Cinematics Recommendation System using Content-based Filtering and Cosine Similarity

Prof. Dhanraj Jadhav[1]
Computer Department
JSPM's Rajarshi Shahu College of Engineering
Pune, India

Aditya Utpat[2]
Computer Department
JSPM's Rajarshi Shahu College of Engineering
Pune, India

Kunal Nilakhe[3]
Computer Department
JSPM's Rajarshi Shahu College of Engineering
Pune, India

Shubham Sonawane[4]
Computer Department
JSPM's Rajarshi Shahu College of Engineering
Pune, India

Sushant Dhore[5]
Computer Department
JSPM's Rajarshi Shahu College of Engineering
Pune, India

***

Abstract— In today's world recommendation systems define our choices online, be it ordering food, buying books, clothes reading an article in the newspaper or be it watching movies online. The number of movies made every year is astonishing, to help a user choose through his liking with wasting time looking for movies we have made this recommendation system. Even though many recommendation models have been proposed, most of them cannot provide the existing user with useful recommendations and do not have a proper plan to recommend movies to new users. In the context of recommendation of movies, this paper is aimed to explain making and implementation of Movie Recommendation Systems Using Machine Learning Algorithms, Sentiment Analysis and Cosine Similarity.

Keywords— Cosine Similarity, Recommendation System, Content-Based Filtering

The Digital Age has brought an explosion of data and digital information which increases the volume the data growth. In fact, 2.5 Exabyte data is created every day. The main use of Recommendation Systems is to help users get a personalized results according to their preferences. Recommendation Systems can also be used as a filtering technique to take out the best result from a set of predicted results by use of some Machine Learning Algorithms, which predict according to the viewer's previous searches and preferences. Movie suggestions for users are dependent on web-based models. As movies can be segregated on basis of genres like thriller, animation, comedy, action, drama etc. Another way to categorize movies are on the basis of some metadata such as cast, year of release, language or director. Nowadays, most of online video-streaming platforms provide a number of similar types of TV shows and movies to the user by the help of utilising users previous search keywords and previous watch history of the user. Taking this history into consideration the machine learning model identifies the movies with similar tastes and provides recommendations. Keeping this in mind we can say that the recommendation systems are a type of information filtering system that aims at predicting the future outcomes, preferences based on pre available data. Recommendation systems are primarily using three approaches which are content based filtering, collaborative filtering and hybrid filtering. In this paper we are proposing a content-based filtering system which recommends movies to users based on their previous preferences. It uses cosine similarity to identify the movies which are similar. These similarity scores are calculated by cosine similarity by taking vector data as its input. Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. In mathematical terms what it does is, it measures the cosine of the angle between two vectors projected in a multi- dimensional spaces.

Content-based filtering

The content based method uses the user-item interaction, but also approaches additional information about users and items. This information can be anything, in the case of a movie recommendation system this additional information can be the age, sex, job and other personal details of the user. The cast of the movie, the category, duration and other deterministic characteristics. The idea of content based methods is to try to build a model, based on the available "features", that explain the observed user-item interactions.

Collaborative filtering

Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items in order to produce new recommendations. the main idea that rules collaborative methods is that these past user-item interactions are sufficient to detect similar users and/or similar items and make predictions based on these estimated proximities.

Hybrid Filtering

Hybrid systems are the one which takes into account both above stated approaches to deal with operational data more concisely.

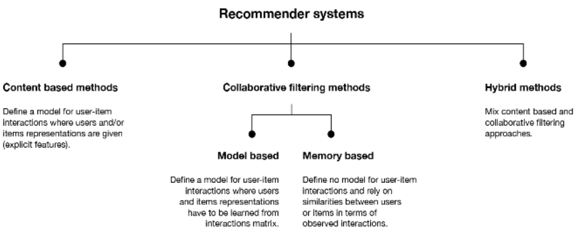

Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.
Hybrid methods
Mix content based and collaborative filtering approaches.

Fig 1. Summary of recommender system methods

I. RELATED WORK

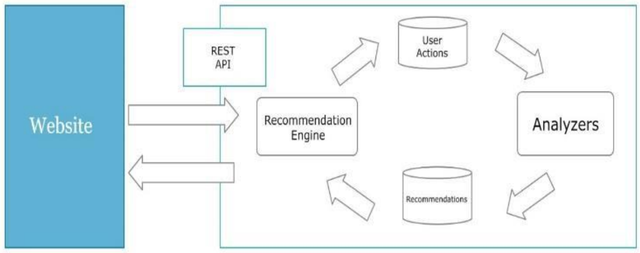
A. Content-Based recommendation system

The content-based recommendation systems treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on an item's features. In this paper we are frequently using words like item and attribute, so what exactly is item and attribute? **Item:** Item would refer to content whose attributes are used in the recommender models. These could be movies, documents, books, etc. In this case it is movies.

Attribute: It refers to the characteristics of an item. A movie tag, words in a document are examples.

In this model we are using Content-Based recommendation system, but how does a Content-Based recommendation system work?

A content based recommender works with the data that user provides, either explicitly (ratings, reviews be it written or scores) or implicitly (by clicking on a link, adding to watch list). 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.



The model works on the similarities derived from the items and then the similarity scores of these items are calculated using cosine similarity. For the cosine similarity to work we need the scores in the form of vectors, as the cosine similarity cannot be calculated using string data which the data type of available items. The concept **of TF-IDF vectorizer** is introduced here to solve this issue. The TF-IDF stands for Term Frequency (TF) and Inverse Document Frequency (IDF). TF is simply the frequency of a word in the document. IDF is used mainly because of two reasons. Suppose we search "Dawn of the planet of the apes" on Google. It is certain that words like "the" and "of" occur frequently than "dawn, planet, apes" but the relative importance of these words is higher than in the search query point of view. In such cases, TF-IDF weighting negates the effect of the high frequency words in determining the importance of a specific word. TF-IDF uses log to dampen effect of high frequency words. It means that it does not measure the relevance of the words in a document or text with a simple row count.

The model works on the similarities derived from the items and then the similarity scores of these items are calculated using cosine similarity. For the cosine similarity to work we need the scores in the form of vectors, as the cosine similarity cannot be calculated using string data which the data type of available items. The concept **of TF-IDF vectorizer** is introduced here to solve this issue. The TF-IDF stands for Term Frequency (TF) and Inverse Document Frequency (IDF). TF is simply the frequency of a word in the document. IDF is used mainly because of two reasons. Suppose we search "Dawn of the planet of the apes" on Google. It is certain that words like "the" and "of" occur frequently than "dawn, planet, apes" but the relative importance of these words is higher than in the search query point of view. In such cases, TF-IDF weighting negates the effect of the high frequency words in determining the importance of a specific word. TF-IDF uses log to dampen effect of high frequency words. It means that it does not measure the relevance of the words in a document or text with a simple row count.

After calculating TF-IDF scores, how do we determine which items are closer to each other, or closer to the user's profile, which is user liking, preference? This is accomplished using the Cosine Similarity.

The similarity between item vectors can be computed by three methods: -

1. Cosine Similarity
2. Euclidian Distance
3. Pearson's Correlation

In the proposed system we are using Cosine Similarity. It is predominantly used approach to match similar documents. It is a metric used to determine how similar the documents are irrespective of their size. Mathematically, it measures the cosine angle between the two vectors projected in a multi- dimensional space. In this context, the two vectors we are talking about are arrays which contain the word counts of the two documents, which are the textual reviews taken from the users. When plotted on a multi-dimensional space, where each dimension corresponds to a word in the document, the cosine similarity captures the orientation (the angle) of the documents and not the magnitude. If magnitude is intended, then we can compute Euclidean distance instead.

The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance because of the size (like word 'movie' appeared 60 times in one document and 20 times in another) they could still have a smaller angle between them.

B. *Sentiment analysis using Multinomial Naïve Bayes*

Most sentiment analysis systems use bag-of-words approach for mining sentiments from the online reviews and social media data. Rather considering the whole sentence/ paragraph for analysis, the bag-of-words approach considers only individual words and their count as the feature vectors. This may mislead the classification algorithm especially when used for problems like sentiment classification. Traditional machine learning algorithms like Naive Bayes, Maximum Entropy, SVM etc. are widely used to solve the classification problems.

In this system we are using Multinomial Naïve Bayes, it tends to be **a baseline solution** for sentiment analysis task. The basic idea of Naive Bayes technique is to find the probabilities of classes assigned to texts by using the joint probabilities of words and classes. According to Bayes theorem:

Posterior = Likelihood * proposition/evidence

Or

$P(A|B) = P(B/A) * P(A)/P(B)$

Multinomial Naïve Bayes works exceptionally well with text classification and spam filtering. The advantage of working with Multinomial Naïve Bayes are that it requires a small amount of training data to learn the parameters and it can also be trained relatively faster compared to sophisticated models.

II. PROPOSED METHODOLOGY AND RESULTS

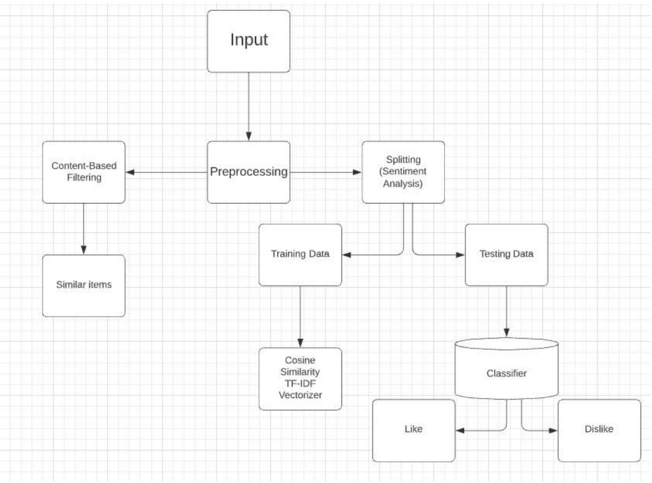


Fig 2. Model of proposed system

A. *Pre-processing*

The input is taken from the user and by using content-based filtering all the items which are similar to the item given as an input by the user are suggested. The sentiment analysis is also done after this step.

B. *Splitting*

The whole dataset is split into training and testing database. The 80% data is taken for training while the remaining 20% data is used for testing.

C. *Classification*

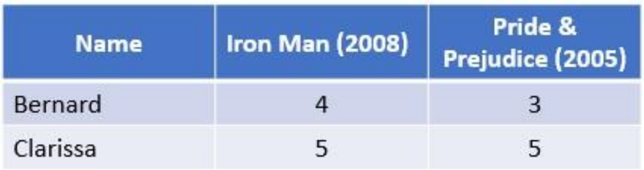
The training data for sentiment analysis is trained by using the cosine similarity algorithm, TF-IDF vectorizer, Multinomial Naïve Bayes.

1. Cosine Similarity

That the results from computing item A to item B or item B to item A is the same. We can therefore compute the score for each pair of nodes once. We don't compute the similarity of items to themselves.

To understand how a cosine similarity works in a recommender we will use an example which helps us understand how similarity scores from reviews works and how it is perceived accurately if the two items are similar/ near in the distance, even if the geometric distance between them is more.

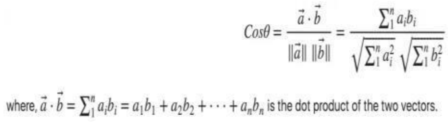
Suppose we want to check if Bernard and Clarissa have similar movie preferences and, we only have two movie reviews. The reviews are scores from 1 to 5, where 5 is the best and 1 is the worst score, and 0 means the person has not yet watched the movie.



Name	Iron Man (2008)	Pride & Prejudice (2005)
Bernard	4	3
Clarissa	5	5

Fig 3. Scores of Bernard and Clarissa

In the above image we have the scores given by Bernard and Clarissa to the 2 movies. Now the cosine similarity will measure the similarity between these two vectors which is a measurement of how similar are the preferences between these two people.

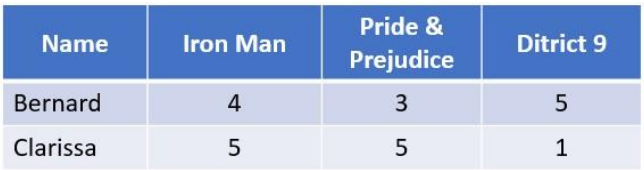


$$Cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\,\|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a}\cdot\vec{b} = \sum_1^n a_i b_i = a_1b_1 + a_2b_2 + \cdots + a_nb_n$ is the dot product of the two vectors.

Fig 4. Cosine Similarity Formula

Each vector represents a person's preferences and they have an angle theta between them. Similar vectors will have a lower angle theta and dissimilar vectors (different film preferences in this case) will have bigger theta value.

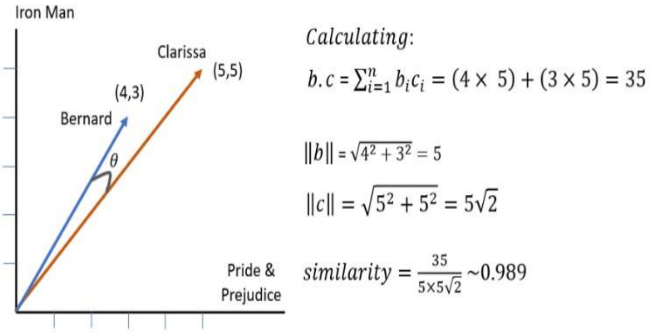


Name	Iron Man	Pride & Prejudice	Ditrict 9
Bernard	4	3	5
Clarissa	5	5	1

Fig 5. Calculating the similarity scores

In the example above the similarity 0.989 is close to the maximum value of 1, this means that given only two movie reviews the two users have similar preferences.

The cosine similarity can be calculated for more than 2 movies. In the example below we will add the ratings for a movie that Bernard liked and Clarissa disliked this should decrease cosine similarity value Cosine similarity algorithm is a symmetrical algorithm, which means



Calculating:

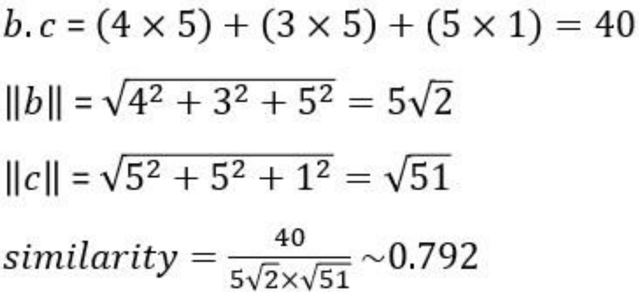
$$b.c = \sum_{i=1}^{n} b_i c_i = (4\times 5) + (3\times 5) = 35$$

$$\|b\| = \sqrt{4^2+3^2} = 5$$

$$\|c\| = \sqrt{5^2+5^2} = 5\sqrt{2}$$

$$similarity = \frac{35}{5\times5\sqrt{2}} \sim 0.989$$

Fig 6. Scores of Bernard and Clarissa



$$b.c = (4\times 5) + (3\times 5) + (5\times 1) = 40$$

$$\|b\| = \sqrt{4^2+3^2+5^2} = 5\sqrt{2}$$

$$\|c\| = \sqrt{5^2+5^2+1^2} = \sqrt{51}$$

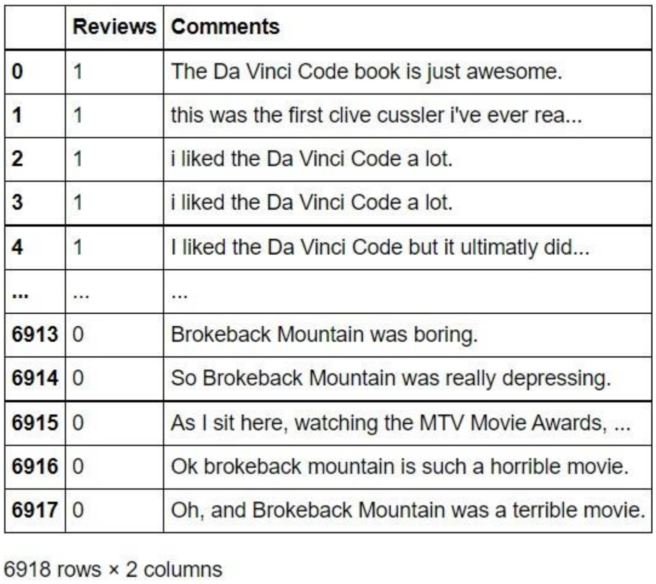
$$similarity = \frac{40}{5\sqrt{2}\times\sqrt{51}} \sim 0.792$$

Fig 7. Calculating the similarity scores

The similarity has reduced from 0.989 to 0.792 due to the difference in ratings of the District 9 movie. The cosine can also be calculated in Python using the Sklearn library.

2. TF-IDF Vectorizer and Multinomial NaïveBayes theorem.

The TF-IDF vectorizer and Multinomial Naïve Bayes are a very essential part of the model as all the user given reviews available in the dataset are textual, i.e. in string format but the data required for cosine similarity should be in vector format as it is unable to perform dot product on any other type of data. In a text the frequency of the words, the significance of the word with respect to the text and also the item all these details are classified using TF-IDF vectorizer and Multinomial Naïve Bayes theorem, to supplement it we are also using stop words in python which also spots insignificant words like an, a which do not weigh in on the inclination of the text be it positive or negative review. Such words are filtered out and data is classified properly. A special dataset which contains all the reviews is used. These incoming sentences from the review dataset are first split up into several words via a process called "Tokenization". Then it is much easier to look at the sentiment value of each word sentence via comparing within the sentiment lexicon. Actually there is no machine learning going on here but this library parses for every tokenized word, compares with its lexicon and returns the polarity scores. This brings up an overall sentiment score for the tweet.



	Reviews	Comments
0	1	The Da Vinci Code book is just awesome.
1	1	this was the first clive cussler i've ever rea...
2	1	i liked the Da Vinci Code a lot.
3	1	i liked the Da Vinci Code a lot.
4	1	I liked the Da Vinci Code but it ultimatly did...
...	...	...
6913	0	Brokeback Mountain was boring.
6914	0	So Brokeback Mountain was really depressing.
6915	0	As I sit here, watching the MTV Movie Awards, ...
6916	0	Ok brokeback mountain is such a horrible movie.
6917	0	Oh, and Brokeback Mountain was a terrible movie.

6918 rows × 2 columns

Fig 8. Ratings and Reviews from dataset

Above image shows how the reviews are scored after analysis on the textual data and to project the accuracy of the proposed system the same reviews can be seen which are scored exactly like its other copies.

III. CONCULSION

Recommendation Systems are a great tool to filter out information and provide only the relevant information to the user. In this paper we have illustrated a machine learning model named Cinematics Recommendation System by making use of Content-Based filtering, which recommends movies similar to the movie user likes and analyses the sentiments on the reviews given by the user for that movie. The main use of Sentiment Analysis in the proposed model was to observe reviews of users for a particular movie. Weighted cosine similarity score was also used to increase accuracy of model. It is found that this proposed model produced much better results than the other basic models. As a vast variety of movies with all kinds of attributes were taken into consideration while training the model, users can be satisfied with the recommended results.

REFERENCES

1. G. Wang, "Survey of personalized recommendation system," Computer Engineering & Applications, 2012.

2. Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state- of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6):734–749, 2005. Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. Modern information retrieval, volume 463. ACM Press New York, 1999

3. ShumeetBaluja, Rohan Seth, D Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In Proceedings of the 17th international conference on World Wide Web, pages 895– 904. ACM, 2008.

4. Xu Hailing, Wu Xiao, Li Xiaodong, and Yan Baoping. Comparison study of internet recommendation system. Journal of Software, 20(2):350–362, 2009.

5. T. E. D. Mining, "Enhancing teaching and learning through educational data mining and learning analytics: An issue brief," in Proceedings of a conference on advanced technology for education, 2012.

6. Nakagawa and T. Ito, "An implementation of a knowledge recommendation system based on similarity among users‟ profiles," in Sice 2002.

roceedings of the Sice Conference, 2002, pp. 326–327 vol.1.

7. Kelvin Luk, Introduction to TWO approaches of content-based recommendation system

8. Graham L. Giller (2012). ”The Statistical Properties of Random Bit-streams and the Sampling Distribution of Cosine Similarity”. Giller Investments Research Notes (20121024/1).

9. Minara P Anto, Mejo Antony, KM Muhsina, Nivya Johny, Vinay James, and Aswathy Wilson. Product rating using sentiment analysis. In International Con-ference on Electrical, Electronics, and Optimization Techniques, pages 3458– 3462. IEEE, 2016.

10. Erik Cambria. Affective computing and sentiment analysis. IEEE Intelligent Systems, 31(2):102–107, 2016. .Ivan´ Cantador, Alejandro Bellog´ın, and David Vallet. Content- based recommen-dation in social tagging systems. In Proceedings of the Fourth Conference on Recommender systems, pages 237–240. ACM, 2010.

11. Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth International Conference on Weblogs and Social Media, 2014.

12. Hui Li, Jiangtao Cui, Bingqing Shen, and Jianfeng Ma. An intelligent movie recommendation system through group-level sentiment analysis in microblogs. Neurocomputing, 210:164–173, 2016.H. Poor, An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, 1985, ch. 4.

13. Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based rec- ommender systems: State of the art and trends. In Recommender Systems Hand-book, pages 73–105. Springer, 2011.

14. Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algo-rithms and applications: A survey. Ain Shams Engineering Journal, 5(4):1093–1113, 2014.

15. Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. Aaai/iaai, 23:187–192,

2002

16. https://www.ijeat.org/wp-content/uploads/papers/v9i5/E9666069520.p df

17. https://medium.com/@bkexcel2014/building

-movie-recommender-systems-using-cosine-similarity-in-python-eff2d4e60d24

18. https://ieeexplore.ieee.org/document/63829

10

19. https://dl.acm.org/doi/10.1145/3411174.341

1194

20. https://www.seas.upenn.edu/~cse400/CSE40

0_2006_2007/ChenJatia/Writeup.pdf

21. https://hcis-journal.springeropen.com/articles/10.1186/s1

3673-018-0161-6

22. https://github.com/kishan0725/AJAX-Movie-Recommendation-System-with-Sentiment-Analysis#architecture

23. http://www.ijmtst.com/volume7/issue01/4.IJMTST0612293.pdf

24. https://www.machinelearningplus.com/nlp/cosine-similarity/

25. https://towardsdatascience.com/introduction

-to-recommender-systems-6c66cf15ada

26. https://ieeexplore.ieee.org/abstract/document/7219856

27. https://ieeexplore.ieee.org/abstract/document/7050801

28. https://towardsdatascience.com/real-time-sentiment-analysis-on-social-media-with- open-source-tools-f864ca239afe

29. https://towardsdatascience.com/real-time-sentiment-analysis-on-social-media-with- open-source-tools-f864ca239afe

30. https://web.stanford.edu/~jurafsky/slp3/4.pd f