Evaluating Item-Item Similarity Algorithms for Movies

Lucas Colucci
Prachi Doshi
Kun-Lin Lee
Jiajie Liang
Yin Lin
Ishan Vashishtha
Jia Zhang
Carnegie Mellon University
Silicon Valley
Moffett Field, CA, USA
lucas.colucci@sv.cmu.edu
prachi.doshi@sv.cmu.edu
kun.lin.lee@sv.cmu.edu
jiajie.liang@sv.cmu.edu
yinlin@cmu.edu
ishan.vashishtha@west.cmu.edu
jia.zhang@sv.cmu.edu

Alvin Jude [+]
Ericsson Research
San Jose, CA, USA
alvinjude@acm.org

[+] Corresponding Author

CHI'16 Extended Abstracts, May 07-12, 2016, San Jose, CA, USA
ACM 978-1-4503-4082-3/16/05.
http://dx.doi.org/10.1145/2851581.2892362

Abstract
Recommender systems such as those used in e-commerce or Video-On-Demand systems generally show users a list of "similar items." Many algorithms exist to calculate item-item similarity and we wished to evaluate how users perceive these numerically expressed similarity. In our experiment, we performed a user study with four similarity algorithms to evaluate perceived correctness in item-item similarity as it relates to movies. We implemented three algorithms: collaborative filtering with Pearson, collaborative filtering with cosine, and content-filtering with TF-IDF. A pre-generated similarity list from TheMovieDB.org (TMDb) was used as the baseline. Our experiment showed that TMDb has the highest perceived similarity, followed by cosine and TF-IDF, while Pearson was practically unusable for users. A by-product of our experiment was a set of similar movie pairs, which we intend to use for offline evaluation.

Author Keywords
Recommender Systems; Item-item similarity; Algorithm Evaluation.

ACM Classification Keywords
H.3.3 Information Search and Retrieval: Relevance feedback.

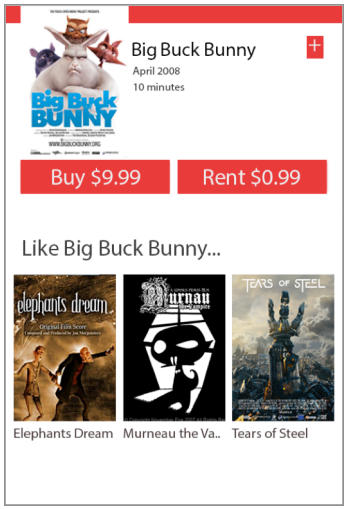

Figure 1 An example of item-item similarity that could be used in practice, with a UI inspired by the Google Play Movies and TV. When a user selects a movie (in this case the movie Big Buck Bunny from 2008), movies that are adjudged to be similar are shown below.

Introduction

Research in the area of recommender systems has had great strides in recent years, especially since the Netflix Prize in 2006. One of the more common approaches to recommendations uses Collaborative Filtering (CF), where the users' opinion of an item is considered as input to the recommender. Two most common forms of CF are user-user CF [6] and item-item CF. The intuition behind user-user CF specifically is that users of a similar taste prefer similar items. It is therefore possible to enumerate *user-A*'s taste based on her consumption history, then to find another user, say *user-B* with similar consumption, then recommend items that were consumed by *user-B* to *user-A*.

User-user CF is commonly used to recommend items to a user. Recommender systems are also known to implement item-item recommendations [7], which can be done with item-item CF or through content-based filtering. In item-item CF, two items *item-X* and *item-Y* are adjudged to be similar if they are frequently consumed together, or by the same users. In content-based filtering, items are adjudged to be similar if they share the same genres, directors, or writers. Item-item recommendations in general are known for being less resource intensive and highly scalable [5]. In addition, unlike CF approach, it is possible to recommend items to a new or anonymous user whose consumption behavior is unknown. This is often seen on websites such as Amazon, Netflix, or Google Play, usually under the heading "Items similar to this…" or "People who liked X also liked…" followed by the list of similar items.

Algorithms that perform user-user recommendations are easier to validate, as there is a set of ground-truth in the form of user ratings, thus allowing for cross-validation. In contrast, item-item algorithms are harder to evaluate, as it is difficult to know if two items are in fact similar. No known dataset has a pair of items and it's perceived similarity for cross-validation. From a machine-learning perspective, we think of user-user recommendations as a supervised learning problem, and item-item recommendations as an unsupervised learning problem, or clustering problem. So the question is: which algorithms' notion of similarity best matches the users' perception?

In this paper we evaluated users' perception of similarity against item-item similarity algorithms with an experiment, and we showed that users disagree with the algorithmic notion of similarity almost half the time. During this process, we also collected labels from the users regarding their perception of similarity, through which we intend to change the problem from an unsupervised learning problem into a supervised learning problem.

Related Works

Von Ahn and Dabbish released a game-with-a-purpose called ESP [2] in 2004. Here, two anonymous users apply labels to an image, and both players are awarded a point each if there is consensus on the labels. The authors proposed the game be used to improve image recognition using the supplied labels. Google later licensed the game for use in their Image Labeller, likely for this very purpose. ESP was the primary inspiration for our project, but we differ in a few ways. Our system was not intended to be a two-player game for fun. Instead we explicitly wanted to evaluate existing algorithms and to collect labels for offline evaluation in the future. In this sense, our work was closer in philosophy to the Open Mind Initiative [1].

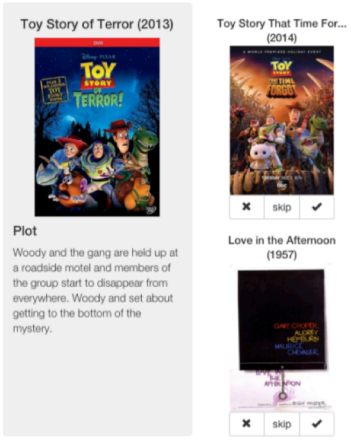

Figure 2 A section of the evaluation front-end, where the selected movie is on the left, while the movies judged to be similar are placed in the two rows on the right. Movies can be judged to be similar, not similar, or the users can skip if they are unsure. Details provided to the users were sparse to encourage them to use preexisting knowledge to infer similarity, rather than making a decision based on contextual cues such as plot line.

GroupLens Research from the Department of Computer Science and Engineering at the University of Minnesota maintains a publicly accessible website MovieLens.org, where users can submit movie ratings between 1 to 5 stars. The dataset collected is made public and often used in recommender systems research. Unlike the ESP game, there is no known entertainment value to using MovieLens. It does suffer from under-contribution, but it has been shown that users tend to contribute more when they are aware that their contributions help others [3].

The researchers at GroupLens have also released LensKit [4], a set of libraries that implements algorithms for recommender systems. We used the item-item collaborative filtering algorithms in our experiments with cosine and Pearson methods from LensKit. Amazon.com reported good performance and high quality recommendations with the cosine method. However, we are unaware of any user study that shows the exact definition of "high quality."

Data set, Similarities, and Algorithms

Our database was seeded with the MovieLens 20M dataset, which contained approximately 27,000 movies. This dataset was augmented with movie metadata from The Open Movie Database (OMDB)[1] to add relevant information such as cast, credits, posters, and plot details. We used additional metadata from The Movie Database (TMDb)[2] for movie revenue, as well as posters when it was not obtainable from OMDB.

[1] http://omdbapi.com

[2] https://www.themoviedb.org

For our evaluation, we built four sets of movie-movie similarity models. Three sets were generated algorithmically and 1 was obtained from TheMovieDB.org. Of the three generated sets, two used Collaborative Filtering and another used Content Filtering techniques. Our three algorithms compared all movies against all movies for similarity (27k X 27k). But only the top 20 similar movies were later in the database to reduce database size. Implementation and details of the algorithms are provided below.

The Movie Database

TMDb has a list of similar movies on their website, which is also obtainable through their API. We considered this a "gold standard" in our evaluation and representative of movie-movie similarity in existing systems. TMDb's similarity algorithm was unknown to us and therefore treated as a black box. The similar items in the list did not contain any numeric value indicating the degree of similarity. We are therefore unaware if the list is sorted.

Item-item Collaborative Filtering

We presented the intuition behind this method in brief earlier: item-A is considered similar to item-B because users who liked item-A also liked item-B. Collaborative filtering therefore requires user input to measure the distance between items. The MovieLens 20M dataset contains about 20 million ratings over 27,000 movies.

The above describes the general intuition behind item-item CF. However specific methods are required to actually calculate the similarities. For which we used two different algorithms: Pearson Correlation and Cosine Similarity, both from the LensKit framework.

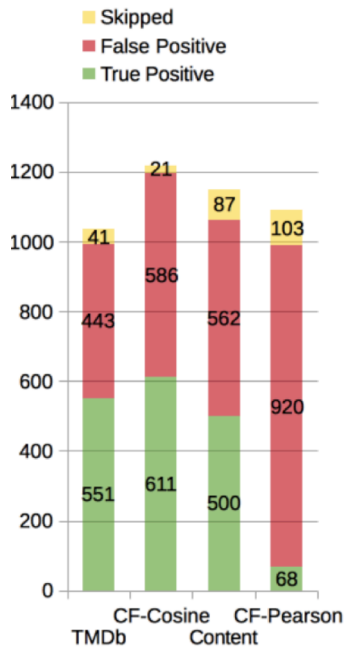

Figure 3 Absolute values of user evaluation per model.

	Total	TP+FP
TMDb	1035	994
CF-Cosine	1218	1197
Content	1149	1062
CF-Pearson	1091	988
Sum	4493	4241

Table 1 Total evaluations (TP + FP + Skipped) and condition positive (TP +FP) for each model.

Content Filtering

Content filtering uses item metadata to infer similarity. Some types of metadata available from movies include the name, production year, plot, cast, crew and genre. We examine the intuition behind content filtering with this example: if movie-A and movie-B are both romantic comedies, then they must be somewhat similar. If movie-C is also a romantic comedy and is written by the same writer as that of movie-A then movie-C is more similar to movie-A than movie-B. The benefit of a pure content filtering approach is that it requires no prior user input. This could especially be useful in a new system where little or no user-input exists, which would make any form of collaborative filtering impossible.

Various approaches exist for content filtering. We specifically used TD-IDF [8] with the bag-of-words approach [9]. Our implementation was done with Lucene[3], where we used dot product as a similarity measure and weighted sum as aggregation function.

We tried a few ways to tune this algorithm, and decided on the following based on a naïve eyeball test of the generated results. The metadata was weighted as follows: movie title 0.25, genres 0.2, cast 0.2 writer 0.15, director 0.1, and plot 0.1. Proper nouns were removed from the plot using the Stanford Named Entity Recognizer library[4], as they affected results.

Methodology

The experiment consisted of 14 graduate students (F=4) with a mean age of 25, and in a within-groups experiment design. The experiment was conducted in a 40-minute block, but participants were allowed to start and finish whenever they wanted. Participants were given a pseudorandom username and password for the system so that no personally identifiable information was collected. Brief training was provided to expose participants to the features of the system, after which they performed as many tasks as they liked. A task consisted of selecting a movie, and then evaluating the displayed movies for similarity. Each selected movie (and therefore each task) allowed up to 20 evaluations. After completing the tasks, participants filled up an exit survey consisting of 8 questions on a five-point Likert scale and one free-form question requesting for any feedback they had about the experiment.

User Interface

The participants could select a movie to evaluate from either one of the popular movies category, the random movies category, or by searching for the movie using the search box. The popular movies category was static, and sorted by highest revenue with data obtained from TMDb. The random movies category showed a list of 12 randomly selected movies from a pool of the 200 most rated movies in the MovieLens dataset. We initially attempted to show a truly random set, but it appeared to frequently show movies that were not at all familiar to the users.

After selecting a movie, participants were presented with a list of eight movies to evaluate. They then decided if the movie is similar, not similar, or to skip in case they were not familiar with the movie. After any input from the user, the evaluated movie is visibly removed from the list and replaced with a new option if one exists.

[3] http://lucene.apache.org

[4] http://nlp.stanford.edu/software/CRF-NER.shtml

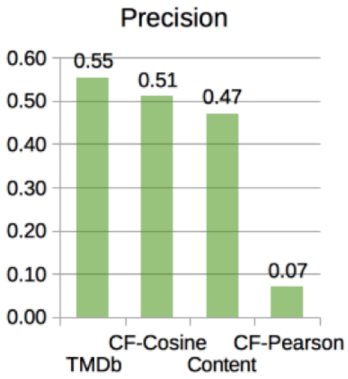

Figure 4 Precision per model, where higher is better. Precision is defined as TP / (TP + FP).

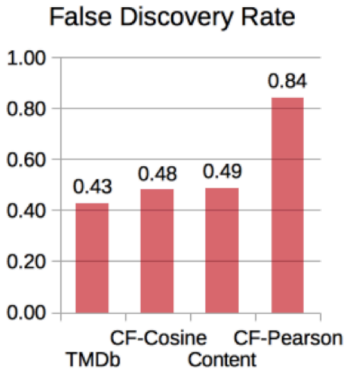

Figure 5 FDR per model, where lower is better. FDR is defined as FP / (TP + FP + Skip).

Model Fairness and Balancing

We balanced the pairwise evaluations to ensure that each model was evaluated an equal number of times per task. Each model generated 20 candidates, which were selected for inclusion in the front-end in real time depending on user input. This was done to give equal opportunity to all models. Participants were not made aware of the fact that there were different models.

Model Evaluation

An evaluation consists of a participants input for a movie-movie pair. It is possible for a pair to be generated by more than one model, in which case the evaluation is added to all models that produced it.

Results

The experiment produced a total of 3802 responses from the users. As some pairs were generated by multiple models, this resulted in a total of 4493 data points (Table 1), 6% of which were "skip". A total of 1416 distinct movies were evaluated. Participants performed an average of 15.7 tasks (stdev=5.3), 272 evaluations (stdev=97) and supplied 321 data points (stdev=93). On average, it took 2.8 minutes per task.

Algorithm Evaluation

We define True Positives (TP) to be any pair produced by a model that the user believe is similar, and False Positive (FP) as any pair produced by a model and the user believes to be dissimilar. Models were evaluated by precision and false discovery rate (FDR). Precision was defined as TP/(TP+FP). False Discovery Rate on the other hand was calculated by including skipped items: FDR=TP/(TP+FP+Skip). The intuition behind FDR is to increase the possibility of a type-1 error in favor of higher statistical power. This is contextualized by the smaller difference between CF-Cosine and content in FDR (Figure 5) than precision (Figure 4).

We found that TMDb has the highest precision, while Pearson had the lowest. A Chi-squared test showed that this difference was significant ($p>0.001$). A Chi-squared post-hoc analysis with FDR adjustment showed a significant difference between all pairs except *CF-Cosine* and *Content*. These results validated our assumption of TMDb as the baseline for comparison. It also highlighted the fact that the *CF-Pearson* algorithm was especially bad for item-item similarity.

Perception of Similarity Among Users

The results showed that 62% of the evaluation pairs had complete consensus, in that no participant disagreed on their similarity. In the rest, there was at least one who disagreed. This showed us that perceived similarity is definitely not universal; an item that is perceived to be similar to one person may not be similar to another. However, there is a good chance that most people would agree in most cases: 62% absolute agreement shows a high number. In the feedback, one participant noted that they understood why some people might think that a recent Superman movie is "similar" to the older ones, but they didn't agree with it. This showed us that different people might have different criteria in defining similarity.

Algorithmic Selection Bias

A Chi-squared test showed unequal representation from each model, despite our attempts ($p<0.01$). The differences were however not particularly large (23%-28%). It is possible this bias could be reduced if the models were balanced globally across all users and tasks, instead of per task.

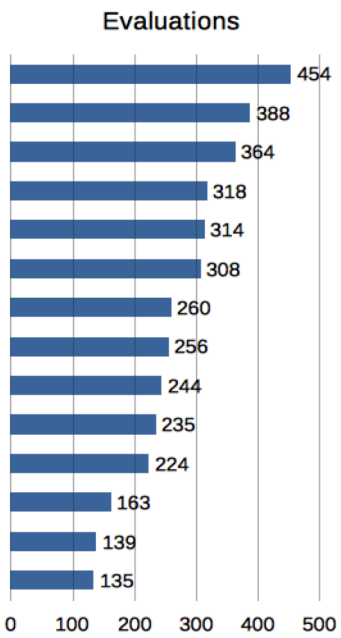

Figure 6 Each horizontal bar represents one participant and the number of evaluations supplied. An evaluation consists of one movie-movie pair and perception of similarity. Total evaluations: 3802, mean: 271.6, stdev: 93.4, median: 258.

Model Overlap

We investigated if there were overlaps between the top-20 similar movies generated by the models. It appeared that only 9% of the pairs generated by one model was also generated by another model. We hypothesized that a large number of overlapping items could lead to an ensemble algorithm, and will be further investigated, possibly by increasing the number of similar movies generated per model from 20 to 30.

User Behavior

More than half the tasks completed with the participant submitting all 20 evaluations. We believe this could mean it is possible for future work to request more than 20 evaluations per task. The search feature was used a total of 80 times, which represents 36% of all tasks. No search term was used more than twice, and most search terms were distinct.

User Feedback

After the tasks, participants were asked to complete an exit survey, and all 14 participants obliged. The results showed a positive experience towards the user interface of the system. We noted that participants seemed to especially like the fact that movies that were rated were immediately replaced with new ones. One participant noted that they discovered new movies, and added it to their watch list. This could be relevant in providing future users motivation to contribute.

Participants apparently prefer more information such as metadata to help them in deciding which movies are similar. This was in fact debated during the design stages, and we decided to give very little information about the movies; only a short version of the plot was provided. We hypothesized that the plot could steer the participants in making a biased decision, rather than based on their preexisting knowledge of both movies.

Another interesting finding was that a majority of the participants claim to prefer the search function over the *random* set, however the random set was actually used more. We believe this is because users exhaust the random list, before moving on to the search feature.

Conclusions and Future Work

In this experiment, we showed that it is possible to evaluate different item-item similarity algorithms or models with a user study. Our study has managed to show which algorithms are more useful than others, and there is a high enough consensus among participants to believe that these results are valid. We showed that participants disagreed with approximately 43% of the algorithmically generated similar items, which shows a clear reason why human-driven research is required in this field. We need to better understand why people perceive items to be similar. After which we can focus on algorithms that model this behavior.

We propose that a public website that collects user's opinion of similarity, like that used in the experiment, can (and needs to) exist. We deduce that people will be motivated to contribute evaluations to help others, as well as to discover movies for themselves. Assuming an average of 2.8 minutes per task, it would be possible to collect similarity evaluations for all 27000 movies against a list of 20 similar movies in 1260 man-hours.

Until then, the relatively small amount of data collected from this experiment could be useful as labels or ground-truth to improve and evaluate algorithms offline. This will be the next step of our research.

References

1. David G. Stork. 2000. Open data collection for training intelligent software in the open mind initiative. In *Proceedings of the Engineering Intelligent Systems (EIS'00)*.
2. Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 319-326. DOI=http://dx.doi.org/10.1145/985692.985733
3. Yan Chen Harper, F. Maxwell, Joseph Konstan, and Sherry Xin Li. 2010. Social comparisons and contributions to online communities: A field experiment on movielens. *The American economic review*: 1358-1398.
4. Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan, and John T. Riedl. 2011. Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit. In *Proceedings of the fifth ACM conference on Recommender systems* (RecSys '11). ACM, New York, NY, USA, 133-140. DOI=http://dx.doi.org/10.1145/2043932.2043958
5. Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE* 7.1: 76-80.
6. David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (December 1992), 61-70. DOI=10.1145/138859.138867 http://doi.acm.org/10.1145/138859.138867
7. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (WWW '01). ACM, New York, NY, USA, 285-295. DOI=http://dx.doi.org/10.1145/371920.372071
8. Michael J. Greg, and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer Berlin Heidelberg, 325-341.
9. Alexandrin Popescul, David M. Pennock, and Steve Lawrence. 2001. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence* (UAI'01), Jack Breese and Daphne Koller (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 437-444.