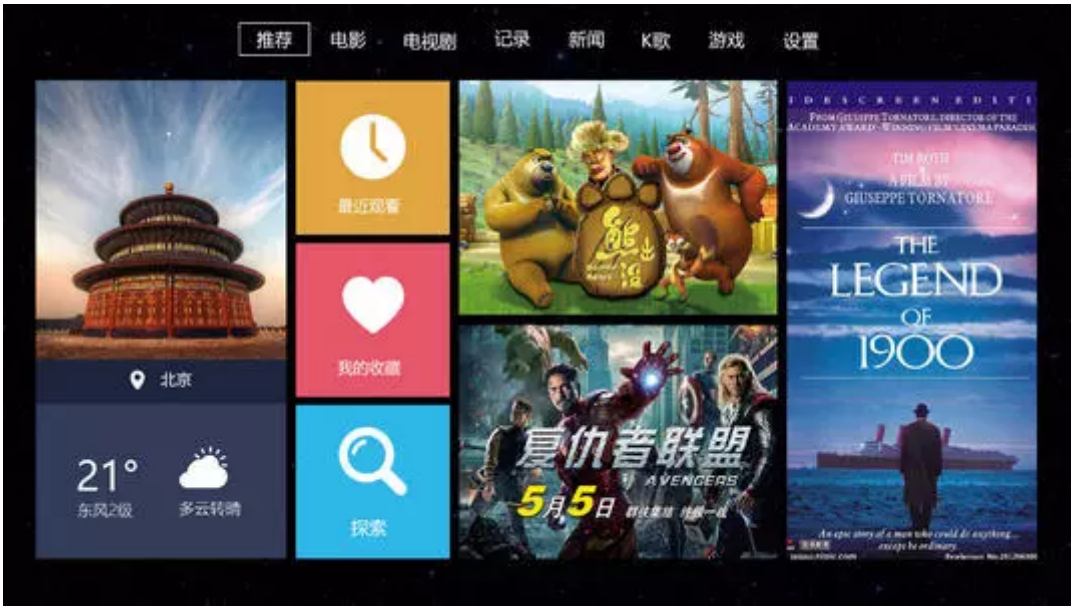
一文精通：用NLP实现基于内容的推荐系统

原创 读芯术 读芯术 2019-12-19

全文共**3490**字，预计学习时长**10**分钟



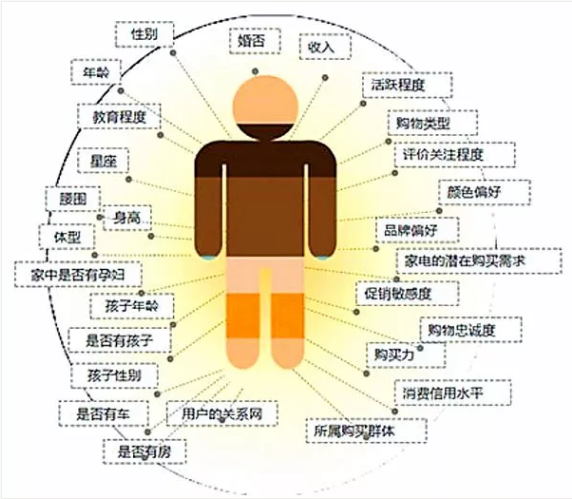
来源：zcool

不光是淘宝、京东会根据消费者的浏览、购买记录进行产品、服务推荐。网易云音乐、哔哩哔哩、腾讯新闻等各大音乐视频新闻等都会根据用户的记录进行相关推荐，一推一个准，令人方便之余，细思极恐。

来源：zhongmicm

下面，就和小芯一起来了解一下这个“神奇的推荐系统”。

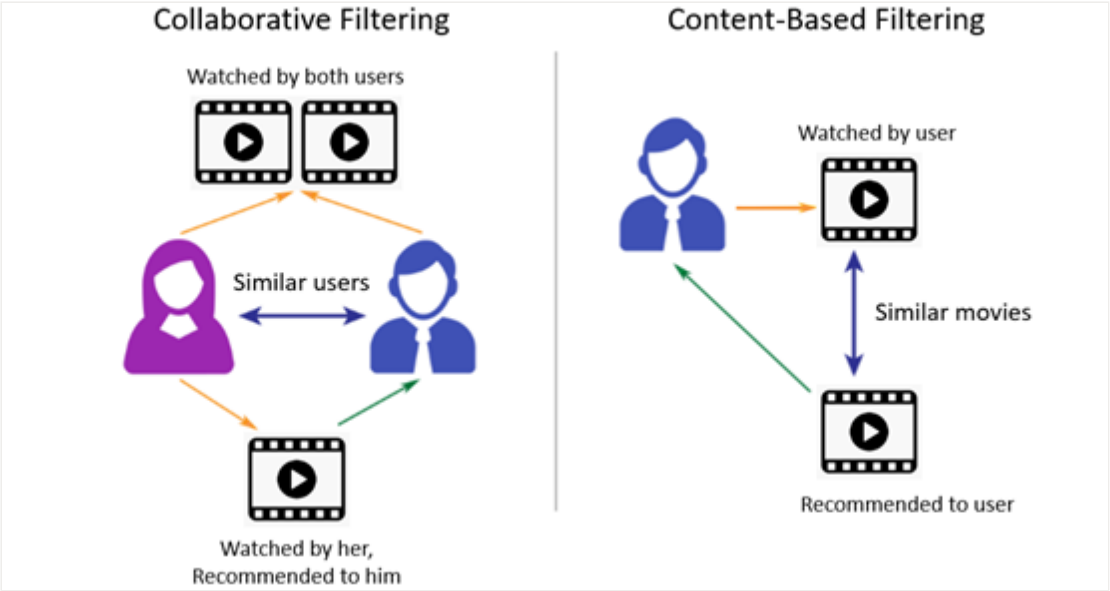
当在网上对产品和服务进行评级时，人们表达的所有偏好和共享的数据(明确地或不明确地)都会被推荐系统用来生成推荐。



来源：cbdio

推荐系统分为两类:

- 协同过滤-基于用户评分和消费将相似的用户分组，然后向用户推荐产品/服务
- 基于内容的过滤——根据相似的产品/服务的属性向用户推荐。



在本文中，小芯将结合电影属性(如类型、情节、导演和主要演员)来计算它与另一部电影的余弦相似性。数据集来源于data.world下载的IMDB最值得观看的前250部英文电影。

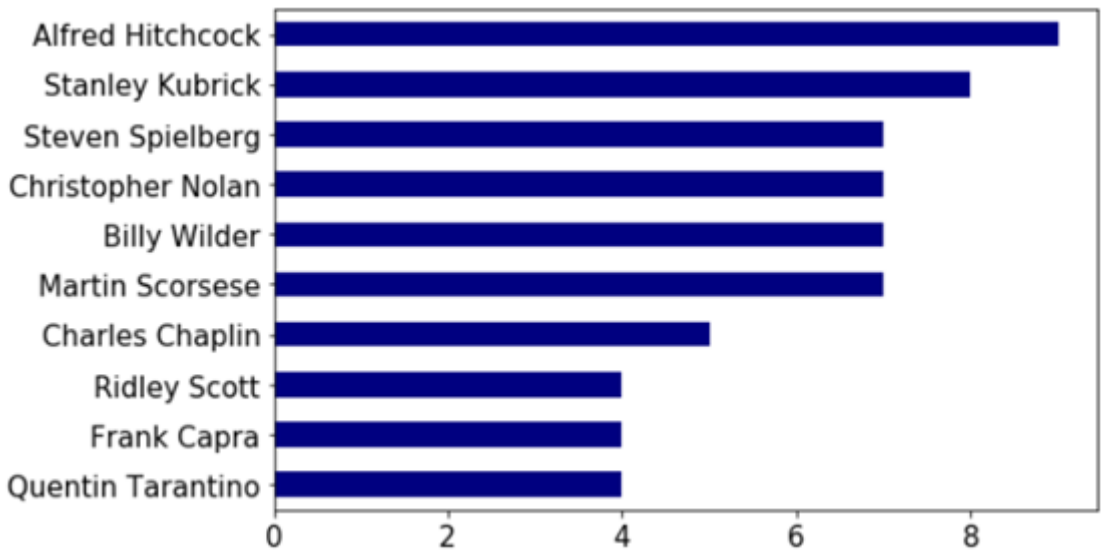
步骤1:导入Python库和数据集，执行EDA

确保已经安装了快速自动关键字提取(RAKE)库(或pip installrake_nltk)。

```
from rake_nltk import Rake
importpandas as pd
importnumpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import
CountVectorizerdf =pd.read_csv('IMDB_Top250Engmovies2_OMDB_Detailed.csv')
df.head()
```

在数据集中，有250部电影（行）和38个属性（列）。然而，只有5个属性是有用的:“Title”、“Director”、“Actors”、“Plot”和“Genre”。以下是10位最受欢迎的导演。

```
df['Director'].value_counts()[0:10].plot('barh',figsize=[8,5],
fontsize=15, color='navy').invert_yaxis()
```



250部电影中最受欢迎的10位导演

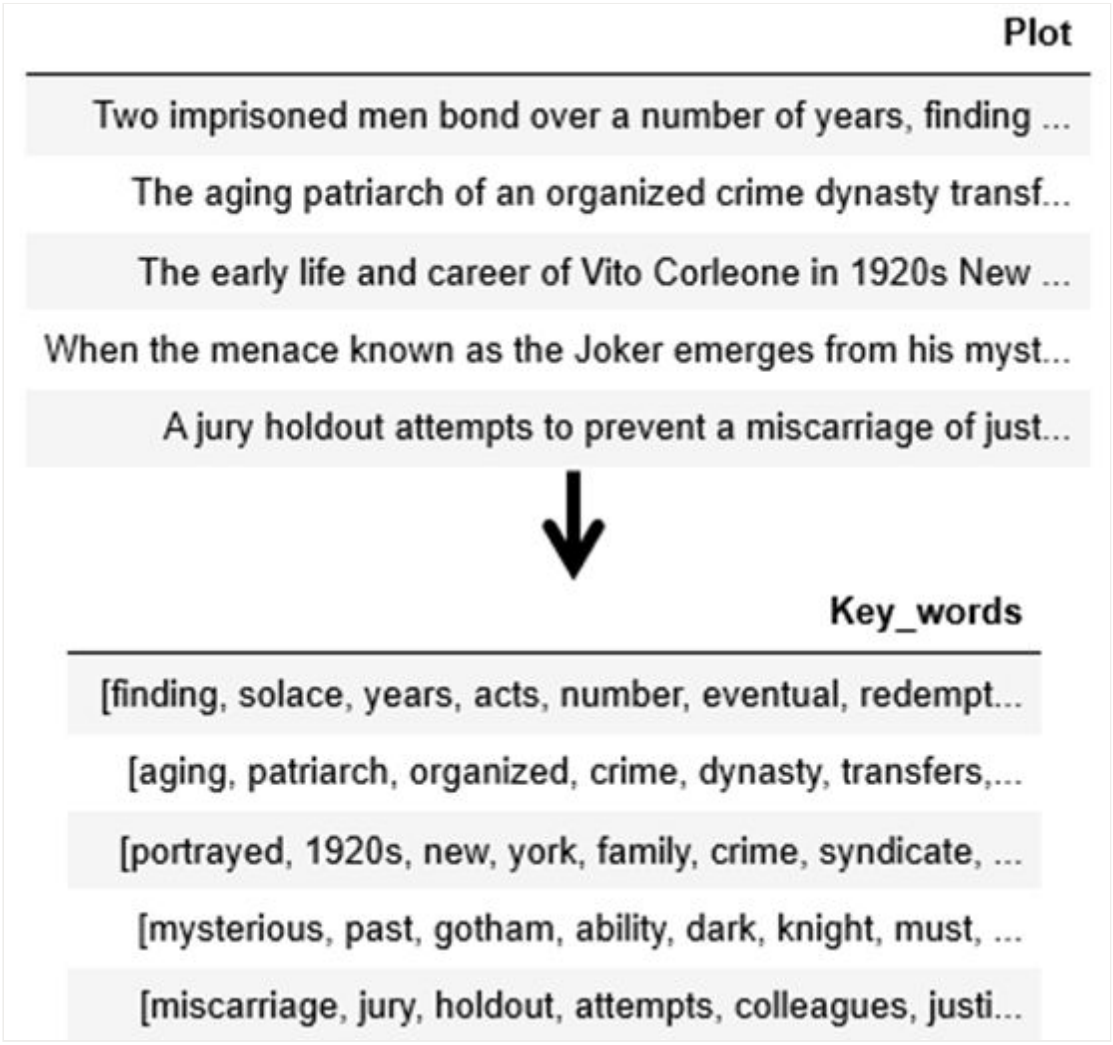
步骤2:数据预处理，删除停用词、标点符号、空白，并将所有单词转换为小写

首先，必须使用自然语言处理对数据进行预处理，得到包含每部电影所有属性(以单词表示)的一列。之后，以向量化的方式将这些信息转换为数字，并为每个单词分配分数，随后计算余弦

相似性。

使用Rake函数从“Plot”列的整个句子中提取最相关的单词。为此，将这个函数应用于“Plot”列下的每一行，并将关键词列表分配给一个新列“Key_words”。

```
df['Key_words']= ''
r =Rake()for index, row in df.iterrows():
    r.extract_keywords_from_text(row['Plot'])
    key_words_dict_scores =r.get_word_degrees()
    row['Key_words'] =list(key_words_dict_scores.keys())
```



Rake函数提取关键词

演员和导演的名字被转换成独有的身份值。这是通过将所有姓和名合并成一个单词来实现的，这样克里斯·埃文斯(Chris Evans)和克里斯·海姆斯沃斯(Chris Hemsworth)就会有不同的值

(如果不做这一步，他们将有50%的相似度，因为他们都有“克里斯”)。每个单词都需要转换成小写字母以避免重复。

```
df['Genre']= df['Genre'].map(lambda x: x.split(','))
df['Actors']= df['Actors'].map(lambda x: x.split(',')[:3])
df['Director']=  df['Director'].map(lambda  x:  x.split(','))for  index,  row  in
df.iterrows():
    row['Genre'] = [x.lower().replace(' ','')for x in row['Genre']]
    row['Actors'] = [x.lower().replace(' ','')for x in row['Actors']]
   row['Director'] = [x.lower().replace(' ','') for x in row['Director']]
```

Director	Actors	Genre
Frank Darabont	Tim Robbins, Morgan Freeman, Bob Gunton, William Sadler	Crime, Drama
Francis Ford Coppola	Marlon Brando, Al Pacino, James Caan, Richard S. Castellano	Crime, Drama
Francis Ford Coppola	Al Pacino, Robert Duvall, Diane Keaton, Robert De Niro	Crime, Drama
Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart, Michael Caine	Action, Crime, Drama
Sidney Lumet	Martin Balsam, John Fiedler, Lee J. Cobb, E.G. Marshall	Crime, Drama

Director	Actors	Genre
[frankdarabont]	[timrobbins, morganfreeman, bobgunton]	[crime, drama]
[francisfordcoppola]	[marlonbrando, alpacino, jamescaan]	[crime, drama]
[francisfordcoppola]	[alpacino, robertduvall, dianekeaton]	[crime, drama]
[christophernolan]	[christianbale, heathledger, aaroneckhart]	[action, crime, drama]
[sidneylumet]	[martinbalsam, johnfiedler, leej.cobb]	[crime, drama]

所有的名字都被转换成独特的身份值

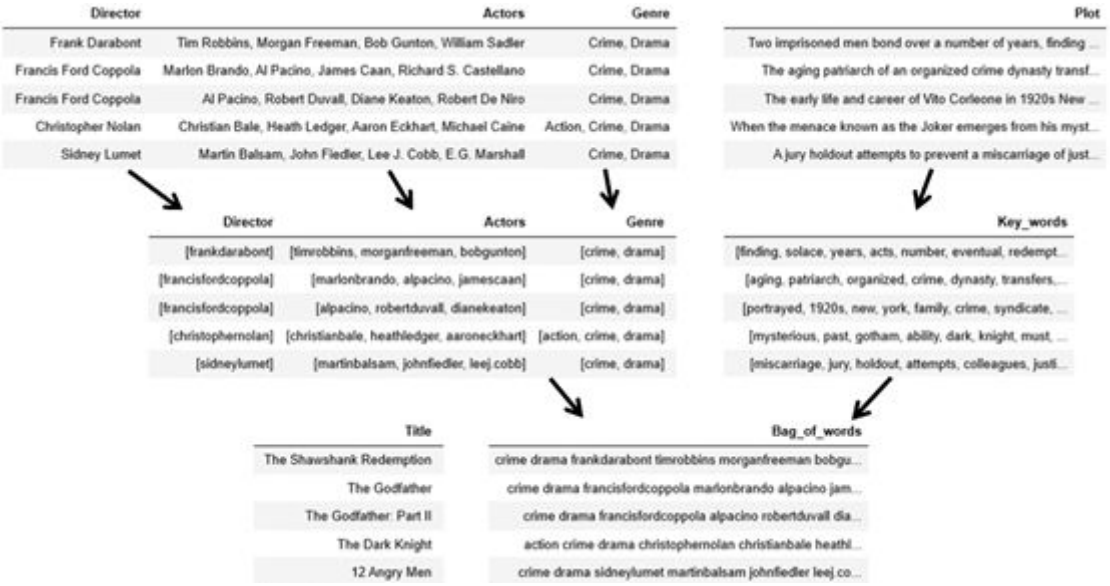
步骤3:合并列属性到新列Bag_of_words中来创建词汇表征

经过数据预处理后，“Genre”、“Director”、“Actors”和“Key_words”被合并为一个新的列“Bag_of_words”。最后的DataFrame只有两列。

```
df['Bag_of_words']= ''
columns=  ['Genre',  'Director',  'Actors',  'Key_words']for  index,  row  in
df.iterrows():
```

```
        words = ''
        for col in columns:
            words += ' '.join(row[col]) + ' '
        row['Bag_of_words'] = words


df =df[['Title','Bag_of_words']]
```



Director	Actors	Genre	Plot
Frank Darabont	Tim Robbins, Morgan Freeman, Bob Gunton, William Sadler	Crime, Drama	Two imprisoned men bond over a number of years, finding ...
Francis Ford Coppola	Marlon Brando, Al Pacino, James Caan, Richard S. Castellano	Crime, Drama	The aging patriarch of an organized crime dynasty transf...
Francis Ford Coppola	Al Pacino, Robert Duvall, Diane Keaton, Robert De Niro	Crime, Drama	The early life and career of Vito Corleone in 1920s New ...
Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart, Michael Caine	Action, Crime, Drama	When the menace known as the Joker emerges from his myst...
Sidney Lumet	Martin Balsam, John Fiedler, Lee J. Cobb, E.G. Marshall	Crime, Drama	A jury holdout attempts to prevent a miscarriage of just...

Director	Actors	Genre	Key_words
[frankdarabont]	[timrobbins, morganfreeman, bobgunton]	[crime, drama]	[finding, solace, years, acts, number, eventual, redempt...
[francisfordcoppola]	[marlonbrando, alpacino, jamescaan]	[crime, drama]	[aging, patriarch, organized, crime, dynasty, transfers,...
[francisfordcoppola]	[alpacino, robertduvall, dianekeaton]	[crime, drama]	[portrayed, 1920s, new, york, family, crime, syndicate, ...
[christophernolan]	[christianbale, heathledger, aaroneckhart]	[action, crime, drama]	[mysterious, past, gotham, ability, dark, knight, must, ...
[sidneylumet]	[martinbalsam, johnfiedler, leej cobb]	[crime, drama]	[miscarriage, jury, holdout, attempts, colleagues, justi...

Title	Bag_of_words
The Shawshank Redemption	crime drama frankdarabont timrobbins morganfreeman bobgu...
The Godfather	crime drama francisfordcoppola marlonbrando alpacino jam...
The Godfather: Part II	crime drama francisfordcoppola alpacino robertduvall dia...
The Dark Knight	action crime drama christophernolan christianbale heathl...
12 Angry Men	crime drama sidneylumet martinbalsam johnfiedler leej co...

最后的词汇表征是新的列“Bag_of_words”

步骤4:为Bag_of_words创建向量表示，并创建相似性矩阵

推荐模型只能读取一个向量（矩阵）并将其与另一个向量进行比较，所以需要使用CountVectorizer将'Bag_of_words'转换成向量表示，CountVectorizer可统计“Bag_of_words”列中每个单词的出现次数。一旦有了包含所有单词出现次数的矩阵，就可以用余弦相似性函数来比较电影之间的相似性。

$$similarity = cos(\theta) = \frac{u \cdot v}{\|u\|\|v\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

$$u \cdot v = [u_1\ u_2\ \dots\ u_n] \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = u_1 v_1 + u_2 v_2 + \dots + u_n v_n = \sum_{i=1}^{n} u_i v_i$$

根据余弦相似性公式来计算相似矩阵的值

```
count= CountVectorizer()
count_matrix= count.
fit_transform(df['Bag_of_words'])cosine_sim       =cosine_similarity(count_matrix,
count_matrix)
print(cosine_sim)
```

	Movie0	Movie1	Movie2	...	Movie247	Movie248	Movie249
Movie0	[1.	0.15789474	0.13764944	...	0.05263158	0.05263158	0.05564149]
Movie1	[0.15789474	1.	0.36706517	...	0.05263158	0.05263158	0.05564149]
Movie2	[0.13764944	0.36706517	1.	...	0.04588315	0.04588315	0.04850713]
...	...						
Movie247	[0.05263158	0.05263158	0.04588315	...	1.	0.05263158	0.05564149]
Movie248	[0.05263158	0.05263158	0.04588315	...	0.05263158	1.	0.05564149]
Movie249	[0.05564149	0.05564149	0.04850713	...	0.05564149	0.05564149	1.]

相似性矩阵（250行x 250列）

下一步是创建一系列电影标题，使系列索引能够匹配相似矩阵的行和列索引。

```
indices= pd.Series(df['Title'])
```

步骤5:运行并测试推荐模型

最后一步是创建一个函数，该函数将电影标题作为输入，并返回10部最相似的电影。该函数将输入的电影标题与相似矩阵的对应索引进行匹配，并按降序提取相似值行。提取前11个值并删除第一个索引(即输入影片本身)，可以找到前10部最相似的影片。

```
defrecommend(title, cosine_sim = cosine_sim):
    recommended_movies = []
    idx = indices[indices == title].index[0]
    score_series = pd.Series(cosine_sim[idx]).
sort_values(ascending= False)
    top_10_indices =list(score_series.iloc[1:11].index)
```

```
    for i in top_10_indices:
        recommended_movies.append(list(df['Title'])[i])

    return recommended_movies
```

现在准备测试模型。输入小芯最喜欢的电影《复仇者联盟》，看看推荐。

```
recommend('TheAvengers')
```

```
['Guardians of the Galaxy Vol. 2',
 'Aliens',
 'Guardians of the Galaxy',
 'The Martian',
 'Interstellar',
 'Blade Runner',
 'Terminator 2: Judgment Day',
 'The Thing',
 'The Terminator',
 'Spider-Man: Homecoming']
```

与《复仇者联盟》最相似的10部电影

结论

该模型推荐了非常相似的电影。根据小芯对漫威的了解，可以看到所推荐电影的导演和情节的相似之处。

现在，小芯已经看了大部分推荐的电影，并期待观看少数几部没看过的电影。

大家快来试试吧~