ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse51952020.pdf
https://doi.org/10.30534/ijatcse/2020/51952020

Collaborative Filter Based Product Recommendation System using Machine Learning KNN and Cosine Similarity over Twitter by Big Data vide Ontology

Pinky Kashyap[1], Dr. Manish Kumar[2]
[1]M.Tech Student, Dept. of Computer Science & Engineering R.D. Engineering College at Duhai, Ghaziabad, India,prof.pkashyap@gmail.com
[2]Professor & Head of Department, Dept. of Computer Science & Engineering R.D. Engineering College at Duhai, Ghaziabad, India,manishtonk@gmail.com

ABSTRACT

Virtually all companies today are existentially dependent on escalating World Wide Web users are the reason by which data is in immense progression. The flow in information is a new dilemma for E-commerce supervisors of their products and services that are trepidation to be indistinguishable to clientele or users. The perfect step that can be in use is by employing a recommendation system. The recommendation system can display objects that are significant to the user. The recommendation system also needs to be seen. If seen from the problem of E-commerce that wants to display relevant products for customers, one method that can be applied is Content-Based combined with Collaborative Filtering, because the aspects that make these products recommended are not only based on the similarity of content but also based on user activity like giving a rating, seeing a product to buying it. However, this scheme is successfully achieved the accuracy of 81.11% percent which depicts that the same can be used for recommendations for E-Commerce products.

Key words: Machine Learning, Product Recommendation, Collaborative Filter, K-Nearest Neighbor, Cosine Similarity, Ontology.

1. INTRODUCTION

Advances in technology have made digital searches easier. Over time various sites that use search engines, whether it's selling sites or other sites also use a recommendation system. The recommendation system itself can be used in various areas, such as films, news, music, books, and so on. According to [1], the recommendation system utilizes the history of user behavior such as articles that have been read, products that have been rated or purchased, music that is played frequently, and so on to identify the user's preferences which then serve as a reference to produce final recommendations in the form of items or products. The recommendation system works based on user information that has been stored previously. This information itself can be in the form of numeric values, ordinal values, or binary values. The 3 main processes in the recommendation system, namely the collection and representation of data, the similarity decision, and computational recommendations [2]; According to [3] the recommendation system is divided into 5 types, namely content based, collaborative filtering, demographic, knowledge-based and hybrid recommender systems. The general principle of content-based filtering is to identify the general characteristics of an item that is highly valued by the user and recommend to the user the item that has the characteristics of the item. Whereas collaborative filtering uses information from users and other items; Demographic recommends items based on their demographic profile or region. Knowledge-based systems recommend items based on specific information on how large an item meets the needs and are useful for the user. Whereas the Hybrid recommender system is a combination of the recommendation systems above; From a variety of recommendation techniques, collaborative filtering has been proven to provide satisfying recommendations for users. Collaborative filtering itself is divided into 2 types, namely user-based collaborative filtering and item-based collaborative filtering. But here researchers will use user-based methods because this method can be said to represent us as users. It is said so because this method pays more attention to the similarity or similarity of the user than the goods that have been valued by users. Whereas item-based is more concerned with valuing goods; This method is included in the k-neighborhood model with cosine similarity that directly uses saved assessments to predict. Some of the advantages of using K-Nearest Neighbors methods and cosine similarity are:

1. *Simplicity*: neighborhood-based methods are relatively easy to implement
2. *Justifiability*: this method also provides a concise and intuitive truth base for computing predictions
3. *Efficiency*: one of the advantages of this method is its efficiency. Because this method does not require pre-computing and storage for data, the neighbors are not too large.
4. *Stability*: this method is not too affected by additional users, items and rating

2. LITERATURE SURVEY

2.1 Recommendation System

Recommender system (recommender system) is a technique that provides suggestions or suggestions for items of interest to certain users [3]. There are 3 main processes of this technique, namely: object data collections and representations, similarity decisions, and recommendation computation [3]. Some recommendation techniques, namely collaborative filtering, Content-based filtering, and Hybrid Recommender systems:

Collaborative Filtering: The collaborative filtering method is supported by congregation and analyzing large amounts of information about the user's behavior, actions, or inclinations and envisaging what the user will like. This method does not depend on content that can be analyzed. Therefore, this method can recommend complicated items such as films without requiring an understanding of the film itself. This is the advantage of the collaborative filtering method. A variety of algorithms have been used to measure the user's similarity or similarity of items in the recommendation system. For example, the k-nearest neighbor (k-NN) approach [4] and Pearson Correlation were first functional by [5]. One of the best known examples of collaborative filtering is item-to-item or also item-based (people who buy x also buy y), an algorithm popularized by the Amazon.com recommendation system [6]. Last FM recommends music based on evaluation of the same user's listening habits, while Readgeek compares book ratings for recommendations. Facebook, MySpace, LinkedIn and social network so thers use collaborative filtering to recommend friends, groups and other social connections (by checking the network of connections between users and their friends). Twitter uses many signals and calculations in memory to recommend to users who to follow [8].

In collaborative filtering there are also user-based methods, where this method will recommend items to users x which are also preferred by other users who are similar to x [3]. So between the user-based method and item-based method (item-to-item) is almost the same. However, the difference lies in what the recommendations refer to. User-based see the similarity of users with other users, while item-based views in terms of goods;

The collaborative filtering approach often experiences three problems, namely cold start, scalability, and sparsity. Cold start means that if the amount of data used is only a small amount then the recommendation results become less accurate, therefore this system often requires large amounts of data to the user to make accurate recommendations [7]. Scalability means that with the increase in the size of the dataset (millions of users and products) used to make recommendations, the amount of computational power needed to calculate recommendations will also increase [3]. Sparsity in other words is not all items in the dataset have a rating, this is what is called sparsity The collaborative filtering method is classified as memory-based and model-based. Examples of well-known memory-based approaches are user-based algorithms [9] and model-based approaches are Kernel-Mapping Recommendations [10].

Memory based techniques: This technique is divided into two, namely user-based and item-based (item-to-item) methods.

User-based: In the user-based technique each user will be counted in common with other users. After that, the rating from users who have a high degree of similarity with the predicted user will be used to calculate predictions.

Item-based: Broadly speaking, this technique is almost the same as user-based. The difference only lies in the search for similarities. If user-based is looking for similarities between each user, then as the name implies item-based is looking for similarities between goods and other goods. After that the same thing is done for the prediction calculation step.

2.2 Content-based Filtering

An additional general approach when scheming a recommendation system is content-based filtering. This method is based on item descriptions and preference profiles from users [11-13]. In content-based filtering, keywords are used to describe items, and then a user profile is created to indicate the type of items that this user likes. In other words, this algorithm tries to recommend items that are similar to what users have liked in the past. The similarity of goods is calculated based on the features associated with the items being compared [3]. So, the recommended items are various candidate items that have been compared with items that have been previously rated by the user. The tf-idf representation algorithm (also called vector space representation), is often used to create features from items. To create a user profile, most systems focus on two types of information, namely the user preference model and the history of user interaction with the recommendation system. Basically, this method uses item profiles that characterize items in the system. This system creates user-based content profiles based on vectors of item features. Weights indicate the importance of each feature to the user and can be calculated from content vectors that are ranked individually using various techniques. A simple approach uses the average value of the item vector being assessed, while other sophisticated methods use machine learning techniques such as Bayesian Classifiers, cluster analysis, decision trees, and artificial neural networks to estimate the probability that users like the item [14]. Direct feedback from users, usually in the form of likes or dislikes buttons, can be used to assign higher or lower weights regarding the importance of certain attributes (using Rocchio classification or other similar techniques).The main problem with content-based filtering is whether the system can learn user preferences from user actions regarding one content source and use it in other types of content. If the system can only recommend content of the same type (limited to recommending content of a different type) that is used by the user, the value of this system becomes less important. For example, recommending news articles based on news browsing is very useful but, it will be far more useful when music, videos, products, films, discussions, etc. from various services can be recommended based on news browsing.

2.3 Hybrid Recommender Systems

Hybrid approaches can be implemented in several ways, by making content-based and collaborative predictions separately and then combining them, by adding content-based capabilities to collaborative-based approaches or vice versa, by uniting several approaches into one model [15]. Several studies empirically compare hybrid performance with collaborative methods and based on pure content and show that hybrid methods can provide more accurate recommendations than pure approaches. This method can also be used to overcome some common problems in the recommendation system such as cold start and sparsity. For example, the collaborative filtering method has problems when recommending new items or when the goods have no value. This has no effect on the content-based method, because this method uses features or descriptions of the item to calculate recommendations.

One example of applying this method is Netflix [16]. This website makes recommendations by comparing viewing habits and searching habits of the same user (collaborative filtering) and also by offering films that have characteristics similar to films that have been highly rated by users (content-based filtering).

2.4 K-Nearest Neighbors

K-Nearest Neighbor (KNN) [17, 18] is a method to do classification of objects based on learning data closest to the object. Learning data is anticipated into multi-dimensional space, where each dimension represents the features of the records. The room is divided into sections that are divided in learning data. A point on this space is marked with class c if class c is the classification most often found in the object of the nearest neighbor the point. Near or neighbor counts are usually calculated based on Euclidean distances with formulas such as in equations as under

$$\boldsymbol{distance} = \sqrt{\sum_{i=1}^{n} (\boldsymbol{X^{i_{training}} - X_{testing}})^2}$$

With

$x^{i_{training}}$: i-training data;

$x_{testing}$: testing data;

i: ith record (row) from the table, n: amount of training data.

2.5 Cosine Similarity

Cosine similarity measures the similarity between 2 n-dimensional vectors based on their angles and is widely used in the information search field and text mining to compare two text documents. Where 1 indicates the same and 0 otherwise [19. 20]

$$\boldsymbol{s(\vec{u}, \vec{v})} = \frac{\vec{u}, \vec{v}}{|\vec{u}|, |\vec{v}|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} * \sqrt{\sum_{i=1}^{n} v_i^2}}$$

Where

s = similarity;

ui = Component of item u to i;

vi = Component item v to i;

3. PROPOSED METHODOLOGY

This segment contains the exploration strategy utilized in the examination and framework proposed. Therefore, next is the scenarios and steps taken for exploration which comprises of different stages which are portrayed in Figure beneath:-

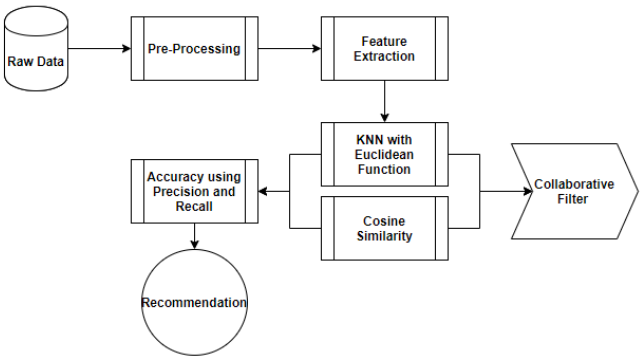

Figure 1: Proposed Scheme and Workflow

The k-NN method calculates similarities between products on the basis of their common buyers. In the training phase, the focus is on the calculation of heuristic similarity measures. In the application phase, the transaction history becomes I_u for each product *u* Based on the calculated neighborhood weights N_u (i)formed from the *k* most similar products. Multiple similarities to the same product are aggregated by summing. Thus the rank of a product *j for* the calculate customers *u* from the similarity weightsw_{ij}analogous to equation below.

$$\boldsymbol{rank(j)} = \sum_{\substack{i \in I_u \\ j \in N_u(i)}} \boldsymbol{w_{ij}}$$

3.1 Algorithm: kNN training process

Input: *basic training data record B (Customer, Product) Training Training target record Z (Customer, Product)*

Output: *Interestingness table l(Product A, Product B,Interesting value,Group table G(Product, Min_Score, Max_Score,probability)*

1: ***function*** *training (B,Z,k)*

2: *kNN WeightComputation (B,Z) #Calculates interestingness measures* w_{ij}

3: *#Output is the table of interest I*

4: ***For*** *User* $u \in U_B$ *# Quantity of customers from B*

5: ***For*** *Item* $i \in j_u^B$

6: $w_{ij} \leftarrow CorrespondWeights(i, k, l)$ *# determines corresponding rules*

7: *add* w_{ij} *to* w_u

8: ***For*** $w_{xj} \in w_u$ *# for all values with the same target product*

9: *Aggregate* w_{xj} *to* s_l

10: *Add* s_{uj} *to* s_j *# Quantity of scores for a target product*

11: *Add* j *to* j_u *#Quantity of target products for the customer u*

12: ***For*** $j \in j_u$

13: ***if****(u,j) in Z then*
14: $b_{uj} \leftarrow 1$
15: *else* $b_{uj} \leftarrow 0$
16: *Add* b_{uj} *to* B_j *# Quantity of purchase variables for a target product*
17: ***For*** $j \in j_Z$ *#Scores for target product to calculate Distance and discretized*
18: Calculate Euclidean Distance ($S_j . B_j$)
19: Discretize($S_j * B_j$) *#Output is the group table G*

Algorithm : Cosine Similarity from group table G as derived from K-NN.

1: ***function Weight_Similarity(G)***
2: ***For****i* $\in G$
3: ***For****u* $\in U_i$
4: ***For****j* $\in J_u$
5: $n_{ij} \leftarrow n_{ij} + 1$
6: *Add* j *to* S_i
7: ***For****j* $\in S_i$
8: Compute w_{ij}
9: return Sort w_{ij}

3.2 Evaluation and Validation

Performance evaluation is done to test the results of classification by measuring the truth value of the system. The statistical measure used to measure the value of system performance is accuracy. Accuracy is the percentage of texts that have been classified correctly by the system. Accuracy is obtained from the calculation results shown in Equation below.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Information:

TP = *True Positive*; is positive data which detected positive.

TN = *True Negative*; is the amount of negative data that is detected negatively.

FP = *False Positive;* is negative data however detected as positive data.

FN = *False Negative*; is positive data however detected as negative data.

4. RESULTS AND SIMULATION

The above section 3 have been adequately clarified, I will concentrate on a couple of usage viewpoints and specialized subtleties in this segment. In the frontal area is here the contribution of the framework, which depends on the K-NN and cosine similarity.

Data Schema Description: The Bookdatacomprisesthreetables, the users, books table and the book ratings table.

1.Schema description of table: users (User-ID;"Location";"Age") comprising of 272529 records.

2. Schema description of table: books (ISBN;"BookeTitle";"BookeAuthor";"YeareOf-Publication"; "Publisher";"ImageeURL-S";"ImageeURL-M";"ImageeURL -L") comprising of 271380 records.

3. Schema description of table: books_rating User-ID;"ISBN";"Book-Rating") comprising of 1048756 records.

Simulation:

Step1`: Pre-Processing

At the primary occurrence, the comma-isolated qualities or CSV records will be mounted on document arrangement of investigation from there on the pre-preparing will be started and highlight extraction will be shaped be figure.2 portrays the situation where the "BX-clients" table is perused from the record framework and from "BX-client" table will be assessed, in the preprocessing stage just the necessary sections and tuples are extricated rest of information or tuples are not perused for instance UserId, Location and Age are the points of view and for highlights 'userID', 'ISBN', 'booking' in "BX-Book-Ratings.csv" and there-after the union is or connection is defined for examination.

```
user = pd.read_csv('BX-Users.csv', sep=';', error_bad_lines=False,
encoding="latin-1")
user.columns = ['userID', 'Location', 'Age']
rating = pd.read_csv('BX-Book-Ratings.csv', sep=';',
error_bad_lines=False, encoding="latin-1")
rating.columns = ['userID', 'ISBN', 'bookRating']
df = pd.merge(user, rating, on='userID', how='inner')
df.drop(['Location', 'Age'], axis=1, inplace=True)
df.head()
```

	userID	ISBN	bookRating
0	2	0195153448	0
1	7	034542252	0
2	8	0002005018	5
3	8	0060973129	0
4	8	0374157065	0

Figure 2*:* Results Produced by Preprocessing

Step2: Rating Distributions

Item evaluation is the most significant component of this calculation; rating is acquired principal clients where the client expressly gives their appraisal to the item. The end is that the framework gives results to the client by preparing this information, as a scale of the size of 0 to 10, which demonstrates the evaluation generally despised to the client's perspective, this information makes it conceivable to do computation measurement which results new to show item where the two sides rating high by the client.

```
from plotly.offline import init_notebook_mode,plot,iplot
import plitly.graph_objs as go
init_notebook_mode(connected=true)
data=df['bookingRating'].value_counts().sort_index(ascending=False)
trace = go.Bar(x=data.index,
        text=['{.1f} %'. format(val)
        for val in (data.values / df.shape[0] * 100)].
        textposition = 'auto',
        textfont = dict(color='#00000'),
        y=data.values)
        layout=dict(title='Distribution of {} book-ratings'.
        format(df.shape[0]),
        xaxis=dict(title='Rating'),
        yaxis=dict(title='Count'))
flg=go.Figure(data=[trace],layout=layout)
iplot(flg)
```

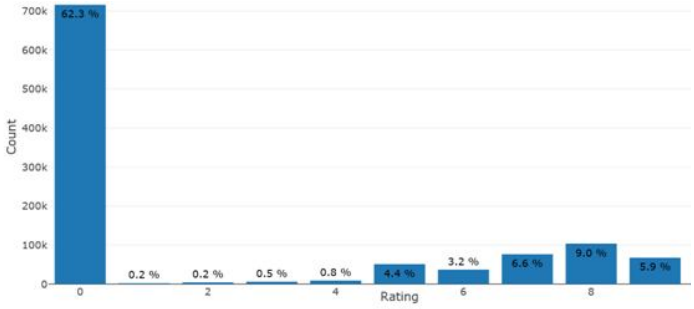

Figure 3: Code Block and Graph Produced using Rating Distribution by Customers

Step 3: KNN and Cosine Similarity Integration

Consider k as the desired number of nearest neighbors and S:=p1,...,pn be the set of training samples in the form p1=(xi,ci), where xi is the dimensional feature vector of the point pi and ci is the class that pi belongs to.

For each p'=(x', c')

- Compute the distance (x′, xi) between p′ and all pi belonging to S
- Sort all points pi according to the key d(x′,xi)
- Selectthefirstk pointsfrom the sorted list, those are thek closesttrainingsamples to p′
- Assign a class to p′ based on majority vote: c′=arg max y∑(xi,ci) belonging to S, I(y=ci)

End For

The steps for the K-NN algorithm are as follows:

1. Determine the parameter k (number of closest neighbors).
2. Calculate the square of the object's Euclidean distance to the given training data.
3. Sorting result number 2 ascending (sequentially from high to low value)
4. Collecting category Y (Classification of nearest neighbors based on value k)
5. By using the category of nearest neighbor which is the majority, it can be predicted the object category.
6. Cosine Similarity $Cosine(Di) = \frac{sum(W_q * W_d)}{sqrt(W_q^2)} * sqrt(W_d^2)$.

```
import distance
import operator

def get_k_neighbours(dict,person,friends,k):
    all_neighbours=[]
    for other in dict:
        if other == person: continue
        #sim =  distance.distance_pearson(dict,person,other)
        sim =  distance.distance_euclid(dict,person,other)
        #sim = distance.cosine_similarity(dict, person, other)
        if sim <= 0: continue
        if friends[other] == 0:
            sim = sim/2

        all_neighbours.append((other,sim))

    all_neighbours.sort(key=operator.itemgetter(1))
    all_neighbours.reverse()
    k_neighbours =all_neighbours[:k]
    return k_neighbours

def getRecommendation(dict,neighbours,person):
    total_sum = {}
    sim_sum = {}
    for user,sim_score in neighbours:
        for item in dict[user]:
            if (item not in dict[person] or dict[person][item] == 0):
                total_sum.setdefault(item,0)
                total_sum[item] += (dict[user][item]*sim_score)
                sim_sum.setdefault(item,0)
                sim_sum[item]+=sim_score
```

Table 1: Top Recommendation with Precision and Recall

S.No	Best Recommendations	Precision	Recall
1	Noli Me Tangere	10	10
2	Walk Two Moons	8	5
3	From the Mixed-Up Files of Mrs. Basil E. Frankweiler	8	5
4	BAG OF BONES : A NOVEL	8	5
5	The Tommyknockers	7	5
6	Midnight Graffiti	7	5
7	Island of the Blue Dolphins	7	5
8	The Gift	6	5
9	On Writing	6	5

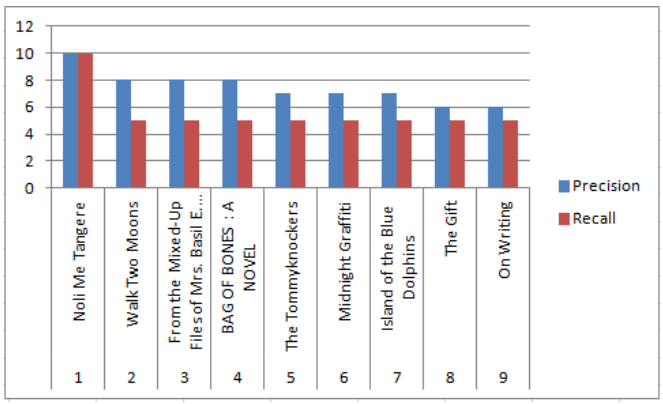

Figure 4: Graph Representation of Top Recommendation with Precision and Recall

Based on above scheme the following results are achieved:-

Table 2: Final Precision 83.00%, Final Recall 87.00% and Accuracy Achieved 81.11%

Overall Precision	Overall Recall	Accuracy Achieved
83.00%	87.00%	81.11%

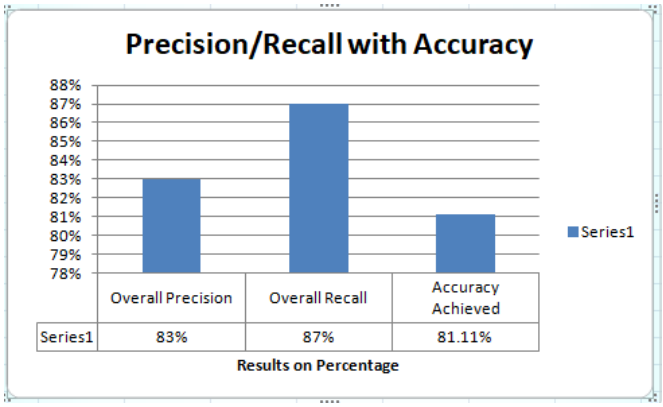

Figure 5: Graph Representation of Table 2 (Results).

5.CONCLUSION AND FUTURE SCOPE

5.1 Conclusions

K-Nearest Neighbor with Euclidean Distance and Cosine Similarity method is applied to datasets therefore, the methods are proven to improve system accuracy and results with 83% of precision, 87% of recall and 81.11% of accuracy. Consequently, Recommender systems have great value in recommending relevant resources to users. It can be quite useful in finding novel and serendipitous recommendations.

5.2 Future Scope

The above scheme should be incorporated with Big-Data and likewise scenarios to evaluate the huge datasets. Even for more accuracy and swift results the scheme should be integrated with ensemble techniques like boosting and bagging

REFERENCES

1. F.O. Isinkaye, Y.O. Folajimi, B.A. Ojokoh, **Recommendation systems: Principles, methods and evaluation**, *Egyptian Informatics Journal, Volume* 16, Issue 3, 2015, Pages 261-273, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2015.06.005.
2. Casey, E. 2014. **Scalable Collaborative Filtering Recommendation Algorithms on Apache Spark**. *Claremont: CLAREMONT McKENNA COLLEGE*
3. Ricci, F., Rokach, L. & Shapira, B. 2015. **Recommender SystemsHandbook (second edition).***New York: Springer Science+Business Media LLC.*
4. D.A. Adeniyi, Z. Wei, Y. Yongquan, **Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method**, *Applied Computing and Informatics*, Volume 12, Issue 1, 2016, Pages 90-108, ISSN 2210-8327, https://doi.org/10.1016/j.aci.2014.10.001.
5. Sheugh, Leily & Alizadeh, Sasan. (2015). **A note on pearson correlation coefficient as a metric of similarity in recommender system**. 1-6. 10.1109/RIOS.2015.7270736.
6. Blake, M. Brian. "**Two Decades of Recommender Systems at Amazon**." (2017).
7. Linden, dkk. 1998. **Collaborative Recommendations Using Item-to-Item Similarity Mappings**
8. Nidhi, R. & Basava, Annappa. (2017). **Twitter-user recommender system using tweets: A content-based approach**. 1-6. 10.1109/ICCIDS.2017.8272631.
9. Valcarce, Daniel & Landin, Alfonso & Parapar, Javier & Barreiro, Alvaro. (2019). **Collaborative filtering embeddings for memory-based recommender systems**. *Engineering Applications of Artificial Intelligence*. 85. 347-356. 10.1016/j.engappai2019.06.2020
10. Ghazanfar, Mustansar ali & Iqbal, Hina & Azam, Muhammad Awais & Aljohani, Naif & Alowibdi, Jalal. (2017). **Building scalable and accurate hybrid kernel mapping recommender systems**. 488-493. 10.1109/IntelliSys.2017.8324338.
11. Glauber, Rafael & Loula, Angelo. (2019). **Collaborative Filtering vs. Content-Based Filtering: differences and similarities**.
12. Kokate, Shrikant. (2018). **Hybrid Content-Based Filtering Recommendation Algorithm on Hadoop**.
13. Kawai, M. & Nogami, S.. (2016). **A hybrid recommender system of collaborative and content based filtering**. 19. 2177-2183.
14. (2020). Machine Learning. 10.1201/9781351006668-4.
15. Jalal, Ahmed. (2019). **A Hybrid Recommendation System**. 10.13140/RG.2.2.31966.18246.
16. Gómez-Uribe, Carlos & Hunt, Neil. (2015). **The Netflix Recommender System. ACM Transactions on Management Information Systems**. 6. 1-19. 10.1145/2843948.
17. Zhou, Hong. (2020). **K-Nearest Neighbors**. 10.1007/978-1-4842-5982-5_7.
18. Boehmke, Brad & Greenwell, Brandon. (2019). **K-Nearest Neighbors**. 10.1201/9780367816377-8.
19. Garcia, Edel. (2015). **Cosine Similarity Tutorial**.
20. Nath, Satyendra & Chakrabartty, S.N.. (2018). **COSINE SIMILARITY APPROACHES TO RELIABILITY OF LIKERT SCALE AND ITEMS.**