Page & Reel: The Cross Book/Movie Recommender

Shubashree Baskar, Brian Merritt, Nirav Shah, Haritha Ramesh, Ryan Place, Irtaza Haider

Introduction - Motivation

There are numerous recommendation systems such as GoodReads, IMDB etc that provide recommendations across a single media type. However, there is no reliable tool, other than tastedive.com, that recommends a different media type. Upon our use, we noticed that the tastedive.com is very poor at predicting a sufficient number of books to users. In addition, it lacks depth as it only recommends items based on it being a similar genre. We propose to create a web based recommendation system to recommend books, movies or both by leveraging shared characteristics of the two media types and the previous history of the user.

Problem Definition

The system will require users to register an account and then log in to view their recommendations. Once logged in, before seeking recommendations, the user may specify parameters to a search: the number of recommendations and the media type. During searching, a short list of title recommendations will be populated, similar to google or amazon search bars. The system will then display the media in an interactive environment once choosing a title. Upon clicking a specific book or movie url, if present, the user will be redirected to the respective GoodReads or IMDB webpage for further curation.

Survey

A variety of techniques have been proposed in literature, from content-based to hybrid approaches that prove useful to our project[7]. [2] presents a technique - Modified EM to learn a new user profile using information from existing user profiles. We will try to leverage this technique to induce information for new users if time permits. [1] presents a Bayesian network based approach to using context to make movie recommendations. However, it works primarily for sparse data. In [3] an approach is presented that classifies 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[5], [6].

Amazon uses item-item collaborative filtering wherein items that are bought together have higher similarity scores [9]. 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. As we are trying to develop a cross book-movie recommender, we have to rely on features that are common amongst the datasets like genre, textual similarity between book and movie descriptions.There are hybrid recommenders that have been proposed in literature[4],[11],[12], which switch between collaborative and content based filtering to make better predictions and use an ensemble of predictors. But due to the above mentioned reason we couldn't implement the hybrid algorithm we proposed[10] in the proposal and instead went for a Naive Bayes classifier based 'Bag of words' model[15]. For textual similarity, we can also experiment with the Lesk algorithm [13] to improve the relevance of results.

During movie recommendations, it was found that determining quality of a review is more accurate when using professional critics as a metric [7]. Also some metrics like serendipity [14], are context driven and are hence difficult to measure. We opted for D3 and the javascript library to allow more flexibility in parsing and displaying data, compared to standard tools such as Tableau[8].

Proposed Method

Intuition

Netflix, Amazon Prime, GoodReads are few of the many examples that currently use top algorithms to predict and suggest recommendations to user based on parameters such as ratings, critiques, plots, history, genres, and many other features [18].

Both movies and books have plot summaries. If a user likes a movie, the assumption is that the user will also like a book of a similar plot. Therefore, we will leverage similarities in the plot summary of the books and movies to provide recommendations to the user. Again, this is something that the other recommendation systems don't do.

Since our system is supposed to recommend movies if given a book and vice versa, what is different in our approach would be to place more importance to the description of the movies and the books rather than just user ratings, popularity, genre etc [16,17]. This can be done by finding the degree of textual similarity amongst the descriptions of books and movies.

Description of our Approach:

There are two major components of our system, the frontend i.e the user interface and backend i.e the service that serves the content for frontend. The flow of the user interface has been

described in the problem definition section. We are using HTML, D3, Javascript and AJAX technologies for the frontend.

For the backend, we use MySQL database to store the user accounts and previous history of users. Flask framework is used to setup the API that the frontend will communicate with. We have the following API calls that the frontend can use:

```
POST:  /user                              #Registers a new user
GET:   /user/username                     #Gets information of a particular user
POST:  /user/username/recommendation      #Sends the search criteria as the body and gets
                                          relevant recommendations for the particular user

GET: /books/bookid                        #Returns details for the given book id
GET: /movies/movieid                      #Returns details for the given movie id
```

For the recommendation algorithm, our approach currently is a Naive Bayes classifier which uses a bag of words model. We use 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 occurence 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.

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

$$h^*(x) = \underset{k}{argmax}\ \Pi_k\ f_{X|Y}(x|k)$$

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:

$$\begin{pmatrix} Probability\ of\ occurence\ of \\ each\ word(in\ the\ dictionary)in\ a\ class \end{pmatrix} * \begin{pmatrix} Word\ occurences\ of \\ the\ dictionary\ words\ in\ the\ given\ input \end{pmatrix} *$$

$$(prior\ probability\ of\ the\ class)$$

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.

After the class is selected, we need to display the five most relevant results to the user. We sort and select the results in terms of the ratings they were previously given, and also based on other factors like language and region. We list results that have maximum textual similarity of descriptions (compared to the input's description) with regards to the dictionary words to ensure maximum relevance.

Say, a user inputs his/her favourite book "Harry Potter", if we were to just take its genre into account, it could fall into a very broad category like fiction. Placing importance on its description to understand its sub genre(fantasy,magic), helps us in making better recommendations and fill in the gaps when genre/sub-genre information of a book is not necessarily available.

The cost of this project will be ten U.S. dollars. This is for acquiring a key in order to use an unofficial imdb api from npm.

Experiments/ Evaluation

We will use three evaluation mechanisms to check the success of our approach

- Survey on recommendation quality: As recommendations are subjective, and there isn't a clear right or wrong answer, we will conduct a survey in which we will ask the users to provide us with both qualitative and quantitative feedback.
- User actions: by scoring the predictions based on if user watches/reads the recommendations or ignores it
- Weekly visits on the website: We will give out the URL of the web application to people and see if people keep coming back to use the website and if the number of visitors increase.

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.

Distribution of team effort

Every author has contributed equally in writing this proposal and has also put similar amount of efforts in the data cleaning, front-end and back-end work that has been done so far. For the future advents, we be dividing responsibilities as Front-End Team(Brian and Ryan), Back-End Team(Shubashree and Haritha), and the API-Team(Irtaza and Nirav), for simplicity and smoother co-ordination.

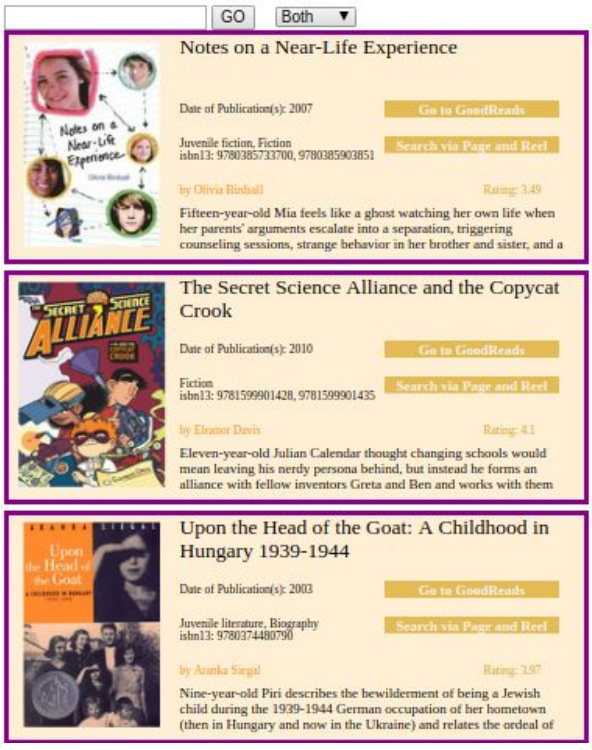
Proposed workflow (from the Proposal):

~~*Data collection from GoodReads, Open Library, and IMDB API - First two weeks*~~
~~*Data Cleaning using Open Refine, Perl, bash scripting - First two weeks*~~
~~*Data Integration using SQLite/Open Refine - Second and third weeks.*~~
Data analysis by our algorithm which was described above - **Ongoing**
Visualization using PHP, Flask-Python (for API), D3, Javascript - **Ongoing**
Buffer and overall completion - Last Week

Results:

To be obtained.

Example page (Visualization):



Discussion and conclusion

Since our algorithm looks at different factors while recommending, the risk of a book/movie being too niche is lessened, but is still present. The risk of creating a feedback loop as stated in the proposal will be mitigated by introducing randomness to our output. The payoff is that due to filtering based on genre, rating and comparing the similarities between the book and movie synopsis, the recommendations will be meaningful to the users. We hope that by the deadline for the project i.e 18 days we will come up with a system that will be worth deploying in the real world.

References

1. Ono, C., Kurokawa, M., Motomura, Y., & Asoh, H. (2007). **A Context-Aware Movie Preference Model Using a Bayesian Network for Recommendation and Promotion**. International Conference on User Modeling .
2. Yi Zhang, J. K. (2007). **Efficient Bayesian Hierarchical User Modeling for Recommendation Systems**. SIGIR.
3. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H. and Demirbas, M., 2010, July. **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 (pp. 841-842). ACM.
4. Hsu K-W: **A Theoretical Analysis of Why Hybrid Ensembles Work.** Computational Intelligence and Neuroscience 2017.
5. Allison J.B. Chaney BMS, Barbara E. Engelhardt: **How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility**. Cornell University Library 2017.
6. Carlos A. Gomez-Uribe NH: **The Netflix Recommender System: Algorithms, Business Value, and Innovation**. ACM Transactions on Management Information Systems 2016, 6(4).
7. Adomavicius, G., & Tuzhilin, A. (2005). **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, 17(6), 734-749.Schaible, J., Carevic, Z., Hopt, O., & Zapilko, B. (2015). Utilizing the Open Movie Database API for Predicting the Review Class of Movies. In KNOW@ LOD.
8. Nair, L. R., Shetty, S. D., & Shetty, S. D. (2016). **Interactive visual analytics on Big Data: Tableau vs D3. js.** Journal of e-Learning and Knowledge Society, 12(4).
9. Greg Linden BS, Jeremy York: **Amazon.com recommendations: item-to-item collaborative filtering**. *IEEE Internet Computing* 2003, **7**(1):76-80.

10. 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.
11. Claypool M, Gokhale A, Miranda T, Murnikov P, Netes D, Sartin M (1999) **Combining content-based and collaborative filters in an online newspaper.** In Proceedings of the ACM SIGIR workshop on recommender systems. Berkeley, CA,http://www.csee.umbc.edu/~ian/sigir99-rec/
12. George Lekakos PC: **A hybrid approach for movie recommendation**. *Multimedia Tools and Applications* 2008, **36**(1-2):55-70.
13. Michael Lesk. 1986. **Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone.** In Proceedings of the 5th annual international conference on Systems documentation(SIGDOC '86), Virginia DeBuys (Ed.). ACM, New York, NY, USA, 24-26.
14. Mouzhi Ge CD-B, Dietmar Jannach: **Beyond accuracy: evaluating recommender systems by coverage and serendipity**. *RecSys '10* 257-260.
15. Kim SB., Rim HC., Yook D., Lim HS. (2002) **Effective Methods for Improving Naive Bayes Text Classifiers.** In: Ishizuka M., Sattar A. (eds) PRICAI 2002: Trends in Artificial Intelligence. PRICAI 2002.
16. 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.
17. Rahul Katarya OPV: **An effective collaborative movie recommender system with cuckoo search**. Egyptian Informatics Journal 2017, 18(2):105-112.
18. Olivier Chapelle TJ, Filip Radlinski, Yisong Yue: **Large-scale validation and analysis of interleaved search evaluation.** ACM Transactions on Information Systems 2012, 30(1).