Fanfiction Recommendations: An Empirical Study in Matrix Factorization (Rough Draft)

Joshua Gang
jegang@rutgers.edu

John Blackmore
john.blackmore@rutgers.edu

ABSTRACT

In this paper, we will explore a few of the more prominent matrix factorization methods used to process very large data sets. We will apply these methods individually to a corpus of fanction stories to predict the likelihood of a story being selected as a user's favorite. We will objectively compare the results across from each method and discuss our conclusions. At a high level, we wish to explore the efficacy of collaborative filtering methods versus content-based methods. Which methods are more informative for predicting the likelihood that a fanction story will be chosen as a favorite, given a user's profille of favorite stories and all other user profiles?

Keywords

Latent Dirichlet Allocation; Singular Value Decomposition; Dimensionality Reduction; Collaborative Filtering; Topic Modeling; Fanfiction; Recommendation Systems

1. INTRODUCTION

Recommendation systems are systems that try to predict choices that a user will make. Usage of recommendation systems has grown substantially over the past few years, as different companies and services attempt to use them in order to present more products to their customers. These predictions are most often predicated on the choices the user and other users have made in the past. Given the proliferation of user information available to online retail and social networking systems, more sophisticated systems can make very good choices.

There are typically tradeoffs to consider, however, between the prediction accuracy and system resources required, especially as datasets grow massively large. As the matrix of products and users grows beyond the confines of the computing resources available, it becomes critical to reduce the dimensionality of the data, but with as little loss of accuracy as possible. Indeed, there is strong motivation to improve both the accuracy and efficiency of these systems.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

In this paper, we will apply several recommendation system techniques to a new dataset, gathered from Fanfiction.net[1], a site that has no recommendation system in place. We will present our comparative results along with total runtime and system memory requirements. We will then present our conclusions and opportunities for future work.

2. RELATED WORK

State-of-the-art recommendation systems either draw from some notion of topic or concept similarity to prior choices made by the user, or from some measure of similarity to other users and the choices they have made, or some combination thereof[2]. For example, if a customer always goes to Amazon.com to buy coffee, they will likely present coffee to the customer the next time he/she logs on. Likewise, the system may also show non-dairy creamer, if many of the users that buy coffee also buy creamer, even if the customer has never bought it before.

Among the best known recommendation systems are Netflix and Amazon. Netflix uses collaborative filtering to predict user ratings of movies they haven't seen yet. The ability to recommend material that is enjoyable to the user adds value to their product. Amazon employs recommendation systems based on additional information such as products viewed, products bought by other users that viewed this product, and other metadata not directly related to a purchase.

3. BACKGROUND

There are a number of matrix factorization methods that have been applied successfully across a wide range of recommendation systems. Different methods may yield different results on different datasets. We do not know of a single technique that is universally adopted for all recommendation systems. For our study, we have elected to focus on fanfiction. Fanfiction is a work of fiction about characters or settings drawn from an original work of fiction, usually created by fans rather than by the original author. Each work of fanfiction takes place inside one or more "fandoms", or universes.

This corpus is a good choice for our research because (a) it has no recommendation system, (b) there are millions of users and millions of stories, (c) users can have "favorite" stories, which means that collaborative filtering approaches

[1]http://www.fanfiction.net/
[2]http://www.ibm.com/developerworks/library/os-recommender1/

can be applied, and (d) we can leverage the text of the stories, summary, and title to derive a topic model. We have several methods that can be applied, and we wish to determine, empirically, which of them will be most accurate in predicting a user's favorite fanfiction stories? Which is most efficient in terms of runtime and memory usage?

4. PROPOSED APPROACH

We present the following approaches to collaborative filtering and content-based filtering, which we will then evaluate on our dataset. Collaborative filtering looks into information about what similar users have liked, and makes no assumptions about the underlying properties of the items. Content-based filtering makes use of the underlying properties of items, and uses those properties in order to make predictions.

4.1 Collaborative Filtering

For our "collaborative filtering" approaches, we can only estimate likelihood of a story being a favorite if other users already have it as a favorite. We cannot make predictions for new, unseen stories.

4.1.1 User Similarity Rankings (USR)

Our User Similarity Rankings (USR) approach was to use information about what other users have liked based off of other users who have liked the same stories. For every story, we used the following equation to determine the rank of the story for the user[3]:

$$R_u(s) = \Sigma_{v \in F_{s,u}} \frac{|S(u) \cap S(v)|^2}{20 + |S(v)|} \quad (1)$$

where $R_u(x)$ is the rank of a story s for user u, v is another user, $F_{s,u}$ is the set of all users who have favored story s, and $S(u)$ are the stories favored by user u.

This approach tends to recommend stories that are favored a lot, as they will appear in more favorites lists. As such, it is very likely for there to be a story in the heldout set to be ranked highly, since it is likely for there to be a popular story in the heldout set.

4.1.2 Singular Value Decomposition (SVD)

Singular Value Decomposition is a method of factorizing large matrices into three smaller matrices, which when multiplied together return back the original matrix. Formally, SVD is

$$M = U\Sigma V^T \quad (2)$$

where M is a m x n matrix, whose entries come from a field K, U is a m x m unitary matrix, which relates documents to concepts, Σ is a m x n non-negative diagonal matrix, which tells us the strength of each concept, and V^T is a nxn unitary matrix over K, which tells us how the users relate to the concepts.

We can make use of this process to create a truncated-SVD, where for a threshold t, we only keep the columns in Σ corresponding to the t highest values. This allows us to keep only the most important concepts. This is shown as

$$\tilde{M} = U_t \Sigma_t V_t^T \quad (3)$$

[3]http://colah.github.io/posts/2014-07-FFN-Graphs-Vis/

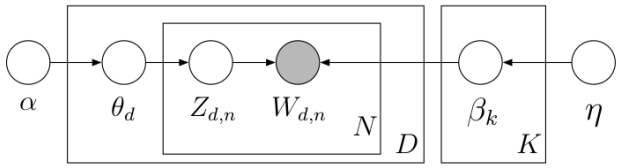

Figure 1: A graphical representation of LDA. The darkened node, $W_{d,n}$, shows what we actually observe, that is, the words in each document.

where

$$M \approx \tilde{M} \quad (4)$$

where U_t is a m x t matrix, Σ_t is an t x t matrix, and V^T is a t x n matrix.

4.2 Content-based Filtering

For our "content-based filtering", we estimate the likelihood of a story being a favorite based on its similarity to other stories in a user's favorite list. While we cannot take advantage of the information from other users, we can make predictions for new, unseen stories.

4.2.1 User Vectors

We attempted to create a "ideal story" vector for each user. Given all of the stories that a user has favored, we transform each of the stories into a normalized 138,277-length feature vector. This vector was created by taking all of the metadata for each of the stories, then converting them into a combination of binary and non-binary feature vectors, then concatenating all of the vectors. The names of the features and their size can be found in Table 1. We then averaged all of these vectors together to create the ideal-story vector for the user. We then computed the Euclidean distance between each story in the corpus and this user-vector, and ranked the stories by these distances.

4.2.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA)[1] is a topic modeling approach that defines a document as a probabilistic distribution over topics, which is a probabilistic distribution over words. We can thus reduce a vector of word counts with dimension n (a vocabulary of 1,579,391 unique tokens!) to a vector with dimension k, where $k = 150$, the number of topics chosen.

$$\begin{aligned} p(\beta, \theta, \mathbf{z}, \mathbf{w}) =& (\Pi_{i=1}^{k} p(\beta_i|n))(\Pi_{d=1}^{D} p(\theta_d|\alpha) \\ & \times \Pi_{n=1}^{N} p(\mathbf{z}_{d,n}|\theta_d) p(\mathbf{w}_{d,n}|\beta_{1:k}, \mathbf{z}_{d,n})) \end{aligned} \quad (5)$$

The generative model of this joint distribution is depicted graphically in Figure 1. It defines a posterior probability, as shown in Equation 5, where z is the $d \times n$ per-word topic assignment matrix, θ the per-document topic proportions, β the topics (k), and $\mathbf{w}$ is the $d \times n$ matrix of word occurrences observed in each document. The *hyperparameters* α and η as well as the number of topics k are inputs to this model.

4.3 Evaluation Metrics

We will evaluate our results from each method using two primary metrics: Mean Reciprocal Rank and Recall@θ. Both

Name of Feature	Description	Type	Size
Wordcount	How long the story in, in words	int	1
Published	When the story was first published	int	1
Updated	When the story was last updated	int	1
Reviews	How many reviews a story has gotten	int	1
Chapters	How many chapters a story is	int	1
Completed	Whether or not a story is completed. Many stories are uploaded in incomplete states and are periodically updated.	boolean	1
Favorites	How many favorites a story has, roughly the measure of its popularity	int	1
Category	The fandom that the story is in	boolean	45366
Rating	The rating of the story: K, K+, T, or M	int	1
Language	What language the story is written in	boolean	44
Tags	The character tags associated with the story; this is usually who the story focuses on, the main characters.	boolean	92859

Table 1: The features that went into the user-vector story transformations.

metrics are on a 0-1 scale, and the closer to 1 the better, as this means that our top predictions are in our held out set. To define our notation, n is our number of users, Q are the query results for that user, and $rank_x$ is the rank of story x in query Q.

4.3.1 Mean Reciprocal Rank

The Mean Reciprocal Rank, or MRR, is defined as

$$MRR = \frac{1}{|n|}\Sigma_{i=1}^{n} max_{x \in Q_i}(\frac{1}{rank_x}) \quad (6)$$

Reciprocal rank is simply $\frac{1}{rank_x}$. Mean reciprocal rank is the average reciprocal rank over all users.

4.3.2 Recall@θ

Recall@θ is defined as

$$Recall@\theta = \frac{1}{n}\Sigma_{i=0}^{n}\Sigma_{x=1}^{|Q_i|}\begin{cases}1 \text{ if } rank_x < \theta \\ 0 \text{ otherwise}\end{cases} \quad (7)$$

This tells us how many of the held out favorites we return in our first θ predictions for each user.

5. EXPERIMENTS

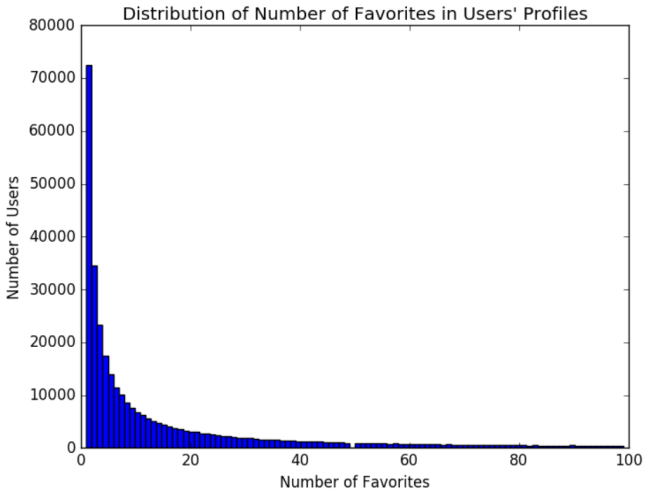

Figure 2: Number of Favorites versus the Number of Users with that amount of favorites. The graph is truncated at the 100 favorites mark; it extends up to 5000 with the same decreasing pattern.

5.1 Approach

We have applied the approaches to our fanfiction dataset and obtained results in terms of our MRR and Recall@θ metrics. We have completed four different experiments, one per method. We wish to determine which method will yield the most accurate predictions for favorite fanfiction stories.

5.2 Dataset

We extracted all of the data for our experiments from Fanfiction.net. Fanfiction.net is one of the most popular online archives of fanfiction, with over 7 million users and 5 million stories on the site. We wrote a scraper and selected 1 million random profiles to add to our dataset. Due to not every user profile being active, and that not every profile has either stories that they've written or that they've favored, this left us with 432,404 user profiles. Out of these, 368,489 users have favorites, while the others only have stories they have written.

The number of favorites that a typical user has is rather skewed (see Figure 2). 199,595 of the users have less than 10 favorites. 147,839 users have less than 5 favorites. There are 149 users who have the maximum number of favorites allowed on the site, 5000. The average number of favorites is 60, with a standard deviation of 233, while the median is 8 and the mode is 1.

For all of our experiments, we limited our dataset to users with 10 or more favorites, which left us with 168,897 profiles to work with.. We then then selected a random subsample of 1000 users to evaluate our methods. For each of the users, we split their favorites into 5 folds, and we tested on 80% of their favorites to try to predict the heldout 20%.

5.3 Details

5.3.1 SVD

We used Truncated-SVDs in order to do user approximations. We assembled a User to Favorites matrix, size

(168,897, 3,314,324), and decomposed the matrix into a 1000 rank SVD matrix. This process took 4.5 hours to complete. Then for each of the users, we took their favorites vector and transformed it into the lower dimension, 1000 rank space. We then inverse transformed the resulting vector, to get an approximation of the original vector. This approximation results in a dense vector that should propagate the concepts that the user enjoys into the stories that they represent. We then take the vector and use this as our rankings.

5.3.2 LDA

We trained the LDA model using the online LDA Python implementation[4] provided by the gensim package. This package allows for training the model in online mode[2], so as to avoid the necessity of loading topic vectors for the entire corpus into memory at once. In this manner, we were able to train the model from the summaries of over 3 million stories.

The values of the hyperparameters α and η were chosen based on the perplexity values of the models at $k = 5$. We selected $\alpha = 1$ and allowed η to be derived from the data. We reduced the size of the vocabulary (n) by eliminating all the words or tokens that appeared only once in the corpus.

Initially, we implemented a similarity metric as the max cosine similarity between a story in the corpus and each of the user's favorite stories used for training, however that meant it would take longer to process users that had more favorites, and runtime was already a concern. Consequently, we computed the mean topic vector for all the user's favorites in the training sample, then computed cosine similarity for each story in the corpus with the mean vector.

5.3.3 Computing Resources

All experiments were run on the Westeros CBIM Cluster. The cluster compromises of 4 machines running Ubuntu 14.04, with jobs organized by the Sun Grid Engine. Each machine has 32 Intel Xeon CPU E5-4650 @ 2.70 GHz cores and 128 GB of ram. Constraints were put onto the number of jobs running simultaneously on each machine to avoid thrashing. Whenever possible, we utilized preprocessing and hashing techniques within each job in order to speed up the overall computations.

5.4 Results

The results for each method are summarized in Table 2. The graph in Figure 3 shows the recall curves for each method at a range of θ values from 0 to $100,000$.

The best performing method, both in terms of results and memory was User Similarity Rankings (USR), followed by SVD. SVD performed better with respect to time. The reason that USR performed so well is likely due to how well it identifies popular stories as being popular, and ends up recommending those most of the time. Each fandom tends to have their own set of popular stories, so a user who has a preference of stories from a single fandom is likely to have seen them. Additionally, since USR doesn't require a full search throughout the entire corpus, its running time remains rather small in most cases. USR is the only one of the methods that is not constant-time in relation to the input, which does cause variations.

SVD managed to capture many of the intrinsic properties of a user's likes. The larger the rank of the SVD matrix, the better the approximation of the original user/favorite matrix

[4]https://radimrehurek.com/gensim/models/ldamodel.html

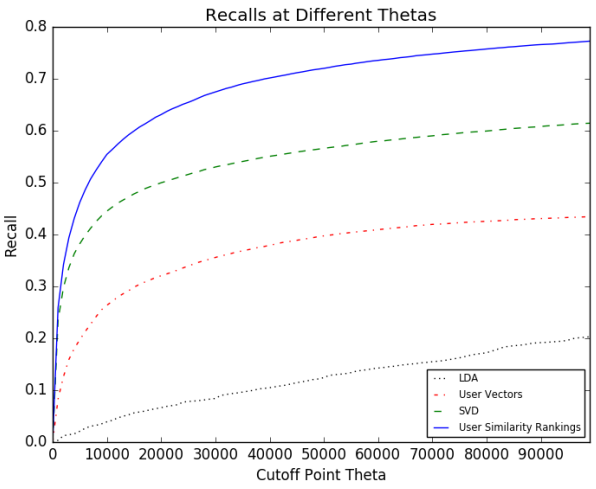

Figure 3: Recall@θ for each method.

that we can capture, and thus the predictions become more and more accurate. It's highly likely that if we were to train a larger rank SVD, then we would get even better results, but due to time and memory constraints this is best left for future work.

Interestingly, both of the feature-based, out-sample methods performed rather poorly when compared to the in-sample methods. This could be for several reasons: for User Vectors, it's likely that a better distance metric than just the plain Euclidean metric should be used, due to the differences in the size of the categories for each feature vector (as seen in Table 1). This method also doesn't take into account that a user might have a very wide variety of stories that they've enjoyed, thus taking the average of two stories that are polar opposites to one another is unlikely to come up with a story that is like either of them. It is unlikely that the user would enjoy this story, and as such would not appear in his favorites list.

The results for LDA were disappointing, though not all that surprising. Due to the inherent nature of fandoms, most of the stories within a fandom share much of the same vocabulary. As such, it can be difficult to distinguish the differences amongst stories based off of their summaries without any semantic parsing and understanding, as the words amongst different summaries are highly likely to repeat.

6. CONCLUSIONS AND FUTURE WORK

We have successfully implemented four alternate methods for predicting the likelihood of a fanfiction story being a user's favorite. Our results have clearly demonstrated that the User Similarity Rankings (USR) approach yields the strongest results in terms of both metrics.

With additional effort, we feel we can improve upon the other approaches. For SVD, we can try to use a higher rank. For LDA, we can adjust k, α, or η, or alter the vocabulary by adjusting the stopwords list. We could also try training a topic model specific to each *fandom*, for example have a topic model that would be used for only *Harry Potter* fanfiction stories. Then it may be better able to make distinctions between the various *Harry Potter* stories.

Method	MRR	SD	Recall @1k	Recall @50k	Recall @100k	Training Time	Memory Usage	Total Runtime (hours)
USR	**0.157**	0.291	**0.259**	**0.719**	**0.773**	0	**250 MB**	70.8
SVD	0.134	0.271	0.235	0.565	0.614	4.5 Hours	50 GB	**28.5**
U-Vecs	0.024	0.099	0.083	0.396	0.434	0	13 GB	112
LDA	0.001	0.019	0.004	0.122	0.203	2.5 Hours	9GB	142.5

Table 2: Results for a random sample of 1000 authors. Best results are in bold.

Further, we may be able to combine some or all of these methods. We were initially optimistic about the efficacy of Collaborative Topic Modeling (CTM), as applied to recommendation of scientific articles by Wang and Blei in 2011[3]. If we can improve the results of topic modeling for this corpus, that may be worth exploring, but for the time being, there seems to be more promise in the top two methods, USR and SVD.

7. ACKNOWLEDGMENTS

We would like to thank Zakary Littlefield for all of his help with managing the Westeros Cluster.

8. REFERENCES

[1] B. A. Frigyik, A. Kapila, and M. R. Gupta. Introduction to the dirichlet distribution and related processes. *Department of Electrical Engineering, University of Washington, UWEETR-2010-0006*, 2010.

[2] M. Hoffman, D. Blei, and F. Bach. Online learning for latent dirichlet allocation. *Neural Information Processing Systems (NIPS) 2010, Vancouver*, 2010.

[3] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. *In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456, 2011.