https://doi.org/10.48009/2_iis_2014_321-327
Issues in Information Systems
Volume 15, Issue II pp. 321-327, 2014

RECOMMENDER SYSTEM FOR ANIMATED VIDEO

Justin Miller, Georgia Southern, justin_j_miller@georgiasouthern.edu

ABSTRACT

Finding entertainment in the domain of animation is a challenging process that often forces many customers to ask others for assistance in forums or chat rooms. The manual process of recommending shows to one another based on often flimsy examples prevents many users from efficiently identifying media that suites their taste. Recommender systems, software tools and techniques for providing suggestions of items to a user ***[12]*** *, present an exceptional use case for resolving this problem. In this paper we compare the various implementations and benefits of using the collaborative filtering (user based approach). Improving the identification of animation customers are interested in is a problem domain recommender systems are well suited for, benefiting both customers and vendors of such media. The aim of this study is to provide a proof of concept system capable of providing valuable recommendations based on show rankings.*

Keywords: Recommender system, weighted voting, collaborative filtering, content-based filtering, matrix factorization, Representational state transfer (REST) Application programming interface (API)

INTRODUCTION

Animation encompasses a vast array of subjects, art styles, and cultural preferences in the area of entertainment. There are foreign films such as Persopolis or Les Triplettes de Belleville that examine social institutions or cultural differences. American films such as Coraline, The Nightmore Before Christmas, and many Disney films have touched the hearts of millions. Japenese animation encompasses a large number of subject matters that, through the medium of animated video, can be explored and expressed in a way that would not otherwise be possible. The technology to make recommendations for this medium has lagged behind the times. Often, the only way to find suggestions based on taste is to inquire via chat rooms or websites. This is a tedious and unreliable process.

Typically a video recommendation site that does not utilize collaborative filtering would require users to input very general categories of movies that they would like, and would only be able to recommend movies based on very general guidelines (such as genre, number of votes, and average ranking). Such systems are known as weighted voting systems [10] and they are unable to provide tailored results based on personal preference. This result is limited because different users may have widely differing opinions on the type of show they enjoy.

STATE OF THE ART REVIEW

Recommender systems provide a competitive advantage in the world of both subscription-based digital distribution (Examples being Netflix, Hulu, and Redbox) and e-commerce (Amazon). These systems are providing a competitive advantage by doing things like

- Enhancing customer service by recommending interesting content
- Driving profits by advertising interesting products
- Providing metrics for future capital investment

The single largest clear driver of improvement in the field was “The Netflix Prize”. Cash prices and glory were offered for any who could deliver significant improvements in the accuracy of recommendations.

The Netflix Prize

The Netflix Prize was a major driver in the innovation in the world of Collaborative Filtering (CF). The competition boosted interest in the field of CF, and resulted in the publication of several new methods [11]. The competition was open to anyone with the goal of predicting user ratings for films using only previous ratings as input. The competition began on October 2nd, 2006 and ended on September 21st, 2009 with the grand prize being awarded to the *BellKor’s Pragmatic Chaos* team which had improved the RMSE of Netflix’s own system by 10.06% [1].

Amazon Recommendations

The internet retailer Amazon.com provides a real world example of how recommendations can improve both customer service and revenue. They use their system to support targeted email marketing and provide a personalized shopping experience most of the web sites' pages in order to create a personalized shopping experience for each user. In order to support their need to scale Amazon designed an item-to-item collaborative filtering technique that matches each of the customer's purchased and rated items to similar items, then combines those similar items into a recommendation list. [9].

RECOMMENDER SYSTEMS

Recommender systems advise users on relevant products and information by predicting interest based on various types of information [6]. There are five general approaches to recommendation techniques used when deciding on how to leverage the current data set in order to provide recommendations. [2]

- Collaborative Filtering (CF): Predictions are made based on rating profiles for different users. Users with a similar rating history are located and recommendations are made based on the neighborhood or matrix factorization model. [5]
- Content-based: Classifier systems are derived from machine learning. [3]
- Demographic: Distinguishes users by demographic profiles and makes recommendations based on said profiles.[8]
- Knowledge-based: Uses knowledge about users and products to generate a recommendation. [3]
- Hybrid: Any recommender system that combines multiple of the above recommendation techniques together to produce its output. [2]

Deciding which type of recommender to use is driven by the data that is available and the use case you want to make out of it. The primary approach to providing recommendations from users based on previous explicit feedback is collaborative filtering. The two standard methods used to implement a collaborative filtering approach are the *neighborhood based approach* (NB) and *matrix factorization* (MF). The neighborhood approach centers on finding users or items which are similar to each other. Matrix factorization uses vectors of factors derived from item ratings patterns to represent users and items [7]. The implemented system uses matrix factorization because the implementation is extremely easy to parallelize and tends to render more accurate results (accuracy being rated as a lower RMSE).

Neighborhood Model

The neighborhood approach centers on finding users or items which are similar to each other. The item-oriented approach evaluates a user's preference for an item based on ratings of "neighboring" items by the same user [7]. The most common formula used to calculate similarity is known as the Pearson correlation coefficient. The similarity measure between two items is measured as S_{ij} where S_{ij} describes the similarity between *i* and *j*. Estimating the unknown r_{ui} value for user *u* for item *i* is taken as the weighted average of the ratings for neighboring items. [5]

$$r_{ui} = \frac{\sum_{j \in S^k(i;u)} s_{ij} r_{uj}}{\sum_{j \in S^k(i;u)} s_{ij}} \qquad (1)$$

The neighborhood model is not the implementation of choice for this system, but it was the standard choice in all earlier CF systems. Current CF systems have adopted a newer approach known as matrix factorization.

Matrix Factorization Approach

Matrix factorization uses vectors of factors derived from item ratings patterns to represent users and items [7]. This method places ratings in a matrix with one dimension representing items and the other representing users. Operating with known ratings provided by users is known as using *explicit feedback*, and this often results in a sparsely populated matrix, since any single user is likely to have rated only a small amount of the possible items.

For the following formulas in dimension f comprising users x items:

- Each item i is associated with a vector: $q_i \in R^f$
- Each user u is associated with a vector: $p_u \in R^f$
- The approximate rating u gives to item i : $\hat{r}_{ui} = q_i^T p_u$

In order to solve for r_{ui} it is necessary to have a mapping of each user and item to factor vectors $q_i, p_u \in R^f$.

This is a challenging task to accomplish, requiring us to convert a sparsely populated matrix into a dense one with ratings for every value. To learn the factors p_u and q the system uses previously known ratings to minimize the regularized squared error.

$$\min_{q*,p*} \sum_{(u,i)\in K}(r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \qquad (2)$$

With K being the set of (u, i) pairs of which r_{ui} is known. The constant λ controls regularization, and to avoid over fitting the data this should be determined using cross-validation.

Equation 2 still has two unknowns, q_i and p, which prevent a quadratic solution. The *alternating least* square (ALS) learning algorithm is the approach Oryx uses to minimize equation 2. ALS fixes one of the two unknowns to a random value and solves for the other. It then takes the resolution of the previous step and inserts it into the equation, solving for the original substituted value. It rotates between fixing the qi and pu values. When all of the one is fixed then the other is recomputed, and vice versa. This makes sure that every step results in a lower result from Equation 2 until convergence is achieved [7]. The computation step of each can be done in parallel, greatly enhancing the scalability of the system.

SYSTEM ARCHITECTURE

The system designed for this project required several components working in tandem. The following section outlines each component, the role it played, and how it integrated into the overall architecture of the system.

I. The MyAnimeList Dataset

The dataset used consists of data from the real anime rating site http://myanimelist.net/. The site hosts profile pages with which users can track the shows they have watched along with a ranking from 1 to 10. From this site the profiles of 465,494 users were scraped yielding a total of 41,155,170 rankings for 32,415 items. The data model describing the extracted data is modeled below.

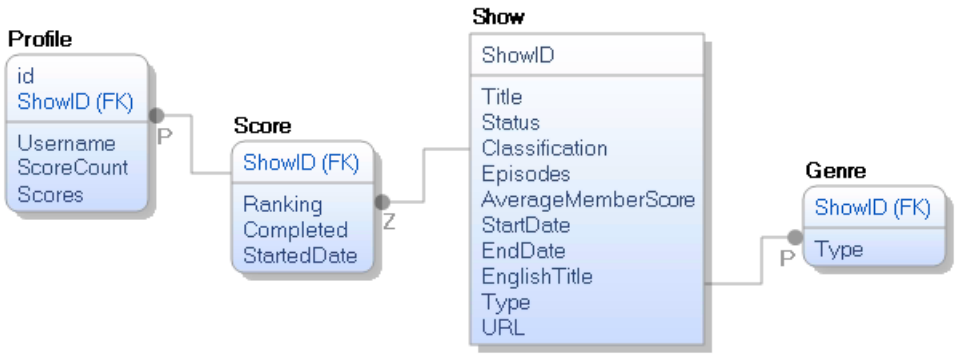

Figure 1- Data model for system

II. Pentaho Data Integrator (PDI) for ETL

The source of all data for the system was JSON scraped from the website http://myanimelist.net/. Two datasets were extracted, one relating to profiles and rankings, and one with metadata about each show. The profile and ranking data was in a form that would require removing discrepancies and transforming the data into the format that Oryx would expect. Rankings were extracted from profiles each profile, profiles without rankings were discarded, entries that had not had a rank assigned were removed, and the interesting values were selected and formatted to CSV.

Figure 2 - ETL flow for ratings

III. Recommendation Engine: Oryx

Two major projects were evaluated as potential machine learning frameworks to support this project. The first was Apache Mahout and the second was Cloudera Oryx. Both are open source frameworks that seek to provide a computation "engine" for typical machine learning activities. Each project has various advantages and disadvantages that must be evaluated before a decision of which is more appropriate can be made.

The Oryx open source project (previously Myrrix) provides a simple, real-time large-scale machine learning predictive analytic infrastructure. Oryx provides applies several data mining techniques to provide collaborative filtering / recommendation, classification / regression, and clustering.

The Oryx architecture depends on two independent pieces, the real-time serving layer and batch computation layer, working together to enable exhaustive real-time recommendations. Recommendations are provided by factoring the user-item into user-feature and feature-item matrix.

The Batch Computation Layer

- Builds machine learning model from input data.
- Computes recommendations/clustering.
- Leverages Apache Hadoop to scale out processing.

The Real-Time Serving Layer

- Exposes a REST API.
- Partitionable to scale.
- Interfaces with Computation Layer.

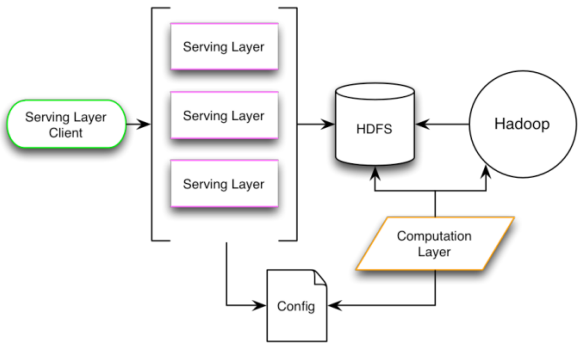

Figure 3- Oryx system architecture

Batch input is loaded by putting a CSV file (formatted as profile, item, ranking) into the input directory of the first generation. Upon start of the computation layer Oryx will import this data and create a user-item matrix from the rankings. The user-item matrix is factored into a user-feature and feature-item matrix and

using the Alternating Least Squire algorithm value estimations are made until the root mean squared error (RMSE) is lowered within bounds [9].

New data is loaded in real-time over the REST API exposed by the serving layer. This allows for quick integration of new data into the model supporting changes in profile or new/updated rankings for users.

Benefits of Oryx over Mahout:

- Uses matrix factorization approach instead of neighborhood models for collaborative filtering
- Out of the box integration with Apache Hadoop
- Presents easy to use API for integration of data
- Uses "black box" approach that "just works" with minor configuration.
- Includes the REST API for interfacing with the system.
- Returns results significantly faster than Mahout.

Drawbacks of Oryx over Mahout:

- Oryx is still in alpha.
- Documentation is extremely sparse.
- Paid support is nonexistent.

Recommendation results are returned by calling the REST API endpoint with a GET request in the form of /recommend/[userID](?howMany=n)(&considerKnownItems=true|false)(&rescorerParams=...). By default on the top 10 results are returned but this is tune-able by adding the "?howMany=#" argument to the URL. The CSV output contains one recommendation per line, and each line is of the form itemID, strength, like 325, 0.53. Strength is an opaque indicator of the relative quality of the recommendation.

Oryx was chosen because it appears to be at the very bleeding edge of the field. In area of collaborative filtering I found it to be using the most cutting edge approach enabling extremely accurate and fast recommendation results. Mahout was much more difficult to configure, was significantly slower, and provided lower quality results.

IV. PostgreSQL for Metadata Persistence

Although Oryx makes storing profiles and rankings unnecessary for supporting recommendation operations it does not store meta-data about each show. Presenting recommendations to users through a web interface requires more than just an obscure id number for a show. For this reason it was necessary to implement a persistent data store that houses the meta-data that correlates to the Oryx item-id for each show.

The data model that describes the data stored in the database. The *ShowID* directly relates to the ID that Oryx stores for recommendations and scoring.

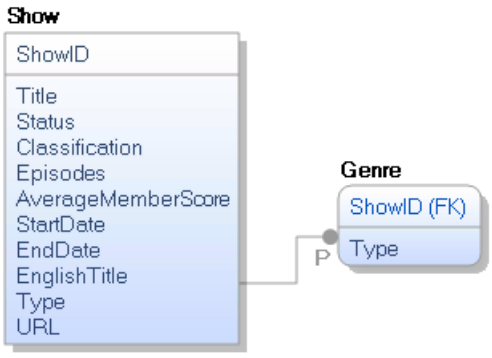

Figure 4 - Data Model for web layer

Presentation of recommendations is orchestrated by the web framework which joins the recommendations from Oryx to the meta-data stores in PostgreSQL.

V. Play Framework for Web Presentation

The orchestration of Oryx API calls and joining/presenting of meta-data related to such calls is done by the Play Framework. Play is an open source MVC web framework with the language of choice being Scala. In order to present the relevant meta-data it is necessary to create logic that will first call Oryx for the recommended show IDs and then use the keys provided by Oryx to pull the associated meta-data from the database.

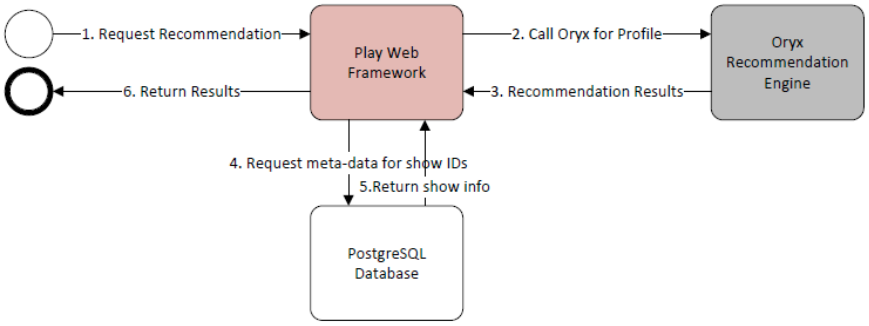

Figure 5 - User recommendation request life cycle.

The URL pattern which requests recommendations of a profile is below:

Table 1 – Endpoint API targets and descriptions.

URL Endpoint	Description
http://servername/	Home page which will by default display the current most popular items. Results contain an ordered list of shows with the title, average score, type, number of episodes, and genres listed.
http://servername/[profileName]	Primary endpoint which presents a recommendations list for the said profile. Results contain an ordered list of shows with the title, average score, type, number of episodes, and genres listed.
http://servername/[profileName]/why/[itemID]	Presents a list of shows that affected the recommendation of the requested item. Results contain an ordered list of shows with the title, average score, type, number of episodes, and genres listed.

An example of site presentation is included below.

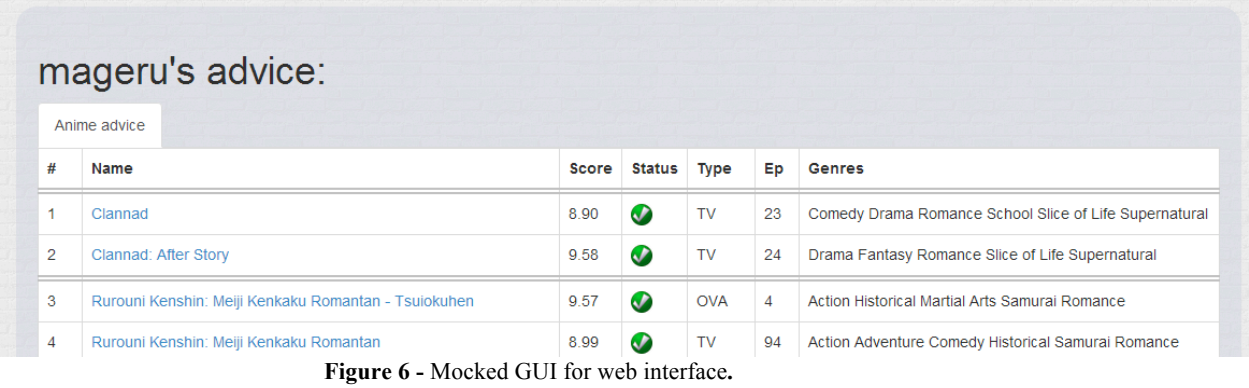

Figure 6 - Mocked GUI for web interface**.**

CONCLUSIONS

It has been shown that a collaborative filtering based recommender system can provide good recommendations to users seeking accurate customized suggestions for animated video. A combination of techniques and systems can be combined to provide fast results based on a user profile containing a sufficient number of rated videos. The system could be further augmented to provide fresh results by setting up automated jobs to scrape the site, clean the data, and load it into Oryx. An alternative to this approach would be to have users supply ratings directly to our system that would circumvent the batch processing approach that scraping would present. The most apparent limitation of a recommendation system utilizing this technology is that it requires a massive amount of profile rankings before it can accurately provide recommendations. This would mean that any new shows would suffer the "cold start" problem until a sufficient amount of users provide rankings. I believe the system I build and described in this paper provides the ideal model for a recommender system for animated video.

Recommendations system can be leveraged to enhance the customer experience by directing users to content they might not otherwise have discovered. I believe that guided direction will to be a standard feature in e-commerce and video content provider systems in the future.

REFERENCES

1. J. Bennet and S. Lanning, "The Netflix Prize," KDD Cup and Workshop, 2007; www.netflixprize.com.
2. Burke, R. (2007). Hybrid web recommender systems. In The adaptive web (pp. 377-408). Springer Berlin Heidelberg
3. Burke, R. (2000). Knowledge-based recommender systems. Encyclopedia of library and information systems, 69(Supplement 32), 175-186.
4. Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.
5. Hu, Y., Koren, Y., & Volinsky, C. (2008, December). Collaborative filtering for implicit feedback datasets. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on (pp. 263-272). IEEE.
6. Huang, Z., Chung, W., Ong, T. H., & Chen, H. (2002, July). A graph-based recommender system for digital library. In Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries (pp. 65-73). ACM.
7. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
8. Krulwich, B. (1997). Lifestyle finder: Intelligent user profiling using large-scale demographic data. AI magazine, 18(2), 37.
9. Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE, 7(1), 76-80.
10. Nordmann, L., & Pham, H. (1999). Weighted voting systems. Reliability, IEEE Transactions on, 48(1), 42-49.
11. Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2008, October). Matrix factorization and neighbor based algorithms for the netflix prize problem. In Proceedings of the 2008 ACM conference on Recommender systems (pp. 267-274). ACM
12. Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008). Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management* (pp. 337-348). Springer Berlin Heidelberg.