Beginners Guide to learn about Content Based Recommender Engines

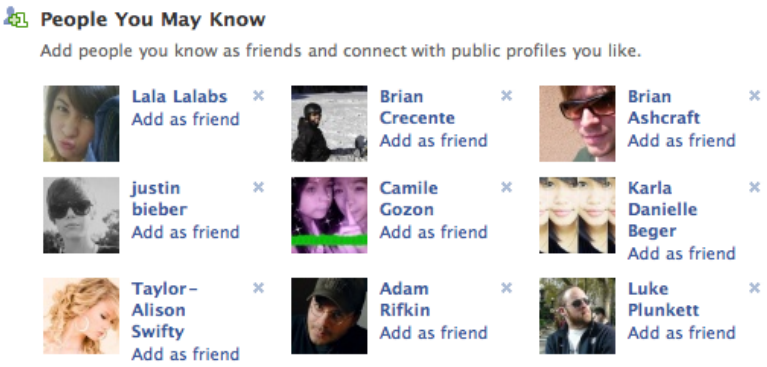
Shuvayan Das — Published On August 11, 2015 and Last Modified On September 24th, 2015

Algorithm Business Analytics Intermediate

Introduction

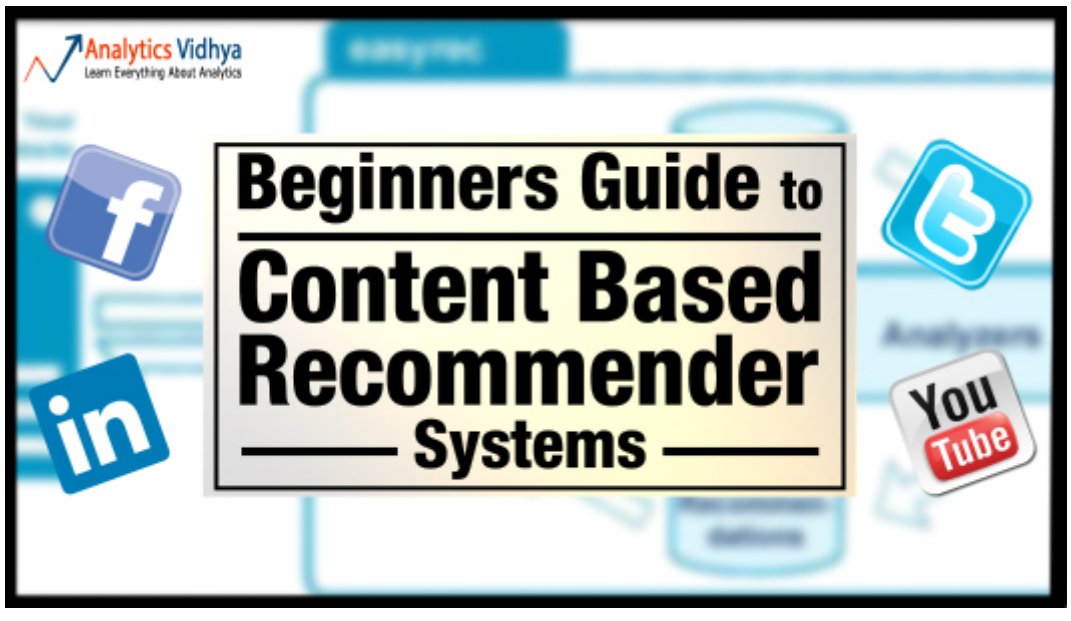
One of the most surprising part about Recommender Systems is, 'we summon to its suggestions / advice every other day, without even realizing that'. Confused? Let me show you some examples. Facebook, YouTube, LinkedIn are among the most used websites on Internet today. Let us see how they use **recommender systems**. You'll be amazed!

Facebook: Suggests us to make more friends using 'People You May Know' section



Similarly **LinkedIn** suggests you to connect with people you may know and **YouTube** suggests you relevant videos based on your previous browsing history. All of these are recommender systems in action.

While most of the people are aware of these features, only a few know that the algorithms used behind these features are known as 'Recommender Systems'. They 'recommend' personalized content on the basis of user's past / current preference to improve the user experience. Broadly, there are two types of recommendation systems – **Content Based & Collaborative filtering** based. In this article, we'll learn about content based recommendation system.



But before we proceed, let me define a couple of terms:

- **Item** would refer to content whose attributes are used in the recommender models. These could be movies, documents, book etc.
- **Attribute** refers to the characteristic of an item. A movie tag, words in a document are examples.

What are Content Recommender Systems?

According to Wikipedia:

> *"**Recommender systems** or **recommendation systems** (sometimes replacing "system" with a synonym such as platform or engine) are a **subclass of information filtering system** that seek to predict the 'rating' or 'preference' that user would give to an item."*

Recommender systems are active information filtering systems which **personalize the information** coming to a user based on his interests, relevance of the information etc. Recommender systems are used widely for recommending movies, articles, restaurants, places to visit, items to buy etc.

How do Content Based Recommender Systems work?

A content based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

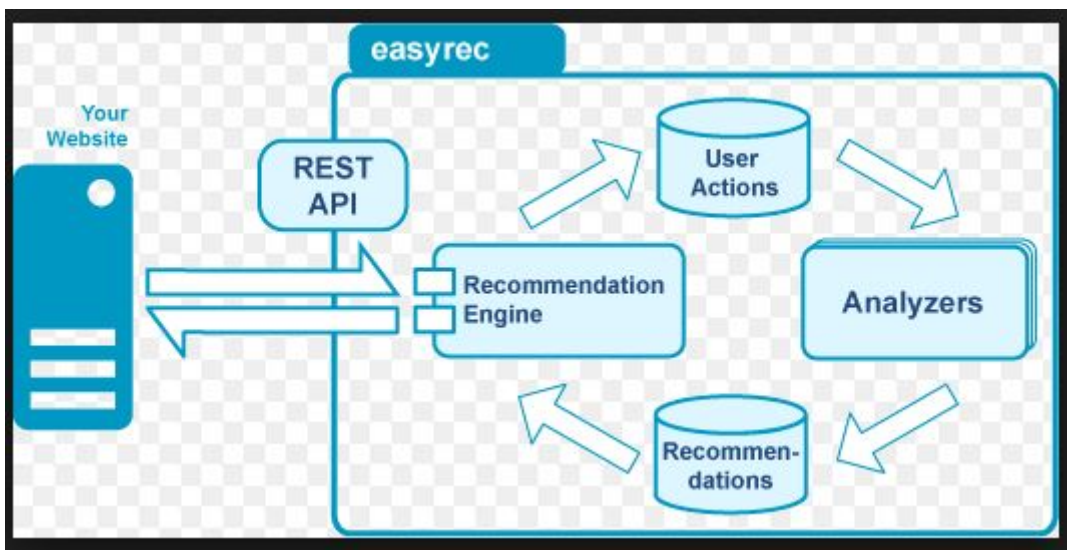


Image Source: easyrec.org

What are the concepts used in Content Based Recommenders?

The concepts of Term Frequency (***TF***) and Inverse Document Frequency (***IDF***) are used in information retrieval systems and also content based filtering mechanisms (such as a content based recommender). They are used to determine the relative importance of a document / article / news item / movie etc.

Term Frequency (TF) and Inverse Document Frequency (IDF)

TF is simply the frequency of a word in a document. IDF is the inverse of the document frequency among the whole corpus of documents. TF-IDF is used mainly because of two reasons: Suppose we search for ***"the rise of analytics"*** on Google**.** It is certain that "***the"*** will occur more frequently than "***analytics"*** but the relative importance of analytics is higher than the search query point of view. In such cases, TF-IDF weighting negates the effect of high frequency words in determining the importance of an item (document).

But while calculating TF-IDF, log is used to dampen the effect of high frequency words. For example: TF = 3 vs TF = 4 is vastly different from TF = 10 vs TF = 1000. In other words the relevance of a word in a document cannot be measured as a simple raw count and hence the equation below:

Equation:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

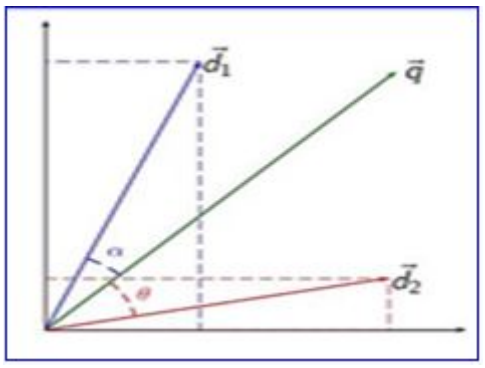
Term Frequency	Weighted Term Frequency
0	0
10	2
1000	4

It can be seen that the effect of high frequency words is dampened and these values are more comparable to each other as opposed to the original raw term frequency.

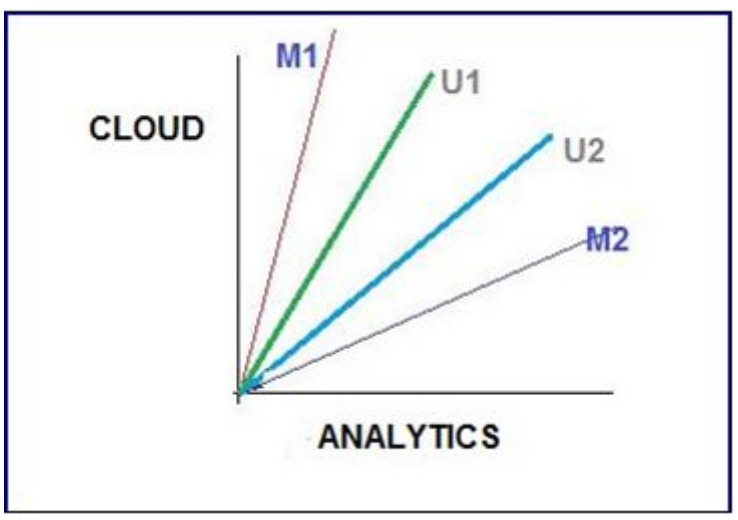
After calculating TF-IDF scores, how do we determine which items are closer to each other, rather closer to the user profile? This is accomplished using the **Vector Space Model** which computes the proximity based on the angle between the vectors.

How does Vector Space Model works?

In this model, each item is stored as a vector of its attributes (which are also vectors) in an **n-dimensional space** and the angles between the vectors are calculated to **determine the similarity between the vectors**. Next, the user profile vectors are also created based on his actions on previous attributes of items and the similarity between an item and a user is also determined in a similar way.



Lets try to understand this with an example.



Shown above is a 2-D representation of a two attributes, Cloud & Analytics. M1 & M2 are documents. U1 & U2 are users. The document M2 is more about Analytics than cloud whereas M1 is more about cloud than Analytics. I am sure you want to know how the relative importance of documents are measures. Hold on, we are coming to that.

User U1, likes articles on the topic 'cloud' more than the ones on 'analytics' and vice-versa for user U2. The method of calculating the user's likes / dislikes / measures is calculated by taking the cosine of the angle between the user profile vector ***(U_i)*** and the document vector.

The ultimate reason behind using cosine is that the **value of cosine will increase with decreasing value of the angle** between which signifies more similarity. The vectors are length normalized after which they become vectors of length 1 and then the cosine calculation is simply the sum-product of vectors. We will be using the function *sumproduct* in excel which does the same to calculate similarities between vectors in the next section.

Case study 1 – How to calculate TF – IDF ?

Let's understand this with an example. Suppose, we search for "***IoT and analytics***" on Google and the top 5 links that appear have some frequency count of certain words as shown below:

Articles	Analytics	Data	Cloud	Smart	Insight
Article 1	21	24	***0***	2	2
Article 2	24	59	2	1	***0***
Article 3	40	115	8	10	19
Article 4	4	28	5	***0***	1
Article 5	8	48	4	3	4
Article 6	17	49	8	***0***	5
DF	5,000	50,000	10,000	5,00,000	7000

Among the corpus of documents / blogs which are used to search for the articles, 5000 contains the word ***analytics***, 50,000 contain ***data*** and similar count goes for other words. Let us assume that the total corpus of docs is 1 million(10^6).

Term Frequency (TF)

Articles	Analytics	Data	Cloud	Smart	Insight	Length of Vector
Article 1	2.322219295	2.380211242	0	1.301029996	1.301029996	**3.800456039**
Article 2	2.380211242	2.770852012	1.301029996	1	0	**4.004460697**
Article 3	2.602059991	3.06069784	1.903089987	2	2.278753601	**5.380804488**
Article 4	1.602059991	2.447158031	1.698970004	0	1	**3.527276247**
Article 5	1.903089987	2.681241237	1.602059991	1.477121255	1.602059991	**4.257450611**
Article 6	2.230448921	2.69019608	1.903089987	0	1.698970004	**4.326697114**

As seen in the image above, for article 1, the term Analytics has a TF of **$1+ \log_{10}21 = 2.322$**. In this way, TF is calculated for other attributes of each of the articles. These values make up the attribute vector for each of the articles.

Inverse Document Frequency (IDF)

IDF is calculated by taking the logarithmic inverse of the document frequency among the whole corpus of documents. So, if there are a total of 1 million documents returned by our search query and amongst those documents, '***smart'*** appears in 0.5 million documents. Thus, it's IDF score will be: **Log_{10} (10^6/500000) = 0.30.**

DF	5000	50000	10000	500000	7000
IDF	2.301029996	1.301029996	2	0.301029996	2.15490196
N =	**10^6**				

We can also see (image above), the most commonly occurring term '**smart'** has been assigned the lowest weight by IDF. The length of these vectors are calculated as the **square root of sum of the squared values** of each attribute in the vector:

Articles	Analytics	Data	Cloud	Smart	Insight	Length of Vector
Article 1	2.322219295	2.380211242	0	1.301029996	1.3	=SQRT(J2^2+K2^2+L2^2+M2^2+N2^2)

After we have found out the TF -IDF weights and also vector lengths, let's normalize the vectors.

Articles	Analytics	Data	Cloud	Smart	Insight	Sum of Normalized Lengths
Article 1	0.61103701	0.626296217	0	0.342335231	0.342335231	1
Article 2	0.594389962	0.691941368	0.324895184	0.249721517	0	1
Article 3	0.483581962	0.568817887	0.353681311	0.371691632	0.423496822	1
Article 4	0.454191812	0.693781224	0.481666273	0	0.283504872	1
Article 5	0.447002246	0.62977624	0.376295614	0.34694971	0.376295614	1
Article 6	0.51550845	0.621766675	0.439848211	0	0.392671352	1

Each term vector is divided by the document vector length to get the normalized vector. So, for the term '**Analytics'** in Article 1, the normalized vector is: ***2.322/3.800***. Similarly all the terms in the document are normalized and as you can see (image above) each document vector now has a length of 1.

Now, that we have obtained the normalized vectors, let's calculate the cosine values to find out the similarity between articles. For this purpose, we will take only three articles and three attributes.

Articles	Analytics	Data	Cloud
Article 1	0.61103701	0.626296217	0
Article 2	0.594389962	0.691941368	0.324895184
Article 3	0.483581962	0.568817887	0.353681311

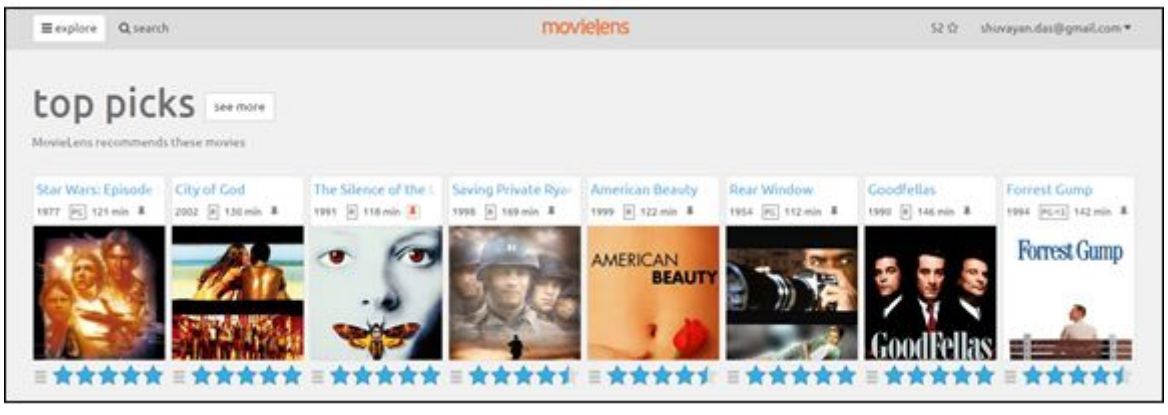
Article Vector	Cosine Values
cos(A1,A2)	0.796554526
cos(A2,A3)	0.795934246
cos(A1,A3)	0.651734967

Cos(A1,A2) is simply the sum of dot product of normalized term vectors from both the articles. The calculation is as follows:

```
Cos(A1,A2) = 0.611*0.59 + 0.63 * 0.69 + 0* 0.32 = =.7965
```

As you can see the articles 1 and 2 are most similar and hence appear at the top two positions of search results. Also, if you remember we had said that Article 1 (M1) is more about analytics than cloud-Analytics has a weight of 0.611 whereas cloud has 0 after length normalization.

Case study 2 – Creating binary representation



This picture is an example of movie recommender system called movielens. This is a **movie recommendation site** which recommends movies you should watch based on which movies you have rated and how much you have rated them.

At a rudimentary level, my inputs are taken in by the system and a profile is created against the attributes which contains my likes and dislikes (*may be based on some keywords / movie tags I have liked,or not done anything about).*

I have liked **52 movies** across various genres. These items (movies) are *attributes* since it can help us to determine a user's taste. Below is the list of top 6 movies recommended by movie-lens. The user columns specify whether a user has liked / disliked a recommended movie. The 1/0 simply represent whether the movie has adventure/action etc. or not. This type of representation is called a **binary representation of attributes**.

Movie	Adventure	Action	Science-Fiction	Drama	Crime	Thiller		User 1	User 2
Star Wars IV	1	1	1	0	0	0		1	-1
Saving Private Ryan	0	0	0	1	0	0			
American Beauty	0	0	0	1	0	0			
City of Gold	0	0	0	1	1	0		-1	1
Interstellar	0	0	1	1	0	0		1	
The Matrix	1	1	1	0	0	1			1

In the above example, user 1 has liked Star Wars whereas user 2 has not. Based on these types of inputs a user profile is generated.

Other metrics like count data, simple 1 / 0 representation are also used. The measure depends upon the context of use. For example: for **normal movie recommender system,** a simple binary representation is more useful whereas for a search query in search engine like google, a **count representation** might be more appropriate.

Putting it all together – Building a Content based recommender for Analytics Vidhya (AV)

Now that we have learned all these concepts, let's actually try to build a simple content based recommender based on user taste for AV. Many articles have been published on AV. Every article published caters to a segment of audience such that some people might like articles on machine learning algorithms, while other might like R over machine learning. Hence, having a recommender system would help.

To build the recommender system:

First Step: In each week, for each user, we will track user engagement (like / share / comments) with various articles. Let's see how that can be done:

Articles	Big Data	R	Python	Machine Learning	Learning Paths		Total attributes		User 1	User 2
Article 1	1	0	1	0	1		3		1	-1
Article 2	0	1	1	1	0		3		-1	1
Article 3	0	0	0	1	1		2			
Article 4	0	0	1	1	0		2			1
Article 5	0	1	0	0	0		1			
Article 6	1	0	0	1	0		2		1	
DF	2	2	3	4	2					

We have used binary representation here. The image shown above represents, Article 1 is about big data, python and learning path. Similarly, article 2 is about R, python and machine learning. User 1 has liked article 1 (shared it on social media) and has not liked article 2. He/She has not engaged with article 3,4,5 except reading them.

Second Step: Normalize

For binary representation, we can perform normalization by dividing the term occurrence(1/0) by the sqrt of number of attributes in the article. Hence, for Article 1: **normalized attribute = 1/sqrt(3) = 0.577.**

A question that you must ask here is: **what happened to TF? and why did we directly do normalization before calculating the TF scores?** If you calculate TF without normalization, the TF scores will be $\mathbf{1+\log_{10} 1 = 1}$ for attributes that occur and simply 0 for attributes that don't. And, thus the TF scores will also becomes 1/0.

Let's see how the user profiles are generated by using the 'sumproduct' function in excel.

Articles	Big Data	R	Python	Machine Learning	Learning Paths
Article 1	0.577350269	0	0.577350269	0	0.577350269
Article 2	0	0.577350269	0.577350269	0.577350269	0
Article 3	0	0	0	0.707106781	0.707106781
Article 4	0	0	0.707106781	0.707106781	0
Article 5	0	1	0	0	0
Article 6	0.707106781	0	0	0.707106781	0
User Profiles					
User1	1.28445705	-0.577350269	0	0.129756512	0.577350269
User2	-0.577350269	0.577350269	0.707106781	1.28445705	=SUMPRODUCT(F16:F21,$K$16:$K$21)

Total attributes	User 1	User 2
3	1	-1
3	-1	1
2		
2		1
1		
2	1	

SUMPRODUCT(array1, [array2], [array3], [array4], ...)

Each attribute column (see the highlighted learning paths columns above) is multiplied with each user attribute value (highlighted part of User 2). This is simply the dot product of vectors that we saw earlier. This gives a user profile (as shown in the User Profiles part above) an understanding of user's taste.

Thus, user 1 likes articles on big data most(highest score of 1.28) followed by learning paths and then machine learning. Similarly, user 2 like articles on machine learning the most.

Now we have the user profile vectors and the article vectors, let's use these to predict which articles will be similar to the user's taste.

Articles	big data	R	python	machine learning	learning paths
Article 1	0.577350269	0	0.577350269	0	0.577350269
Article 2	0	0.577350269	0.577350269	0.577350269	0
Article 3	0	0	0	0.707106781	0.707106781
Article 4	0	0	0.707106781	0.707106781	0
Article 5	0	1	0	0	0
Article 6	0.707106781	0	0	0.707106781	0
User Profiles					
User1	1.28445705	-0.577350269	0	0.129756512	0.577350269
User2	-0.577350269	0.577350269	0.707106781	1.28445705	-0.577350269
DF	2	2	3	4	2
IDF	0.698970004	0.698970004	0.52287875	0.397940009	0.698970004

User 1	User 2	Pred User1	PredUser2
1	-1	0.751333312	
-1	1	-0.20317834	
		0.321864985	
	1	0.036511676	
		-0.40355052	
1	=SUMPRODUCT(B36:F36,$B$39:$F$39,$B$43:$F$43)		

SUMPRODUCT(**array1**, [array2], [array3], [array4], [array5], ...)

Note : *The SUMPRODUCT function in the image above contains some vectors which are highlighted in the image. B36:F36 is the article 6 vector.*

The dot product of article vectors and IDF vectors gives us the **weighted scores** of each article.These weighted scores are again used for a dot product with the user profile vector (user 1 here). This gives a probability that the user will like a particular article. For article 1, the probability is 75%.

This concept can be applied to 'n' articles and we can find out which article a user will like the most. Therefore, along with new articles in a week, a separate recommendation can be made to a particular user based on the articles which he hasn't read already.

Limitations of Content Based Recommender Systems

Content based recommenders have their own limitations. They are not good at capturing inter-dependencies or complex behaviors. For example: I might like articles on Machine Learning, only when they include practical application along with the theory, and not just theory. This type of information cannot be captured by these recommenders.

End Notes

In this article, we have seen two approaches to content based recommenders. They both use the ***TF-IDF*** weighting and vector space model implementations, albeit in different ways. So while the count data helped us understand the methodology of calculating the weighted scores of articles, the binary representation helped us understand how to calculate the scores for data represented as 1/0 and we also saw how user profiles are generated and predictions are made based on that.

I have intentionally left the column PredUser2 empty in case you would like to go ahead and calculate it.

Did you find this article useful ? Have you also worked on recommender systems? Share your opinions / views in the comments section below.

If you like what you just read & want to continue your analytics learning, subscribe to our emails, follow us on twitter or like our facebook page.

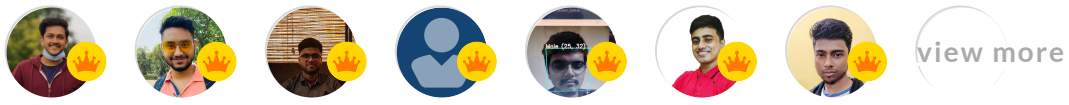
content recommender system inverse document frequency Microsoft Excel Term Frequency

About the Author

Shuvayan Das

I am Shuvayan Das, a B.Tech graduate having 4 years of experience in TCS as an SQL Server Developer/DBA. I am an analytics enthusiast. I began my journey in Analytics through a course in Jigsaw. A self - learner who believes that there just isn't enough time to learn but nevertheless we gotta keep trying . I have worked on SAS/R/SQL and currently I am focused on gaining extensive knowledge and experience in Analytics because "In god we trust,all others must bring data"-W.Edwards Deming.

Our Top Authors



Download

Analytics Vidhya App for the Latest blog/Article

Previous Post

Marketing Analytics: Essentials of Cross-Selling and Upselling (with a case study)

Next Post

7 Regression Techniques You Should Know!

23 thoughts on "Beginners Guide to learn about Content Based Recommender Engines"

Barbara McGillivray says:

August 12, 2015 at 9:15 am

Great summary, thanks! One thing I noticed in other articles on Analytics Vidhya as well is the dominant use of masculine pronouns to indicate a generic person. I think it would look better if instead of "his interests" and "his actions" you had "his/her interests" and "his/her actions".

Reply

Kunal Jain says:
August 12, 2015 at 7:38 pm

Barbara, Thanks for bringing this up. What you mention is true. From the perspective of readability of articles, it is better to keep a single pronoun rather than both. Hence, we have just stuck to a tone / style of writing than anything else. By no means this is a reflection of any gender bias / inequality. I promise that we will switch the gender completely in some of the articles to come. Regards, Kunal

Reply

Karthikeyan says:
August 13, 2015 at 1:22 pm

Awesome !! Awesome !!! Please write about collaborative filtering as well, with a similar demo.

Reply

Karthikeyan says:
August 13, 2015 at 3:24 pm

Hi Shuvayan, I have two questions 1. Why should a movie recommender system should consider a binary representation? Can a star rating give more insight whether a customer likes or dislikes rather than a binary value? 2. How do you measure whether a user has disliked a content? Is that based on star rating again? for example, if the user rates 3 and above then the content is good and anything less the content is bad?

Reply

Shuvayan Das says:
August 13, 2015 at 3:32 pm

Hi Karthikeyan, Glad that you liked it. Will try for collaborative also,though that is a bit more complex. :) 1.Movie recommendations are based more on ratings and reviews and tags applied to movies than a simple binary representation,though binary can also be used.I have used it just for illustration purpose. 2.You are right about the rating thing to measure user's dislike.However generally ratings are in increments of 0.5 like 1 , 1.5 etc and I guess below 2-2.5 is a pretty bad rating on a scale of 5.

Reply

Madhavan Sriram says:
August 17, 2015 at 11:37 am

Why is it that only three attributes are taken into consideration in the content recommendation study? Is there any particular reason for that?

Reply

Tugrul says:
August 18, 2015 at 5:30 am

Hi, It's a great article. Could you please explain how you calculated IDF values for your binary representation example "Building a Content based recommender for Analytics Vidhya (AV)" ? Thanks, Tugrul

Reply

Shuvayan Das says:
August 18, 2015 at 5:41 am

Hi Madhavan, The use of 3 attributes is only for ease of illustration as I have used excel and managing more was making it complex. :)

Reply

Madhavan Sriram says:
August 18, 2015 at 6:51 am

Ah okay. That makes sense :-)

Reply

Shuvayan Das says:
August 18, 2015 at 5:47 pm

Hi Tugrul, For that part IDF is log(10/DF).10 is the total corpus of documents(10 is just an assumption). :)

Reply

Tugrul says:
August 19, 2015 at 9:13 am

Thank you Shuvayan, that explains it. Tugrul

Reply

John Hui says:
August 19, 2015 at 10:00 pm

Hi, This is a good article. I am interested if there are examples of people using a mix of content based *AND* collaborative filtering for recommender systems. Is there anyway we can get the best of both worlds? John

Reply

Suresh Gorakala says:
October 25, 2015 at 6:54 am

Hi Would you please explain the last bit, the SUMPRODUCT(ARTICLE,IDF,USERPREF), how do I replicate in R code, Is it dotproduct(ARTICLE,IDF) then the result would be another dotproduct with UserPref?

Reply

Josh says:
August 22, 2016 at 2:36 pm

How can we choose the total corpus of documents(as in this case we took 10). Is there any way to find total corpus of documents? Can we take 13 which is the total of all the DF values? Thank you Josh

Reply

Christian Yonathan S. says:
September 04, 2016 at 1:25 pm

Hello, i would like ask, how to calculate the IDF in part Binary Encoding which score : 0.6987, 0.6987, 0.5228, 0.3979, and 0.6987? look likes your not mention it, so many thanks, Yohan

Reply

Christian Yonathan S. says:
September 05, 2016 at 3:42 am

Hello, how to calculate the IDF using binary encoding? BTW, why you delete my comment? so many thanks, Yohan

Reply

Shudhan says:
September 08, 2016 at 7:59 am

Hello Das, In case study 1, the calculated IDF was not used anywhere. Why are you calculating it then? Thanks!

Reply

Shudhan says:
September 08, 2016 at 8:03 am

Usual way. Log (total doc/#of docs containing the word) Here total doc in the corpus was assumed to be 10 For instance, IDF for the word 'analytics' is Log(10/2) = 0.69897

Reply

Aneesh says:
November 16, 2016 at 6:31 am

Why TF is not multiplying with the IDF in the first example given?

Reply

Valery says:
April 01, 2018 at 11:08 pm

great tutorial, thanks!

Reply

Aishwarya Singh says:
April 03, 2018 at 11:17 am

Hi, Glad you found this useful!

Reply

sanjiv thapa says:
April 22, 2018 at 7:37 pm

hello sir i am doing my college project about recommending colleges for SEE studetns. i am thinking about using content based filtering approach can you give some idea on this topic ?

Reply

Aishwarya Singh says:
April 23, 2018 at 2:09 pm

Hi Sanjiv, The article itself mentions some examples on content based filtering and presents a walk through. If you have any problem, you can always post your questions on the discuss portal.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

☑ Notify me of follow-up comments by email.

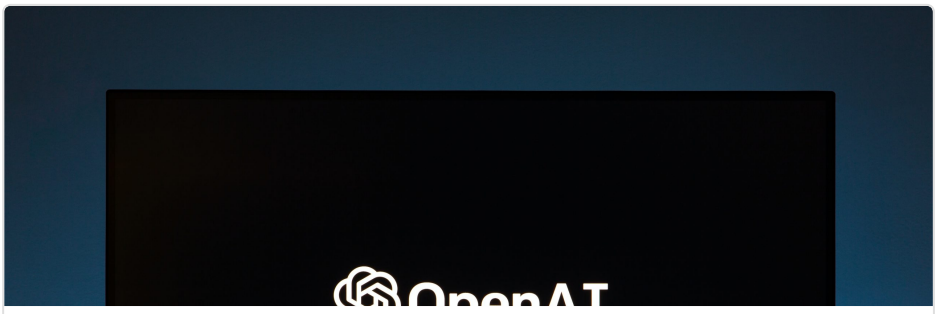
☑ Notify me of new posts by email.

Submit

Top Resources

Beginner's Guide to Build Your Own Large Language Models from..

Aravindpai Pai - JUL 05, 2023



OpenAI Provides Access For GPT-4

Sakshi Khanna - JUL 08, 2023

[Falcon AI: The New Open Source Large Language Model]

Ajay Kumar Reddy - JUL 06, 2023

[Building LLM-Powered Applications with LangChain]

Babina Banjara - JUL 05, 2023

Download App

Analytics Vidhya

About Us

Our Team

Careers

Contact us

Companies

Post Jobs

Trainings

Hiring Hackathons

Advertising

Data Scientists

Blog

Hackathon

Discussions

Apply Jobs

Visit us

Privacy Policy Terms of Use Refund Policy