1)
39
40 U, sigma, Vt = svds(R_demeaned, k = 50)
41 sigma = np.diag(sigma)
```

```

270 41
271 42 all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt) +
272 43 user_ratings_mean.reshape(-1, 1)
273 44 preds_df = pd.DataFrame(all_user_predicted_ratings, columns = R_df.
274 45 columns)
275 46
276 47 def recommend_article(predictions_df, userID, article_df,
277 48 original_ratings_df, num_recommendations=1):
278 49
279 50     # Pega e classifica as predições do usuário
280 51     user_row_number = userID - 1 # Assumindo que o ID do usuário se
281 52     inicia em 1
282 53     sorted_user_predictions = predictions_df.iloc[user_row_number].
283 54     sort_values(ascending=False)
284 55
285 56     # Pega os dados do usuário e mescla com as informações do artigo.
286 57     user_data = original_ratings_df[original_ratings_df.UserID == (userID
287 58 )]
288 59     user_full = (user_data.merge(article_df, how = 'left', left_on = '
289 60 ArticleID', right_on = 'ArticleID').
290 61 sort_values(['Rating'], ascending=False)
291 62 )
292 63
293 64     # Recomenda os artigos melhores avaliados e que não foram lidos pelo
294 65     usuário
295 66     recommendations = (article_df[~article_df['ArticleID'].isin(user_full
296 67 ['ArticleID'])]).
297 68     merge(pd.DataFrame(sorted_user_predictions).reset_index(), how =
298 69 'left',
299 70 left_on = 'ArticleID',
300 71 right_on = 'ArticleID').
301 72 rename(columns = {user_row_number: 'Predictions'}).
302 73 sort_values('Predictions', ascending = False).
303 74     iloc[:num_recommendations, :-1]
304 75 )
305 76
306 77     return user_full, recommendations
307 78
308 79 for index, user in enumerate(users[0]):
309 80     already_rated, predictions = recommend_article(preds_df, index+1,
310 81 article_df, ratings_df, 1)
311 82     print('The recommendation for the user', index, 'is the article:',
312 83 predictions)
313 84
314 85 Out (10 primeiros leitores):
315 86
316 87 The recommendation for the user 0 is the article: ArticleID
317 88 Title \
318 89 0 1 The metabolic world of Escherichia coli is not...
319 90
320 91 The recommendation for the user 1 is the article: ArticleID
321 92 Title \
322 93 0 1 The metabolic world of Escherichia coli is not...
323 94
324 95 The recommendation for the user 2 is the article: ArticleID
325 96 Title \
326 97 0 1 The metabolic world of Escherichia coli is not...
327 98
328 99 The recommendation for the user 3 is the article: ArticleID
329 100 Title \
330 101 0 1 The metabolic world of Escherichia coli is not...
331 102
332 103 The recommendation for the user 4 is the article: ArticleID
333 104 Title \
334 105 0 1 The metabolic world of Escherichia coli is not...

```

```

324 89
325 90 The recommendation for the user 5 is the article: ArticleID
326                                Title \
327 0          1 The metabolic world of Escherichia coli is not...
328 92
329 93 The recommendation for the user 6 is the article: ArticleID
330                                Title \
331 0          1 The metabolic world of Escherichia coli is not...
332 95
333 96 The recommendation for the user 7 is the article: ArticleID
334                                Title \
335 0          1 The metabolic world of Escherichia coli is not...
336 98
337 99 The recommendation for the user 8 is the article: ArticleID
338                                Title \
339 0          1 The metabolic world of Escherichia coli is not...
340 101
341 102 The recommendation for the user 9 is the article: ArticleID
342                                Title \
343 0          1 The metabolic world of Escherichia coli is not...
344 103

```

Listing 4: Collaborative Filtering

## 6 References

- [1] Beginners Guide to learn about Content Based Recommender Engines. [https :  
//www.analyticsvidhya.com/blog/2015/08/beginners - guide - learn - content - based -  
recommender - systems/](https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/)
- [2] Content-based filterin. [http : //recommender - systems.org/content - based - filtering/](http://recommender-systems.org/content-based-filtering/)
- [3] Role of Matrix Factorization Model in Collaborative. [https : //arxiv.org/pdf/1503.07475.pdf](https://arxiv.org/pdf/1503.07475.pdf)
- [4] Modelos de Fatoração Matricial. [https : //www.maxwell.vrac.puc - rio.br/19273/192734.pdf](https://www.maxwell.vrac.puc-rio.br/19273/192734.pdf)