**Page & Reel: The Cross Book/Movie Recommender**

Brian Merrit1, Haritha Ramesh2, Irtaza Haider3,   
Nirav N. Shah4, Ryan Place5, Shubhashree Baskar6

1*Department of Bioinformatics, School of Biology*

2*School of Electrical and Computer Engineering*

3*College of Computing*

Georgia Institute of Technology: CSE6242: Data and Visual Analytics

**Group 17: Final Report**

1. Motivation
2. Problem Definition
3. Literature Review
4. Proposed Method
   1. Intuition
   2. User Interface
   3. API and operations
   4. Recommendation Algorithm
5. Conclusion and Discussion
   1. Conclusion
   2. Future Work
6. Distribution of Effort

Reference: http://poloclub.gatech.edu/cse6242/2018spring/project.html#final-report

# Motivation

Netflix, Amazon, IMDB, GoodReads are few among many of the examples that use advanced machine learning algorithms to recommend a choice of a movie, a book, or a product based on your history of choice, critique reviews and other significant features. However, Netflix and IMDB would only recommend movies and series, GoodReads would only recommend books and so on and so forth. There are no current applications in the market that would perform cross recommendation between movies and books.

Movies and Books often seem to have similar themes, and this often occurs because a lot of movies have been inspired by books. This will help the user (1) to watch movies based on their choice of books and (2) to read books which would be similar to their choice of genres.

We believe that this would not only interest users, but also parents who would like to inculcate the habit of reading in the younger generations. In the recent advancements, the significance of reading books has been known to decrease and thus its value. This project can thus serve to be one of the factors to improve the social culture of the community as well.

# Problem Definition

Currently no cross book-movie recommender exists within the market. Page-Reel will be a extremely user-friendly socializing network to predict the interest of the users for their books & movies based on their recommendations. Users will have an extremely simple friendly interface where they could choose up to 5 books and movies at the start and can obtain recommendations based on the options they select. Also, unlike many other website, we expect our recommendations to be prompt.

# Literature Review

A variety of techniques have been proposed in literature, from content-based to hybrid approaches that prove useful to our project [1]. Few papers present a technique - Modified EM to learn a new user profile using information from existing user profiles [2, 3]. This induces information for new users if time permits which uses a Bayesian network-based approach to using context to make movie recommendations.

However, it works primarily for sparse data. The paper by Sriram et al. present a to classify short texts in a set of generic classes. This approach will not work in our case as it takes context into account, which is not available. Naive Bayes assumes statistical independence between attributes and can apply density estimation on numeric or numerically derived attributes [4]. Bias can unintentionally shape the data and present the user with choices that aren’t what they want [2, 5-7].

Amazon uses item-item collaborative filtering wherein items that are bought together have higher similarity scores [8]. This model scales well when the user base is very large. However, we cannot leverage this as we have datasets from entirely different sources (Goodreads and IMDB) rated by different user bases, in which the correlation between user interests cannot be exploited. For textual similarity, cosine similarity which is a text-similarity best approach and can be used from cross book-movie recommendations. Hybrid recommendations need to be scored to be able to efficiently use their output. Ineffective scaling and scoring can result in poor recommendations and thus using hybrid algorithms wasn’t within the scope of this project [9]. The Naïve based “Bag of Words” approach can instead be used for the same media datasets as the literature proves that they would be more accurate then [10, 11].

# Proposed Method

## 

## Intuition

There are quite a few applications available that use a mix of various algorithms to predict and suggest recommendations based on quantitative features such as ratings, critiques, plots, history, genres and various other features [12]. However, there are none available online for free (as per our knowledge), that performs a cross-recommendation. Since the two happen to have more than just simple common features, we can leverage this by using similarity search algorithms as Naïve Based “Bag of Words” and Cosine Similarity [13].

We would work under the impression that the user prefers books and movies of similar genres (apart from filtering based on ratings and text-similarity searches) [10, 14]. This is something that other recommendation systems haven’t implemented so far. We believe our simple and straightforward approach could be useful, efficient and quick.

List of Innovations

* Creation of a web-based recommendation system to recommend both books and movies and not just one media type.
* Use of description of books and movies and leveraging the similarities between them.
* Leveraging the history of the user.
* Using multiple datasets from different source as the data.

## API and operations

The structure of this software relies on three layers. The bottommost layer are the algorithms that need to be implemented for the creating the recommendations. The middle layer is the API (Application-Program Interface) which will help in smoother and effective communication between the database and the algorithms. This is basically a multimedia view controller (MVC) that has been created for effective communication with the front-end.



The back-end aspect of our code would include codes that would output a recommendation. The Application-Program Interface (API) would serve this data to the front-end when asked for. The API includes certain routine and protocols so the graphical user interface (GUI) can communicate with the back-end.

We have written a custom API with JQuery-Ajax to communicate between the form that would occur on the front-end the prediction algorithms on the back-end.

The MySQL database would store the user accounts and previous history of the users. Flask framework is used to setup the API that the front-end will communicate with. The following API calls will be used on the front-end side.

POST: /user #Registration of a new user  
GET: /user/username #Information of a particular user  
POST: /user/username/recommendation #Search criteria (relevant recommendation for user)  
GET: /books/bookid #Returns details of book id  
GET: /movies/movieid #Returns details of movie id

## User Interface

The user gets to interact through a web-application portal by first signing up. As soon as the user signs-up, he is given a choice to select up to five media (books and movies) of each which (s)he has read/watched or likes. There is also a search bar option which the user is free to use to look for stuff. This information would be stored in the database (for remembering the user history).

The information and details such as the Title, Author/Director, and Genres, and description is returned of the media chosen by the user on the next page. The next page is a D3-JS framework and thus an interactive web-page where the user can explore the movies, get linked to IMDB or GoodReads for movies and books respectively or enlarge upon the poster.

The final output of the python script on the back-end is a JSON file which will be processed by the user-interface to provide the required output.

## Recommendation Algorithm

For extracting similarity out of texts (description and plots), we used two approaches – the Naïve Bayes “Bag-of-words” and Cosine Similarity Measure. The Bag of Words approach analyzes the description of the media chosen by the user and performs a similarity search based on the number of similar words. It uses Natural Language Processing and Information Retrieval (IR) and performs similarity by building a vector space model. This approach has been often used for classification of texts and models which have more information in common than quantitative features. In fact, this approach is often used to generate features that can be used for better classifiers to classify.

Python’s scikit-learn library for the actual implementation of the algorithm. The “Bag of words” contain a set of words which are the most descriptive of a set of text. A set of “Stop words” are used to make the vectorizer ignore words such as “the”, “a”, “is” etc. This is generated for all the movies in the database and a “dictionary” of words is generated for them.

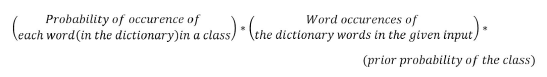
To characterize the set of classes (a sub genre), we compute the probability of occurrence of each word in the dictionary. We use “sub-genre” because, although books are broadly divided by genre, they may fall under completely different themes and the resulting prediction may be irrelevant if we used “genre” alone as a deciding factor. We also check for the occurrence of words in the dictionary of user input’s description. We then follow the Naive Bayes classifier’s hypothesis to determine the probability of the input falling under a given class.

The bag-of-words model is strongly influenced by the frequency. Thus, it does not take the synonyms or implication of the statements into consideration. This is critically important because this is one of the drawbacks of using the bag-of-word approach. For this reason, we restrict performing this approach only for same media recommendation (Book-Book or Movie-Movie).

Likewise, the probability in our model can be calculated by:



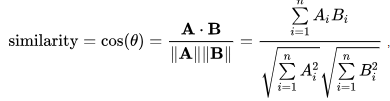
The above probability is computed for each class present in the system and the class with the maximum probability is chosen as the class of the input. The computation of the term we are maximizing, is done as follows:



The prior probability of a class can be calculated by taking the ratio of the number of books in the class to the total number of books.

The cosine approach looks for the similarity between two vectors created based on the title and the description of the movies. This is basically a way to measure two non-zero vectors and if they are exactly opposite, the similarity would be -1 and if they are the same, the score would be 1.



Ai, Bi – Components of the vector

The cosine of two non-zero vectors is derived from the Euclidean dot product formula –



The range would vary from -1 to 1, and the highest scored matrices would be given the highest scores. This approach would consider the similarity between the words but not the implied meanings. Thus, this approach is used for performing cross book-movie recommendation.

Moreover, we used user history to extract genres that the user prefers the most. The user history and text similarity are given equal weightage and then the results are filtered based on ratings of the book/movie and the number of awards won. Apart from that, we performed content-based filtering to recommend our users the best options available.

# Experiments and Evaluation

We performed a simple Multiple-Choice Questionnaire for users. We asked 20 users the following questions –

1. Do you like books, movies or both?

Based on the answers, we asked them to choose 5 books or 5 movies or both

1. Did you like the recommendations? – Rate from 0 to 10

Made them read the description of the books and watch the movies of trailers.

1. Asked them why and why not they would read/watch each recommendation.

Our user-review tells us that on an average, users were satisfied with about 83% of the recommendations. The reason observed for not reading/watching a recommendation would be a social habit as not read or as to not watching movies often.

# Conclusion and Discussion

## Conclusion

The aim of the project was to use hybridization algorithms to provide a high quality cross movie-book recommendation. The current application uses the naïve based “Bag of words” and cosine similarity approach which takes qualitative features such as genres, description and user history and other quantitative features such as ratings, number of awards and other similar stuff into consideration. The performance of our approach was measured using MAE, OEE, RMSE and SD. The personalization of recommendations to users being based on user history and similar features based on text-similarity would lead to automatic improved user-base and develop a social culture of reading books. The limitation of the recommendation is that they do not take the implication of the statements and the meanings of the words into consideration. Furthermore, they also fail to hybridize the algorithms and provide a better recommendation system.

## Future Work

Based on user’s feedback and the current structure of the algorithm, the following steps can be used for better recommendations.

1. Recommendations based on scoring features such as genres, user history, updates on the previous recommendations.
2. Use of Hybridized Algorithms
3. Use of advanced text-based similarity algorithms
4. Features of Books don’t necessarily correspond to Features of movies, thus an approach to balance the two. If there are movies or books which do not have any genres in common, the movie/book will not be recommended.
5. Use of multiple genres to evaluate recommendations

# Distribution of Effort

All team members have contributed similar amount of effort.

# References

1. Gediminas Adomavicius, Tuzhilin A: **Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions**. *IEEE Transactions on Knowledge and Data Engineering* 2005, **17**(6):734-749.

2. Ono C, Kurokawa, M., Motomura, Y., & Asoh, H.: **A Context-Aware Movie Preference Model Using a Bayesian Network for Recommendation and Promotion**. *International Conference on User Modeling* 2007.

3. Yi Zhang JK: **Efficient Bayesian Hierarchical User Modeling for Recommendation Systems. SIGIR.** 2007.

4. Hsu K-W: **A Theoretical Analysis of Why Hybrid Ensembles Work**. *Computational Intelligence and Neuroscience* 2017.

5. Sriram B, Fuhry, D., Demir, E., Ferhatosmanoglu, H. and Demirbas, M.: **Short text classification in twitter to improve information filtering. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval**. *ACM* 2010:841-842.

6. Allison J.B. Chaney BMS, Barbara E. Engelhardt: **How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility**. *Cornell University Library* 2017.

7. Carlos A. Gomez-Uribe NH: **The Netflix Recommender System: Algorithms, Business Value, and Innovation**. *ACM Transactions on Management Information Systems* 2016, **6**(4).

8. Greg Linden BS, Jeremy York: **Amazon.com recommendations: item-to-item collaborative filtering**. *IEEE Internet Computing* 2003, **7**(1):76-80.

9. George Lekakos PC: **A hybrid approach for movie recommendation**. *Multimedia Tools and Applications* 2008, **36**(1-2):55-70.

10. Sang-Ki Ko S-MC, Hae-Sung Eom,Jeong-Won Cha,Hyunchul Cho,Laehyum Kim,Yo-Sub Han: **A Smart Movie Recommendation System**: Springer, Berlin, Heidelberg; 2011.

11. Rahul Katarya OPV: **An effective collaborative movie recommender system with cuckoo search**. *Egyptian Informatics Journal* 2017, **18**(2):105-112.

12. Olivier Chapelle TJ, Filip Radlinski, Yisong Yue: **Large-scale validation and analysis of interleaved search evaluation**. *ACM Transactions on Information Systems* 2012, **30**(1).

13. Dewan Md.Farid LZ, Chowdhury MofizurRahman, M.A. Hossain, Rebecca Strachana: **Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks**. *Expert Systems with Applications* 2014, **41**(4-2):1937-1946.

14. Mouzhi Ge CD-B, Dietmar Jannach: **Beyond accuracy: evaluating recommender systems by coverage and serendipity**. *RecSys'10* 2010:257-260.