2013 Hawaii University International Conferences
Education & Technology
Math & Engineering Technology
June 10th to June 12th
Ala Moana Hotel, Honolulu, Hawaii

The Evaluation of Similarity Metrics in Collaborative Filtering Recommenders

Dr. Sergey Morozov

Dr. Xiaohui Zhong

University of Detroit Mercy, Detroit MI

The Evaluation of Similarity Metrics in Collaborative Filtering Recommenders

Serhiy Morozov and Xiaohui Zhong
Department of Mathematics, Computer Science, and Software Engineering
University of Detroit Mercy, Detroit MI, USA
{morozose, zhongk}@udmercy.edu

***Abstract*—Collaborative filtering recommenders predict user opinions regardless of item content. Memory-based implementations generate recommendations on demand, without any pre-computation. Even though such algorithms are slow, they are known for their superior personalization. We study collaborative filtering recommendations on the Netflix dataset. It is essentially a set of (user_id, item_id, rating) tuples that fit perfectly into a relational database model, with standard data manipulation operations. The simple structure of our system allows consistent and repeatable measurements. Additionally, it natively supports multi-threaded processing, which helps address the inherent performance drawbacks of our approach. We evaluate multiple similarity measures in a traditional collaborative filtering process. We also consider combinations of complementary measures, especially in edge cases when one of them falls short, e.g., a user with uniform ratings. We examine prediction accuracy, classification accuracy, confusion statistics, and actual/predicted distribution compatibility to find the best way to quantify vector similarity.**

***Index Terms*—Recommender systems, Collaborative filtering, Neighbor selection, Prediction aggregation, Netflix dataset**

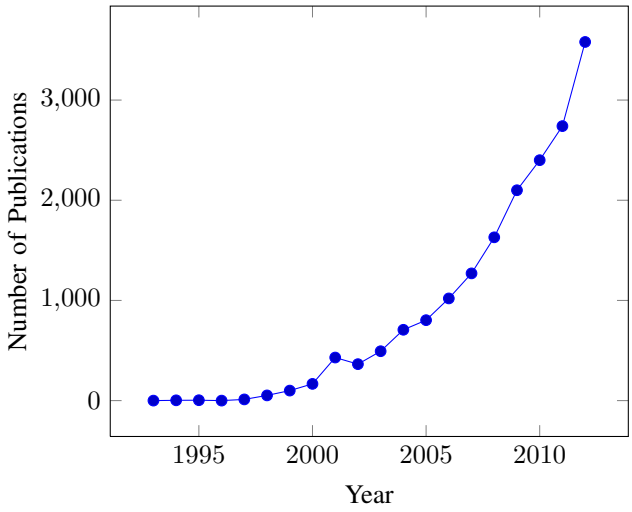

Fig. 1. Recommender System Popularity

I. Introduction

Modern technology has made it easy to collect and store large amounts of data. Few people manage to have an empty email inbox and almost everyone carries a smartphone with thousands of songs. Manually inspecting every item is time-consuming, since most of them are irrelevant. Recommender systems automate this process by suggesting a subset of items, depending on user preferences. For example, a recommender system can go through your inbox and select only the most urgent emails. Unlike spam filters that apply the same logic to everyone, recommender systems offer personalization [1]. This ability has been gaining popularity as an elegant way to explore large volumes of data. Figure I plots the growing number of publications that mention the term "Recommender System" since the mid 1990s.

Such rapid growth of interest may be due to a large commercial potential. A well-placed recommendation could result in a purchase that would not otherwise happen, thus increasing the retailer's revenue [1], [9], [26]. In fact, many retailers have embraced recommender algorithms on the Internet, where extensive archives of user behavior are common. Many successful applications recommend movies, videos, music, books, and articles [2], [3], [24]. However, the pressure to make quality recommendations is high, because a poor recommendation could prompt the user to abandon the service. As a result, there is a high demand for improvement in this field.

All recommender systems mimic human decision process. For example, to choose a movie that they will most likely enjoy, people often look for new releases with their favorite actors or by their favorite directors. In fact, some movies market themselves by explicitly advertising these facts. Alternatively, people may ask friends and family if they have seen any good movies recently. These two approaches are know as content-based and collaborative filtering recommendations, depending on whether the insight came from similarities in content or opinions [1], [3], [21], [22]. Collaborative filtering recommenders are more accurate and adaptable than content-based methods [1], [7], [11], [13], [20], [21]. Content affects opinions too, but users are more predictable from their social circle. In other words, we can learn about a person by looking at their friends. We focus on purely collaborative filtering recommenders.

Collaborative filtering systems differ by their learning approach. The model- and memory-based classifications resemble the lazy and eager commitment in machine learning [1], [4]. In both cases, the system learns the patterns hidden in the data, but it has more time to do so before the query than during the query, while the user is

waiting. Memory-based approaches process the entire dataset on demand, which makes them slow [14]. However, parallel processing and intermediate result caching may improve performance. We implement memory-based recommenders because they are better suited for rapid prototyping and partial evaluation.

II. Data Model

To evaluate our recommender system, we split the entire dataset into two parts, training and testing set. Each algorithm learns the training set and guesses the test set. Since the actual ratings in the test set are known, we can evaluate the quality of the guesses by comparing the actual ratings and their predictions. This method is a version of cross-validation, a popular method of offline evaluation [4], [7], [10], [12], [24]. This means that our results may be easily compared to existing work on this dataset.

The Netflix dataset comes with some of the ratings identified for evaluation purposes, i.e., the quiz subset. This dataset is a popular benchmark of recommender system research [8], [12], [25]. It was published by the online DVD rental company (Netflix) in October 2006 and it contains over 100 million ratings from over 480 thousand randomly selected customers on over 17 thousand movies. Because it is unlikely that every user has rated every item, the dataset is very sparse. In fact, it represents only 1.1% of all possible ratings. However, there is enough data to predict the 1.4 million ratings in the quiz dataset.

The following is a model of our dataset in the Z notation. The system maintains a set of users U, items I, and possible ratings R. We model all three as sets of natural numbers. The dataset is a collection of tuples (u, i, r). For example, (229, 386, 4) represents user 229 who has rated the item 386 as 4 stars. The user, item, and rating in question are called active user, active item, and active rating. Note that the user can rate each item at most once, hence the functional relationship between the user-item pair and the rating. Also, the dataset does not contain every possible user-item combination, hence the relationship is partial.

$$U, I, R : \mathbb{P}\,\mathbb{N}$$
$$DATASET : (U \times I) \nrightarrow R$$

If the dataset contains at most one rating per user-item, we can not track change of preferences over time. Even though such assumption is not realistic [1], [24], it lets us model the dataset as a collection of user or item vectors [1], [5], [11], [19]. In this context, a vector is a partial function that maps dimensions to ratings. The only difference between a user vector and an item vector is the identity, but not the type, of dimensions. User vectors have item dimensions and item vectors have user dimensions. Since both users and items are modeled as a natural numbers, the two vectors have the same structure. Therefore, the same algorithm can compute user- and item-based recommendations.

$$VECTOR : \mathbb{N} \nrightarrow R$$
$$DATASET : \mathbb{N} \nrightarrow VECTOR$$

Collaborative filters assume that existing rating similarity indicates future rating similarity. For example, if a person enjoyed one item, he/she will enjoy a similar item as well. Likewise, if one person enjoyed an item, a similar person will enjoy that item too [24]. Therefore, the effectiveness of a recommendation depends on our ability to identify similar vectors, called neighbors [4], [8], [16]. Most metrics quantify similarity based on the vector overlap [2], [5], [8], [14], [20], [22], [24], [26]. Even though vectors that have nothing in common may still be useful, we require a minimum overlap between the active vector and its neighbors. Given an active vector A and its neighbor N, only a portion of each vector participates in the similarity computation. The a and n vectors contain only the dimensions that are common for both of the original vectors. The set of common dimensions constitutes the overlap.

$$A, N : VECTOR$$
$$a, n : VECTOR$$
$$overlap : \mathbb{P}\,\mathbb{N}$$
$$overlap\_size : \mathbb{N}$$
$$\hline$$
$$overlap = \{\mathrm{dom}\,A \cap \mathrm{dom}\,N\}$$
$$overlap\_size = \#\{\mathrm{dom}\,A \cap \mathrm{dom}\,N\}$$
$$a = \{d : \mathbb{N} \mid d \in \mathrm{dom}\,A \wedge d \in \mathrm{dom}\,N \bullet d \mapsto A(d)\}$$
$$n = \{d : \mathbb{N} \mid d \in \mathrm{dom}\,N \wedge d \in \mathrm{dom}\,N \bullet d \mapsto N(d)\}$$

III. Collaborative Filtering Process

Collaborative filtering process consists of neighborhood formation, similarity computation, neighbor selection, and rating aggregation phases. The active vector does not have a rating on the active dimension. Its neighborhood consists of the vectors that do have a rating on the active dimension. Each neighbor exhibits a measurable similarity to the active vector. The best neighbors have a large overlap with the active vector and their ratings in matching dimensions agree. The recommendation is the consensus of the neighborhood on the active dimension, where some neighbors have more influence than others. Therefore, the quality of the recommendation depends on the number and similarity of the neighbors.

A. Neighborhood Formation

The initial neighborhood consists of the vectors that have a rating on the active dimension. The relevant vectors represent a small portion of the dataset, which is easier to process. The following schema models the minimum requirement on the neighborhood G. Given the active vector id, we can look it up in the dataset. Note that the active vector and all of its neighbors are complete vectors, even though only a portion of each vector participates in the similarity computation.

$$
\begin{array}{|l}
G : \mathbb{N} \nrightarrow \mathit{VECTOR} \\
A : \mathit{VECTOR} \\
\mathit{vector\_id}? : \mathbb{N} \\
\mathit{dimension\_id}? : \mathbb{N} \\
\hline
A = \mathit{DATASET}(\mathit{vector\_id}?) \\
G = \{j : \mathbb{N}, N : \mathit{VECTOR} \mid \\
j \mapsto N \in \mathit{DATASET} \land \mathit{dimension\_id}? \in \operatorname{dom} N\}
\end{array}
$$

One neighbor may be relevant for multiple consecutive recommendations, so reusing its similarity may improve performance. We compute all vector similarities once and then use some of them to produce different recommendations. The performance advantage depends on the recommendations. A long list likely contains multiple recommendations for the same vector, e.g., multiple recommendations for the same movie. We use this method for our item-based recommendations that have an average of 83.150 predictions for each item.

B. Similarity Computation

Many similarity metics have a lot in common. To reduce redundant work, we compute prime statistics and then combine them into various similarities. The following formulas define the dot product, norm active, norm neighbor, mean active, and mean neighbor statistics on the overlap of the active vector a and its neighbor n. Because the overlap changes from neighbor to neighbor, we must compute the active vector length and its average rating for each neighbor.

$$a \cdot n = \sum_{i \in overlap} a_i n_i$$

$$\mid a \mid = \sqrt{\sum_{i \in overlap} a_i^2}$$

$$\mid n \mid = \sqrt{\sum_{i \in overlap} n_i^2}$$

$$\bar{a} = \frac{\sum_{i \in overlap} a_i}{overlap\_size}$$

$$\bar{n} = \frac{\sum_{i \in overlap} n_i}{overlap\_size}$$

Cosine similarity measures the cosine of the angle between two vectors. If the vectors point in the same direction, their cosine similarity is 1. This similarity considers the alignment of vectors to be more important than the agreement of their ratings. It uses the dot product and the two vector norms.

$$cosine(a,n) = \frac{a \cdot n}{\mid a \mid\mid n \mid}$$

If the overlap size is 1, the cosine similarity is always 1. This happens because the single rating defines the magnitude of the vector and the same rating participates in the dot product. For example, Alice has rated "Titanic" as 5 and Bob has rated the same movie as 2. The dot product is 10 and the product of the two vector norms is also 10. This means that neighbors with a single rating overlap will have a perfect similarity, even if their ratings are different. Therefore, cosine similarity with single rating overlap is not trustworthy.

Covariance is the measure of how much two random variables change in relation to each other. If the two sets of values rise and fall together, their covariance is positive. If they change in the opposite directions, the covariance is negative. Covariance is the only similarity that is not bound to a 0..1 range. We derive a formula that relies on the dot product, vector norms, and the overlap size. Note that Covariance is always 0 for any two vectors with a two-rating overlap, regardless of the actual ratings.

$$covariance(a,n) = \frac{\sum (a_i - \bar{a})(n_i - \bar{n})}{overlap\_size}$$

$$covariance(a,n) = \frac{a \cdot n}{overlap\_size} - (\bar{a}\bar{n})$$

Pearson's correlation quantifies the strength and direction of the linear relationship between two random variables. It is similar to cosine, except it normalizes the ratings of both vectors relative to their mean. Because this similarity considers the difference in rating scales, it is typically better than cosine similarity, which uses raw ratings [4], [8], [15]. We derive a formula that relies on covariance, vector norms, their means, and the overlap size. Note that because Pearson and Covariance similarities are related, Pearson similarity is also 0 for any two vectors with a two-rating overlap. Also note that if the vector overlap contains uniform ratings, the Pearson similarity is undefined.

$$pearson(a,n) = \frac{\sum (a_i - \bar{a})(n_i - \bar{n})}{\sqrt{\sum (a_i - \bar{a})^2}\sqrt{\sum (n_i - \bar{n})^2}}$$

$$\sum_{i \in overlap} (a_i - \bar{a})^2 = overlap\_size(\frac{\mid a \mid^2}{overlap\_size} - \bar{a}^2)$$

$$pearson(a,n) = \frac{covariance(a,n)}{\sqrt{\frac{|a|^2}{overlap\_size} - \bar{a}^2}\sqrt{\frac{|n|^2}{overlap\_size} - \bar{n}^2}}$$

Euclidean similarity is the opposite of the Euclidean distance. It is proportional to the length of a line drawn between the edges of the two vectors. This similarity metric favors distance over direction and makes no adjustments for the actual ratings. We derive Euclidean distance from the dot product and the norms of the two vectors.

$$euclidean\_distance(a,n) = \sqrt{\mid a \mid^2 -2(a \cdot n) + \mid n \mid^2}$$

In our dataset, the maximum difference between two ratings on a 1..5 scale is 4. We divide the Euclidean distance by 4 to make it range from 0 to 1. We also divide it by the overlap size to derive an average distance per shared dimension [7], [22]. We subtract the Euclidean distance from 1 to get Euclidean similarity. This way the similarity is 1 when two vectors are

identical (distance = 0) and 0 when they are different (distance = 1).

$$euclidean(a,n) = 1 - \frac{euclidean\_distance(a,n)}{\sqrt{overlap} * 4}$$

C. Neighborhood Selection

it is often necessary to prune the neighborhood to improve performance and reduce over-fitting. There are three main approaches to neighbor selection: top N most similar neighbors, everyone with a similarity above a certain threshold, and a random sample of neighbors [4], [9], [11], [14], [15]. In any approach, the size of the neighborhood has a tremendous effect on the prediction [5], [6]. For example, setting the similarity threshold too high could limit the recommendation coverage. Likewise, the other approaches may include dissimilar vectors, thus skewing the recommendation in the wrong direction. We select all neighbors with a positive similarity.

D. Prediction Aggregation

The neighborhood forms the recommendation by aggregating the neighbors' ratings. The aggregation could be a simple arithmetic mean [24]. It is a fast, but inaccurate. We use it as a benchmark of performance and usefulness. A better aggregation would be a weighted average of neighbors' ratings weighted by their similarities.

$$regular(a,i) = \frac{\sum_{n_j \in G} n_j(i) * sim(a,n_j)}{\sum_{n_j \in G} sim(a,n_j)}$$

IV. Process Modifications

A. Similarity Combination

When a pure collaborative filtering approach cannot be improved any further, it may be beneficial to combine multiple methods to counter-balance their drawbacks [3], [5], [6], [20]. The Combo similarity metric is a linear combination of the Pearson and Euclidean similarities. For this work, we used an arithmetic average of the two. These two metrics offer different perspectives of the vector similarity. Pearson measures the trend similarity and Euclidean captures the rating similarity relative to the overlap size. Typically, Pearson is better, but it may miss some cases. Consider the following two vectors and their similarities. The combination of these two metrics is more likely to capture the inherent vector similarity.

$$a =< 4,5,4,4,4,4,4,4,4,4,4 >$$

$$n =< 5,4,5,4,4,4,4,4,4,4,4 >$$

$$pearson(a,n) = -0.15$$

$$euclidean(a,n) = 0.8694$$

B. Recommendation Combination

Depending on the dataset, more insight may come from the user-based approach or the item-based one. For example, if predicting a rating for a bestseller movie, the item-based ratings are more likely to have a consensus [18]. However, user-based approach is likely to be more accurate for new releases that are not yet popular. Therefore, relying on one vector orientation often ignores valuable information. Combining user- and item-based approaches can produce more accurate recommendations [11], [15], [23], [26].

V. Evaluation Metrics

A. Accuracy

Predictive accuracy measures how close a typical prediction is to an actual rating. High prediction accuracy does not necessarily mean high user satisfaction, but it is a pretty good indicator of the overall system value [10], [17], [27]. Despite the inherent limitations, prediction accuracy is the most popular evaluation metric with much of the research attempting to improve their metric on a particular dataset [1], [4], [10], [24]. A number of predictive accuracy metrics aggregate the difference between the set of n predictions P and the set of actual ratings R. For example, Mean Absolute Error (MAE) averages the absolute error of the individual predictions.

$$MAE = \frac{\sum_{i=1}^{n} \mid P_i - R_i \mid}{n}$$

MAE is essentially the Manhattan distance between a set of predictions and the actual ratings. A small MAE means that a typical prediction is close to the actual rating. This could mean half perfect predictions and half terrible predictions, because all errors have the same influence.

The Root Mean Squared Error (RMSE) is a variation of MAE that amplifies the influence of the large errors. RMSE is essentially the Euclidean distance between a set of predictions and the actual ratings. Smaller RMSE means that most of the errors are small.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (P_i - R_i)^2}{n}}$$

Since many accuracy metrics mirror vector similarity measures, we also compute the correlation between the predictions and actual ratings. Correlation measures the extent of a linear relationship between predictions and the actual ratings. If such a relationship exists, we may improve recommendation accuracy by adjusting the predictions up or down. Correlation is not sensitive to the amount of predictions and it considers the mean of each distribution. Therefore two sets of predictions with the same MAE/RMSE could have different correlations.

$$CORR = \frac{\sum (P_i - \bar{P})(R_i - \bar{R})}{\sqrt{\sum (P_i - \bar{P})^2} \sqrt{\sum (R_i - \bar{R})^2}}$$

A measure of typical error is a convenient, but potentially misleading metric because it ignores the direction of the error. From the user satisfaction perspective, the direction of an error

may be more important than its size. For example, a user does not care if the recommendation is 0.4 or 0.3 stars off, he/she just wants to know if the item is good enough [1], [4], [10]. Therefore, the value of a recommendation depends on choosing the correct side of the user's tolerance threshold.

B. Classification

Classification accuracy measures how good a recommender system is at correctly classifying an item as interesting. The distance from the tolerance threshold does not matter, but different people have difference tolerances. In other words, we cannot assume that a rating above two stars is good enough for everyone [2], [5], [12]. A more reasonable threshold would be the vector average, i.e., a rating is positive if it is greater than or equal to the vector average. The tolerance threshold produces a binary scale, either the person likes an item (positive) or not (negative). Likewise, the recommendation could be correct (true) or incorrect (false). The following metrics tally the number of occurrences of each prediction p and actual rating r.

- True Positive (TP): p = positive, r = positive
- False Negative (FN): p = negative, r = positive
- False Positive (FP): p = positive, r = negative
- True Negative (TN): p = negative, r = negative

Precision and recall summarize these numbers into more intuitive metrics. Precision is the fraction of all positive ratings that were correctly classified as such. It measures how good the system is at recognizing positive recommendations. For instance, 60% precision means that the user can expect to actually enjoy three of every five recommendations. Recall is the fraction of all positive recommendations that are actually positive. It measures how good the system is at finding positive recommendations [4], [10], [24]. For example, 80% recall means that the system can correctly guess four out of every five of your favorite movies.

$$PRECISION = TP/(TP + FP)$$

$$RECALL = TP/(TP + FN)$$

Even though precision and recall are separate measures, they are related. Typically, high precision means low recall and high recall means low precision. One popular way to combine precision and recall into a single metric is to compute the F-measure, which is a harmonic mean of the two.

$$F - MEASURE = \frac{2 * (PRECISION * RECALL)}{PRECISION + RECALL}$$

C. Distribution

Histograms of the actual and predicted ratings describe their respective distributions. The chi-squared metric, commonly used to test the goodness of fit, can quantify distribution similarity. Because it requires nominal values we round each prediction to the nearest integer to obtain 5 distinct values. The chi-squared statistic is a normalized sum of squared deviations between observed (P) and expected (R) frequencies.

$$X^2 = \sum_{i=1}^{5} \frac{(FREQUENCY(P_i) - FREQUENCY(R_i))^2}{FREQUENCY(R_i)}$$

To determine if a particular distribution is sufficiently similar, we compare the statistic against the significance threshold defined by the degrees of freedom. We do not anticipate that any set of predictions will be sufficiently similar to the actual rating distribution. Therefore, we use the this statistic to rank our recommenders from best to worst.

D. Confusion

A confusion matrix is a table that visualizes the mistakes of a recommender algorithm. Each column r is labeled with an actual value. Each row p is labeled with a predicted value. The cells contain the count of predictions that were actually r, but classified as p. As the name suggest, this table makes it easy to see if a recommender confuses two or more ratings.

Comparing multiple confusion matrices from different recommenders is difficult, so we compare a cross section of these tables according to the actual rating. This gives us five tables with confusion statistics for each recommender. To figure out which recommender is best at guessing a particular rating, we measure the RMSE and MAE of their predictions. This report compares the recommenders on their ability to predict a particular rating. It can also confirm that some recommenders are consistently better than others.

VI. Netflix Dataset Case Study

To improve the reliability of our results, we remove the ratings in the quiz subset from the training set. We use a standard neighborhood selection approach where any vector with a rating on the active dimension counts as a neighbor. There is no neighborhood restriction, i.e., all neighbors with a positive similarity contribute to the prediction. Because Cosine similarity is 1 when the overlap is 1 and Pearson/Covariance similarity is 0 when overlap is 2, we require at least a three-rating overlap. In cases when the overlap contains uniform ratings and the similarity is undefined, the default recommendation is the mean active rating. The following sections present the results of our case study.

A. Accuracy

Figure 2 shows the recommender accuracy metrics. Covariance and Pearson are the best across all accuracy metrics and vector types. Pearson is slightly better than Covariance, but the difference is not significant. Because accuracy ranking is approximately the same for user- and item-based recommendations, either method could have identified that Cosine and Euclidean are the worst. Note that RMSE, MAE, and CORR rankings agree, meaning low errors translate into high correlation and vice versa.

Item orientation is slightly better for Pearson and Covariance similarity, while user orientation is better for Euclidian and Cosine similarity. Every recommender is better

than the baseline in user orientation. Perhaps this is because many users do not have enough ratings for a meaningful vector mean. For example, an average person has seen at most a few hundred movies, but an average movie can have thousands of opinions that offer a solid indication of the overall item quality [4]. Overall, vector orientation does not dramatically affect recommendation accuracy.

	RMSE	MAE	CORR
user_pearson	1.03196	0.82993	0.40474
user_covariance	1.03486	0.83382	0.40172
user_combo	1.04402	0.8419	0.3799
user_euclidean	1.0499	0.84751	0.36724
user_cosine	1.05118	0.84828	0.36399
user_average	1.06879	0.85086	0.32954
	RMSE	MAE	CORR
item_pearson	1.03057	0.81452	0.40729
item_covariance	1.03454	0.81952	0.40003
item_average	1.05282	0.84985	0.36059
item_combo	1.0613	0.84419	0.34482
item_cosine	1.06753	0.84942	0.33202
item_euclidean	1.06854	0.85058	0.33002

Fig. 2. Recommender Accuracy Metrics (Sorted Best to Worst by RMSE)

B. Classification

Figure 3 shows the recommender classification metrics. By design, the average baseline would always produce positive recommendations, even though some of them would be false. This translates into the worst precision, but best recall, which combine into the best F-Measure across vector orientations. Covariance and Pearson have the second best precision across vector orientations. However, the difference is more significant for users than items. In both cases, Pearson had better recall and Covariance had better precision. The F-measure, shows that Pearson and Covariance are the best overall. Cosine and Euclidean perform similarly, but are at the bottom of the list in every category. Item-based vectors are slightly better overall, but the ranking of the methods is the same, which suggests that the vector orientation does not affect recommendation classification accuracy.

C. Distribution

Figure 4 shows the recommender distribution metrics. Distribution comparison checks the number of predictions in a particular category. Similar to the recommendation accuracy, all predictors are worse than the baseline for user orientation and better than baseline for item orientation. Covariance and Pearson distributions most closely resemble the actual rating distribution across vector orientations. Both similarity measures are more adventurous because they predict more 1's, 2's, and 5's. Euclidean and Cosine have the most conservative distributions, clinging to overall dataset average of 3.6. In fact, all methods correctly recognize that 4 is the most common rating in the quiz dataset, but Covariance and Pearson similarities are the most modest in this category.

	PRECISION	RECALL	F-MEASURE
user_average	55.527%	100.000%	71.405%
user_pearson	65.568%	56.311%	60.588%
user_covariance	66.062%	54.664%	59.825%
user_combo	64.581%	54.901%	59.349%
user_cosine	64.103%	54.468%	58.894%
user_euclidean	64.186%	54.233%	58.791%
	PRECISION	RECALL	F-MEASURE
item_average	56.747%	100.000%	72.406%
item_pearson	65.245%	62.452%	63.818%
item_covariance	65.337%	60.928%	63.055%
item_combo	63.404%	59.546%	61.415%
item_cosine	63.079%	59.295%	61.129%
item_euclidean	63.059%	58.990%	60.956%

Fig. 3. Recommender Classification Metrics (Sorted Best to Worst by F-Measure)

All methods also correctly mimic the skewed bell shape of the actual rating distribution. However, none of them recognize that there are more 5's than 3's in the quiz dataset. Covariance gives out more 3's than any other method and that is the only category where it fails. With the exception of Pearson for item vectors, Cosine similarity gives out the least 3's and that is the only category it excels. No recommender has a perfect distribution, but Covariance and Pearson distributions are slightly better than others. Likewise, item vector orientation is considerably better for every recommender.

	CHI-SQUARED
actual_rating	0
user_average	866277.306
user_covariance	896217.955
user_pearson	927893.59
user_combo	973415.629
user_euclidean	979649.872
user_cosine	982969.72
	CHI-SQUARED
actual_rating	0
item_covariance	796545.953
item_pearson	815992.754
item_combo	866221.863
item_euclidean	866848.522
item_cosine	869844.062
item_average	984520.964

Fig. 4. Recommender Distribution Metrics (Sorted Best to Worst by Chi-Squared)

D. Confusion

The misclassification frequency distribution has the same shape across vector orientations. All methods have trouble guessing 1's, 2's, and 5's. Some methods are better than others, but none of them struggle at guessing the most popular ratings, 3's and 4's. On the other hand, every rating is most often misclassified as 3 or 4, the adjacent categories for the overall rating average. In some cases, large errors are very common.

For example, for an actual rating 1, the least popular guess should be 5, but it is actually more popular than 1. Such gross misclassifications greatly contribute to the overall recommendation error. For an actual rating 2, the worst error would come from misclassifying it as a 5, but it is actually the second least popular misclassification. For an actual rating 3, the least popular options should be 1 and 5 and user-based methods do it perfectly. In general, item-based methods misclassify 3 as a 5 more often than as a 2, which results in a larger error. For an actual rating 4, the worst errors come from misclassifying it as a 1 or a 2. This is the least popular mistake for all methods. For an actual rating 5, the least popular categories should be 1 and 2. All methods correctly reflect that across vector orientations.

In addition to being the best at guessing a particular rating, we also check which predictor is best overall. If the two are different, running them sequentially may be beneficial. Covariance and Pearson are best overall predictors of 1's, 2's, and 3's. They are also best at guessing the actual rating for those categories. Typically Covariance is better than Pearson, except for 4's and 5's. For item vector 4's, Pearson is best overall and with the actual rating. For user vector 4's, Cosine is best overall and Combo is best at guessing the actual rating. For guessing 5's, Pearson and Covariance are best at guessing the actual rating, but Pearson is better overall.

E. Combination

We consider the linear combination of every possible pair of recommendations using different weights with a 10% increment. There are a total of 45 unique combinations of the five user- and five item-based recommenders. For each combination we generate nine new sets of recommendations, each with a different balance between the two recommenders. The results agree with our previous findings. The smallest MAE, RMSE, and the highest CORR came from user_pearson:item_pearson, user_covariance:item_pearson, user_pearson:item_covariance, and user_covariance:item_covariance blends (Best MAE = 0.79276, RMSE = 0.98526, and CORR = 0.52995). All of these combinations perform better than the individual components. The most optimal blends are balanced, 50/50 and 40/60 split between the two recommenders. Furthermore, these results show that user- and item-based approaches are complementary. Therefore, both vector orientations are valuable.

VII. Conclusion

In this paper, we have briefly described the mathematical model of our dataset and gave an overview of the memory-based collaborative filtering process. We also developed a detailed evaluation harness that gives us multiple perspectives into the recommendation quality: accuracy, confusion, classifications, and distribution. Among all similarity metrics, Cosine similarity was the worst. It was often assumed to be the baseline metric for other research papers. Covariance and Pearson similarity measures are by far the best in every category. The accuracy results are consistent across both vector orientations. Item-based recommendation are better and faster than user-based recommendation. However, there is a benefit to blending user- and item-based recommendations that use Covariance and Pearson similarities. We will explore them further in our future work. In particular, we plan to study the Combo similarity weights and improve the accuracy of predicting 1's and 5's.

References

[1] S. M. Ali and I. Ghani, "A rview on rrecommender techniques, systems and evaluation metrics," *Science International*, vol. 24, no. 4, pp. 503–511, 2012.

[2] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowledge-Based Systems*, vol. 23, no. 6, pp. 520 – 528, 2010.

[3] J. Bobadilla, F. Ortega, A. Hernando, and J. Alcalá, "Improving collaborative filtering recommender system results and performance using genetic algorithms," *Knowledge-Based Systems*, vol. 24, no. 8, pp. 1310 – 1316, 2011.

[4] C. Bøe, "Collaborative filtering for recommending movies," Master's thesis, Norwegian University of Science and Technology, 2007.

[5] A. Brun, S. Castagnos, and A. Boyer, "A positively directed mutual information measure for collaborative filtering," in *2nd International Conference on Information Systems and Economic Intelligence - SIIE 2009*, 2009, pp. 943–958.

[6] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Improving k-nearest neighbors algorithms: practical application of dataset analysis," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, ser. CIKM '11. New York, NY, USA: ACM, 2011, pp. 2253–2256.

[7] L. Candillier, F. Meyer, and M. Boullé, "Comparing state-of-the-art collaborative filtering systems," in *Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern Recognition*, ser. MLDM '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 548–562.

[8] K. Choi and Y. Suh, "A new similarity function for selecting neighbors for each target item in collaborative filtering," *Knowledge-Based Systems*, no. 0, 2012.

[9] S. Cleger-Tamayo, J. M. Fernández-Luna, and J. F. Huete, "A new criteria for selecting neighborhood in memory-based recommender systems," in *Proceedings of the 14th international conference on Advances in artificial intelligence: spanish association for artificial intelligence*, ser. CAEPIA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 423–432.

[10] J. de Wit, "Evaluating recommender systems," Master's thesis, University of Twente, May 2008.

[11] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," *Recommender Systems Handbook*, vol. 69, no. 11, pp. 107–144, 2011.

[12] S. Gualdi, M. Medo, and Y.-C. Zhang, "Crowd avoidance and diversity in Socio-Economic systems and recommendation," Jan. 2013.

[13] D. Jia, F. Zhang, and S. Liu, "A robust collaborative filtering recommendation algorithm based on multidimensional trust model," *Journal of Software*, vol. 8, no. 1, 2013.

[14] O. Jojic, M. Shukla, and N. Bhosarekar, "A probabilistic definition of item similarity," in *Proceedings of the fifth ACM conference on Recommender systems*, ser. RecSys '11. New York, NY, USA: ACM, 2011, pp. 229–236.

[15] H. MA, "Learning to recommend," Ph.D. dissertation, The Chinese University of Hong Kong, December 2009.

[16] B. Marlin, "Collaborative filtering: A machine learning perspective," Master's thesis, University of Toronto, 2004.

[17] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: how accuracy metrics have hurt recommender systems," in *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '06. New York, NY, USA: ACM, 2006, pp. 1097–1101.

[18] A. Nanopoulos, M. Radovanović, and M. Ivanović, "How does high dimensionality affect collaborative filtering?" in *Proceedings of the third ACM conference on Recommender systems*, ser. RecSys '09. New York, NY, USA: ACM, 2009, pp. 293–296.

[19] G. Özbal, "A content boosted collaborative filtering approach for movie recommendation based on local & global user similarity and missing data prediction," Master's thesis, Middle East Technical University, September 2009.

[20] A. Said, B. J. Jain, and S. Albayrak, "Analyzing weighting schemes in collaborative filtering: cold start, post cold start and power users," in *SAC*, 2012, pp. 2035–2040.

[21] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. in Artif. Intell.*, vol. 2009, pp. 4:2–4:2, Jan. 2009.

[22] H. Sun, Y. Peng, J. Chen, C. Liu, and Y. Sun, "A new similarity measure based on adjusted euclidean distance for memory-based collaborative filtering," *JSW*, vol. 6, no. 6, pp. 993–1000, 2011.

[23] J. Wang, A. P. de Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '06. New York, NY, USA: ACM, 2006, pp. 501–508.

[24] W. Z. Yongli Ren, Gang Li, "Automatic generation of recommendations from data: a multifaceted survey," Deakin University, Tech. Rep. TR C11/4, August 2011.

[25] C. Zhao, Q. Peng, and C. Liu, "An improved structural equivalence weighted similarity for recommender systems," *Procedia Engineering*, vol. 15, no. 0, pp. 1869 – 1873, 2011, cEIS 2011.

[26] Z. G. Zhenxue Zhang, Dongsong Zhang, "Improving memory-based collaborative filtering using a factor-based approach," in *In Proceedings of AAAI'07 Workshop on Recommender Systems in E-commerce*, 2007.

[27] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web*, ser. WWW '05. New York, NY, USA: ACM, 2005, pp. 22–32.