JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Exploring movie recommendations using content analysis

Prudhvi Shankar A[1], Suhasini TS[2], Komal Ramapragada[3], Pruthvi Teja[4], and Aparna Laxmi[5]

CMR Engineering College, Medchal, Telangana, 501401, INDIA

Abstract. Recommender System is a tool helping users find content and overcome information overload. It predicts interests of users and makes recommendation according to the interest of users.

The original content-based movie recommender system is the continuation and development of collaborative filtering, which doesn't need the user's evaluation for items. Instead, the similarity is calculated based on the information of items that are chose by users, and then make the recommendation accordingly [1]. Using natural language processing and vectorization techniques, it is possible to extract features from movie attributes like overview, genre, keywords, cast and crew to find movies with similar features. In this thesis, a feature extraction method is presented and the use of the extracted features in finding similar movies is investigated. We do the text pre-processing on a collection of movie attributes. We then extract tags from the collection using simple preprocessing techniques and store those tags to vectorize and then take out the similarity for each movie and recommend top-n items to the user.

Keywords: Recommender Systems, Content-based filtering, Similarity.

1 Introduction

Recommendation systems are a type of software application or algorithm that provides suggestions or recommendations to users. These recommendations are typically personalized and are based on the user's preferences, behavior, or characteristics. The primary goal of recommendation systems is to assist users in discovering items or content they may find interesting or relevant.

These systems predict the most likely product that the users are most likely

to purchase and are of interest to. The goal of the recommender system is to predict the most relevant item to the user. Some of the most popular examples of recommender systems include the ones used by Amazon, Netflix and Facebook.

Recommender systems are highly useful as they help users discover products and services they like, otherwise they might not have found them on their own. It helps users discover new items that they are likely to be interested in, and to improve engagement and user satisfaction with the system. These systems can provide personalized recommendations to individual users based on their past behavior, preferences and demographic information.

1.1 Personalized recommender system

The personalized recommender system analyzes users data, their purchase, interests, rating and their relationships with other users in more detail. In that way, every user will get customized recommendations. Content-based and col-laborative filtering comes under personalized recommender systems.
Examples : Content-based filtering, collaborative-based filtering.

1.2 Non-Personalized recommender system

As the name suggests, this recommender system provides general recommendations to the user without any context of what the user wants or what his preference is. For example, when a website is visited by the user, it provides a list of most popular items. These popular items can be based on different parameters like geography, age, gender etc.
Examples : Popularity-based filtering.

1.3 Relationship between Content based and collaborative based filtering

Let’s first talk about collaborative filtering. Collaborative filtering is an advanced filtering technique to generate recommendations based on mutual user interests. In this concept, the interests of users are based on their preferences and behaviors of other similar users. In order to recommend items to a user with collaborative filtering approach, the user needs to provide data like their favourite genres, actors and such attributes explicitly. If the user denies to provide the data, then collaborative filtering approach fails and goes into a condition calledCold Start.

In this scenario, to keep the model working even with no user explicit data, other filtering methods like content based and popularity based filtering are used where the model works well without data from user. In combination of these filtering methods, a robust hybrid recommender model can be built which can generate recommendations in any case.

2 Machine learning concepts for Content analysis

2.1 Content based Filtering

Content-based filtering is a recommendation system technique that suggests items to users based on the attributes or features of the items themselves. Un-like collaborative filtering, which relies on user-item interactions, content-based filtering focuses on the characteristics of the items and matches them with the user's preferences. This approach is particularly useful in scenarios where ex-plicit user feedback data is scarce or unreliable.

2.2 Vectorization overview

Vectorization is a powerful concept in computer science. This process involves in process of converting raw data into numbers that a machine learning (ML) model can support.
Some common vectorization techniques include:

a) Bag of words
b) Term frequency-inverse document frequency
c) Word2vec (or) Doc2vec

2.3 Finding Similarity

In machine learning, there are several methods to find similarities between data points, which are commonly used in tasks such as recommendation systems, clustering, and information retrieval. Here are some of the most commonly used methods:

i) Cosine Similarity : Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space. It is computed as the dot product of the two vectors divided by the product of their magnitudes. Cosine similarity is widely used in text similarity tasks, recommendation systems, and information retrieval, especially when dealing with high-dimensional sparse data like text documents.
And other similarity finding methods are,

ii) Euclidean Distance
iii) Pearson Correlation Coefficient
iv) Jaccard Similarity
v) Manhattan Distance (City Block Distance)

For this project, we particularly have chosen cosine similarity as our similarity finding method because it is precise when we are working with huge data.

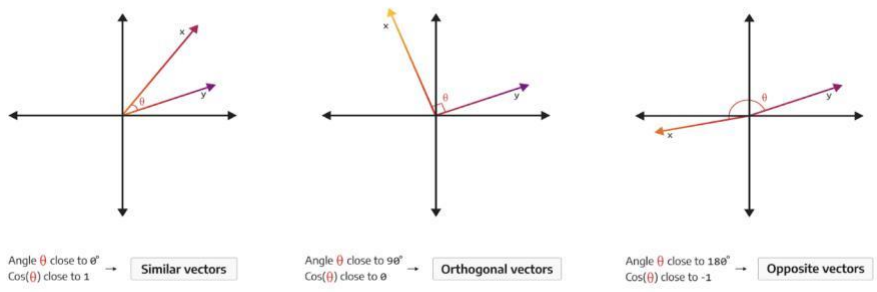

Fig. 1. This figure illustrates the similarity of the vectors

3 Related works and publications

A Recommendation System provides items and suggestions to a person based on his or her interests and past usage history. Such a system is the backbone to many of today's content streaming services such as Netflix, Pandora and Youtube. Recommendation Systems (RS) are usually classified based on the approach used to filter information such as content based filtering, collaborative filtering (based on users activities) and hybrid (combining both). Collaborative filtering based RS have seen interest lately because of the Netflix competition

10. whereas content based systems face challenges of efficient feature representation of meta-data from audio, video and text. Luckily for the movie domain, lot of textual information is readily available such as plot lines, dialogues and reviews.

Movie data in the form of keywords, script, dialogue, and overview has been used in research activity in the past decade [2]–[5]. [3] explores movie recommendations using cultural metadata such as user comments, plot outlines, keywords, etc, and shows highest precision with user comments. The report [4] discusses movie classification using NLP based methods such as Named Entity Recognizer (NER) and Part-of-Speech (POS) tagger with movie script as input. It concludes that NLP based features perform well when compared to non-NLP features (without the use of NER and POS) although it reports only 50% accuracy because of the small corpus size.

In order to use movie review as data, it is necessary to remove irrelevant words, symbols, html tags, etc,. In NLP, a large number of open source tools and libraries [6]–[7] are available and used as the first step in any kind of text process-ing. The chapter 7 of [6] mentions the steps involved in extracting information from text. [9] uses nltk toolkit for stopwords and stemming steps. Text data with noise can drastically affect the result of any kind of NLP based model training. Text filtering helps in removing unnecessary information and allows us to use

complex mathematical models on it.

The paper [11] compares the results obtained by preprocessing of meta data. It simply computes the cosine similarity from the document-term matrix of movie data. Although the paper showed highest precision with user comments of movies, it did not analyze the data further by advanced techniques such as LSA. After preprocessing, reducing the data dimensionality is the next step in feature extraction.

Document-term matrix is used as the input to many semantic analysis tech-niques starting from basic tf-idf schemes to complex models such as LSI, LSA and LDA. These dimensionality reduction techniques could yield semantic fea-tures of big data from off-the-shelf hardware [12]. Such models are interesting candidates to investigate semantic concepts from movie data. The thesis [13] studies sentiment analysis done on movie review using LSA but concludes that dimensions that capture the most variance are not always the most discriminating features for classification.

On the other hand, [14] shows interesting results when using topic modeling in content based RS. Probabilistic topic modeling allows us to extract hidden features i.e. “topics” from documents. LDA, a model based on topic modeling, shows good results in both document clustering [15] and recommendation sys-tem. It can capture important intra-document statistical structure by considering mixture models for exchangeability of both words and documents within a cor-pus. With probabilistic techniques such as LDA, it is possible to derive semantic similarities from textual movie data. Such extracted semantic information can be used to find similar movies. Moreover, LDA can assign topic distribution to new unseen documents, an important requirement for building scalable RS for movies as it should be trivial to add new movies on a regular basis. For a RS, computing similarity is an essential part, be it either similarity of content or user rating.

4 Proposed Approach

4.1 Summarizing the system

1) Two datasets were taken from https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata and preprocessed for the project.

– tmdb 5000 movies.csv, is a dataset of 5000 movies from different genres with attributes like overview, keywords, budget, titles etc.

– tmdb 5000 credits.csv, is a dataset of all the names of cast and crew of extracted movies.

2) Both datasets are merged based on movie titles so that we have collective information about cast and crew in single corpus.

3) A machine learning based model is built on the corpus to generate movie recommendations.

4) Using similarity metrics, a list of five similar movies for each selected movie is created.

4.2 Corpus attributes

In the dataset, there are movie attributes like movie id, title, genre, keywords, overview, cast, crew etc. We decided to take only relevant attributes to build the model. There are also other attributes like budget, revenue, runtime, production companies etc. These are not taken into consideration because a recommenda-tion cannot be generated based on these attributes. For instance, let us assume budget has been considered. If a high budget movie is liked by a user, it is not necessary that he/she will like all other high budget movies. So, there would be no meaning in taking these type of attributes while building recommender model.

4.3 System Architecture

The system architecture of the executed movie recommendation project is struc-tured to encompass several interconnected components and modules. In the ex-ecuted project, data ingestion and collection are realized through the loading of movie-related data from CSV files (tmdb5000 movies.csv & tmdb 5000 credits.csv). This step involves reading and parsing the raw data from these files into Pandas DataFrames (movies and credits), serving as the initial source of movie metadata and credits information.

The preprocessing and feature extraction steps are evident throughout the code, where various operations are performed to clean, transform, and extract relevant features from the raw data. The primary output of the recommendation engine is a list of recommended movies for each user. These recommendations are generated based on the similarities between movies, computed using cosine similarity between their respective tags. The recommendations are personalized to each user’s preferences and viewing history, aiming to provide relevant and engaging movie suggestions.

4.4 Data Preprocessing

The data in the corpus is in raw format (unwanted spaces, null values etc.) Also, to be able to vectorize the required data, it must be cleaned by removing null values, duplicates and spaces. Also the data type should be compatible while

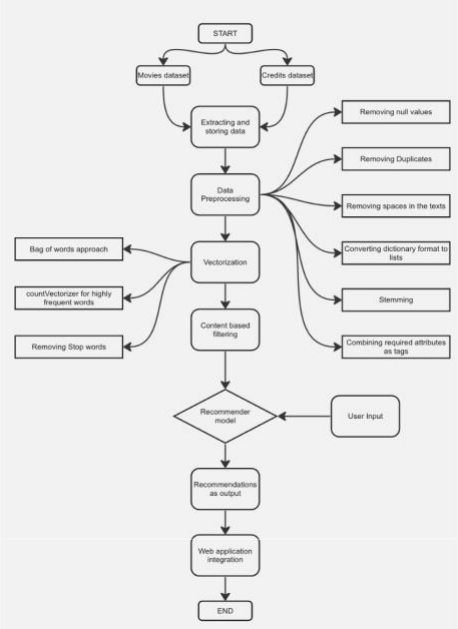

Fig. 2. System Architecture

working with vectors. The data in the corpus are strings in key:value pair (dic-tionary) format, however we need them in list type. For this conversion, we used AST [16] library from python. This converts the strings into proper key value pairs, then literal eval function from this module is used to convert those pairs into lists.
This process is performed on all columns with above textual characteristics. Now from cast attribute, we have selected the three main leads and from crew we have chose to consider only director. Now the data is ready to apply trans-formations over these columns.

For text processing, we used NLTK [17], an open source library. The spaces between the words like Science Fiction has been removed to make it a single word. Overview is basically a small description which describes the heart of the movie plot. It is basically a paragraph on which python's split() fucntion is applied to make them into separate words. Stop words have been removed. Stopwords are high-frequency grammatical words which are usually ignored as they do not provide any useful information. Examples of stopwords are other, there, the, of, are. NLTK's default stopword for english language is used.
8 Prudhvi Shankar et al.

4.5 Vectorization

Now, a new attribute is created which is the concatenation of other attributes like overview, keywords, genre, cast and crew. This new attribute is a collection of many words on which now we apply vectorization to calculate similarity and generate recommendations. Let's address this attribute as "tags".

On these tags, we apply stemming using NLTK's PorterStemmer. Stemming is the process of producing morphological variants of a root/base word. Stem-ming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words "chocolates", "chocolatey", and "choco" to the root word, "chocolate". More advanced and dynamic stemmers like lan-caster stemmer can be used but the drawback is that it is more aggressive and it can be confusing while working on small words.

We chose to utilize bag of words approach for vectorization. If we assume that each tag of these 5000 tags contain 100 words, that's 5,00,000 words which we totally get from this attribute. We have taken the top 5000 words whose fre-quency is high (repeating in most of the tags). One can choose more words with high frequency, but must remember that it becomes more complex.

The process of fetching top 5000 most frequent words like done by utilizing Sklearn library's countvectorizer. Let's address these 5000 words as "feature names". Now using these feature names, we check every tag for word repetitions, hence depicting the similarity between each tag, in overall, each movie.

4.6 Calculating Similarity

After preprocessing and vectorization, we are remained with tags and feature names. We now compare each feature name with each tag to find word repetitions and fill up the graph. We have usedCosine Similarity from Sklearn library to find the similarity between these vectors. Cosine similarity is the angle between two vectors, the less it is, the more similar the vectors are. Using the index (movie id), the titles of the movies are fetched from main corpus.

5 Evaluation and Results

The goal of our experiment is to evaluate the performance of the system and effectiveness in recommending appropriate movies to the selected film. The us-ability of the recommender system is increased when the recommendations are precise.

5.1 Evaluation Criteria

Genre, Plot, Keywords, Overview, Cast and Crew describe how similar the movies are. We observed that the presence of similar genre, crew and cast resulted in more similar movies. Hence the overlap of genres and actors could be useful.

5.2 Web Based movie recommender setup

A web based movie recommender is created to evaluate the recommendations from experiment. This was built using Streamlit, an open source python library.

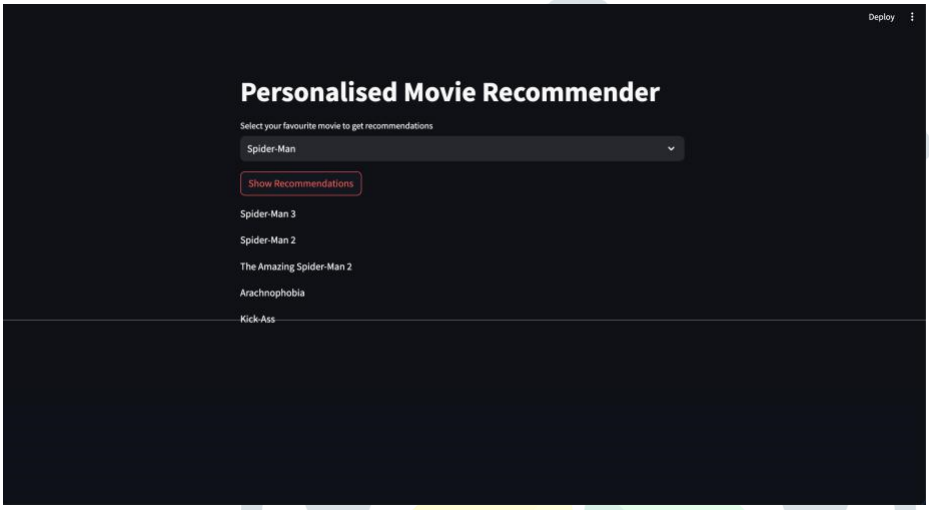

Fig. 3. Homepage of the system

6 Conclusion and further enhancements

6.1 Conclusion

Recommender systems are a powerful new technology for extracting additional value for a business from its customer databases. These systems help customers find products they want to buy from a business. Recommender systems benefit customers by enabling them to find products they desire. Conversely, they help the businesses by generating more sales. Recommender systems are rapidly be-coming a crucial tool in E-commerce on the web.

In this project, we recommended movies for a user using a content based filtering using cosine similarity and bag of words approach. We also compared different models built

using methods and identified the best model and justified that content based filtering is recommended for building recommender systems when the user interaction with the model is less Which means the user history or data is not necessary to provide recommendations. We have used datasets from kaggle. Movies and credits datasets are used for this particular project which helped to leverage the whole system into an efficiently performing model.

In the context of this project, the primary objective has been to enhance the accuracy of the recommendation model which uses content based filtering to pre-dict movies based on item feature similarity. The dataset which we have taken underwent many transformations, preprocessing and vectorization before excel-lently predicting recommendations. Finally, the deployment of this system onto cloud platforms such as Heroku, Vercel or streamlit to be undertaken.

6.2 Future Enhancements

As the future endeavors of the project and the system, it has adequate scope of modification; if it is necessary. Development and launching of a mobile app, refining existing services and adding more services. Also, system security, data security and reliability are the key features which can further be included in future work.

In addition to this, the API for shopping and payment gateway can be added so that the user can be redirected to a website where he/ she can rent or buy that particular movie. Furthermore, in the existing system, we have used only content based filtering whereas, a hybrid approach in which collaborative filtering can be used to make the model much better.

Adding to this, we can add admin side with some functionalities like movies catalog management, user management and server management. Using Collaborative filtering alone cannot overcome unsolvable problems like cold-start, so we need to integrate both content and collaborative filtering techniques to make powerful hybrid models.

References

1. Survir Bhargav. Efficient features for movie recommendations. 2014.

2. B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? sentiment classification using machine learning techniques”, in Proceedings of EMNLP, 2002, pp. 79– 86.

3. S. Ah and C.-K. Shi, “Exploring movie recommendation systems using cultural metadata”, in 2008 International Conference on Cyberworlds, Sep. 2008, pp. 431–438. doi: 10.1109/CW.2008.13.

4. A. Blackstock and M. Spitz, “Classifying movie scripts by genre with a MEMM using NLP-Based features”, Stanford, M.Sc.Course Natural Language Pro-cessing, Student report, Jun. 2008. [Online]. Available: http://nlp.stanford. edu/courses/cs224n/2008/reports/06.pdf.

5. R. Berendsen, “Movie reviews: do words add up to a sentiment?”, PhD thesis, Rijksuniversiteit Groningen, Sep. 2010.

6. S. Bird, E. Klein, and E. Loper, Natural language processing with Python, 1st ed. O’Reilly, 2009.
˘

7. R. Reh˚u˘rek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora”, English, in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, http://is.muni.cz/publication/ 884893/en, Valletta, Malta: ELRA, May 22, 2010, pp. 45–50.

8. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: machine learning in Python”, Journal of Machine Learning Research, vol. 12, pp. 2825– 2830, 2011.

9. J. Vig, S. Sen, and J. Riedl, “Tagsplanations”, Proceedings of the 13th interna-tional conference on Intelligent user interfaces - IUI ’09, 2008. doi: 10 . 1145 / 1502650 . 1502661. [Online]. Available: http : / / dx . doi . org / 10.1145/1502650.150

10. X. Amatriain, The netflix tech blog: netflix recommendations: beyond the 5 stars (part 1). [Online]. Available: http://techblog.netflix.com/2012/ 04/netflix-recommendations-beyond-5-stars.html (visited on Jun. 11, 2014).

11. S. Ah and C.-K. Shi, “Exploring movie recommendation system using cultural metadata”, in 2008 International Conference on Cyberworlds, Sep. 2008, pp. 431–438. doi: 10.1109/CW.2008.13.
˘

12. R. Reh˚u˘rek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora”, English, in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, http://is.muni.cz/publication/ 884893/en, Valletta, Malta: ELRA, May 22, 2010, pp. 45–50.

13. R. Berendsen, “Movie reviews: do words add up to a sentiment?”, PhD thesis, Rijksuniversiteit Groningen, Sep. 2010.

14. T. Luostarinen and o. Kohonen, “Using topic models in content-based news recommender systems”, English, in nodalida13, ser. 19, vol. 85, Oslo, Norway: Linkoping University Electronic Press; 581 83 Linkoping; Sweden, May 2013, 239 of 474, isbn: 978-91-7519-589-6. [Online]. Available: http://emmtee.net/

oe/nodalida13/conference/11.pdf.

15. R. K. V and K. Raghuveer, “Article: legal documents clustering using latent dirichlet allocation”, International Journal of Applied Information Systems, vol. 2, no. 6, pp. 27–33, May 2012, Published by Foundation of Computer Science, New York, USA.

16. https://docs.python.org/3/library/ast.html

17. https://www.nltk.org/index.html

18. https://streamlit.io/

19. Van der Geer, J., Hanraads, J. A. J., & Lupton, R. A. (2000). The art of writing a scientific article. Journal of Science Communication, 163, 51–59.

20. Strunk, W., Jr., & White, E. B. (1979). The elements of style (3rd ed.). New York: MacMillan.