Judging Similarity: A User-Centric Study of Related Item Recommendations

Yuan Yao
University of Minnesota
yaoxx340@umn.edu

F. Maxwell Harper
University of Minnesota
max@umn.edu

ABSTRACT

Related item recommenders operate in the context of a particular item. For instance, a music system's page about the artist Radiohead might recommend other similar artists such as The Flaming Lips. Often central to these recommendations is the computation of similarity between pairs of items. Prior work has explored many algorithms and features that allow for the computation of similarity scores, but little work has evaluated these approaches from a user-centric perspective. In this work, we build and evaluate six similarity scoring algorithms that span a range of activity- and content-based approaches. We evaluate the performance of these algorithms using both offline metrics and a new set of more than 22,000 user-contributed evaluations. We integrate these results with a survey of more than 700 participants concerning their expectations about item similarity and related item recommendations. We find that content-based algorithms outperform ratings- and clickstream-based algorithms in terms of how well they match user expectations for similarity and recommendation quality. Our results yield a number of implications to guide the construction of related item recommendation algorithms.

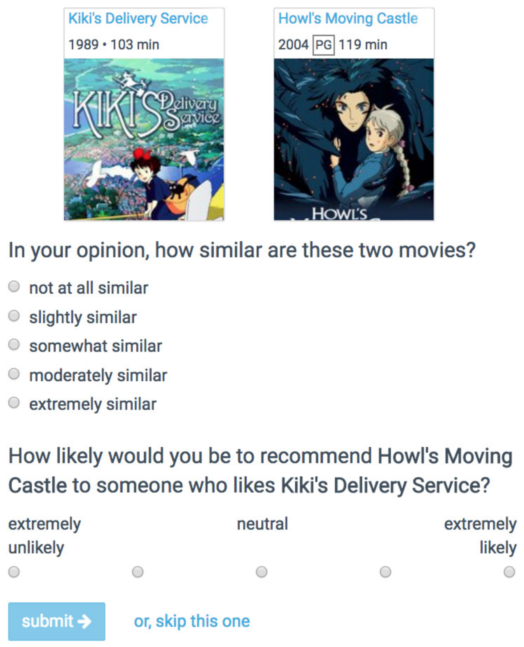
CCS CONCEPTS

• **Information systems → Recommender systems**; *Similarity measures*; • **Human-centered computing → Collaborative filtering**; *User studies*;

KEYWORDS

recommender systems; related item recommendations; similarity metrics; rule mining, collaborative filtering, word2vec, user survey.

ACM Reference Format:
Yuan Yao and F. Maxwell Harper. 2018. Judging Similarity: A User-Centric Study of Related Item Recommendations. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240323.3240351



In your opinion, how similar are these two movies?

- not at all similar
- slightly similar
- somewhat similar
- moderately similar
- extremely similar

How likely would you be to recommend Howl's Moving Castle to someone who likes Kiki's Delivery Service?

extremely unlikely — neutral — extremely likely

Figure 1: Partial screenshot of the survey interface for evaluating seed-neighbor pairs.

1 INTRODUCTION

One of the most pervasive content recommendation strategies, known as "related item recommendations" [1, 32], shows users relevant items in the context of a particular *seed item*. For instance, the music streaming service Spotify shows "Related Artists" when viewing information about an artist; e-commerce site Amazon shows "Customers who viewed this item also viewed" when viewing information about a product; video streaming service YouTube shows what's "up next" near the video that is currently playing.

Fundamental to related item recommendations is the computation of similarity between items. Algorithms that compute similarity are typically designed to reduce the complex relationship between a pair of items down to a single score that represents relatedness. A related item recommender may either use these scores directly to rank, or may build a candidate set from similarity scores that is the input to a separate ranking step [12, 38].

There are many possible approaches to computing similarity scores between pairs of information entities, since the entities themselves may be described by different types of data. For instance, a music system might track many features about artists, including metadata (e.g., artist name, genres, musicians), unstructured text (e.g., critic or user reviews), user behavioral data (plays, clicks, 5-star ratings), and raw content (audio or video files) [33]. Any of these features may potentially be processed to build item representations or statistical models that may be used to compute similarity scores between pairs of items.

Understanding similarity algorithms is made more complex because of the challenges in evaluating algorithmic output. Although we can build metrics that *describe* the output of these algorithms in terms of metadata or user behavior, there is typically no ground truth concerning similarity. Who are the most similar artists to The Beatles? The answer is open to users' interpretation.

Therefore, in this research, we seek to build a user-centric [19] understanding of item similarity algorithms and their relationship to recommendation quality. We consider two research questions:

> *RQ-ALG. Which related item algorithms best match user perceptions of relatedness and recommendation quality?*

Prior work has examined similarity metrics from the perspective of optimizing an input to a collaborative filtering algorithm (e.g., [3, 31, 39, 40]), and from the perspective of optimizing business outcomes such as click-through rate (e.g., [13, 17]). However, there are few instances of work attempting to relate algorithmic similarity scores to user perceptions of suitability for recommendation. In this research, we supplement offline metrics with a new dataset of tens of thousands of human judgments to learn more about the factors that lead people to judge a related item to be similar or a good recommendation.

> *RQ-UX. How should related item algorithms be designed to improve the user experience?*

The vast majority of prior work on related item recommendations has focused on algorithmic and computational issues (e.g., [12, 17, 38]), while less work has focused on user-centric issues such as the "similarity hole" that item-based collaborative filtering systems can fall into [25]. Through a user survey, we seek to develop an understanding of user expectations and beliefs around item similarity and related item recommendations. For example, should algorithms prioritize similarity or user relevance?

2 RELATED WORK

This research seeks to learn more about two concepts: *item similarity* and *related item recommendations*. These concepts are related, but distinct. It is possible to use the n-most-similar items directly as recommendations in an interface. A two-stage recommendation process is also common [7], where the first stage builds recommendation candidates from the set of similar items, and the second stage ranks those candidates (e.g., [12, 17]). In this section we discuss prior research related to both of these concepts.

2.1 Item Similarity

Item similarity computations are central to item-based collaborative filtering (CF) recommender systems [14, 23, 31]. These systems typically work by applying a k-nearest neighbors (KNN) algorithm to find predicted rating values. The KNN algorithm locates the nearest neighbors using a similarity metric computed across items. Because item-based CF algorithms are so commonly used, there has been substantial work to understand which processes for computing similarity scores lead to the greatest accuracy in predicting ratings. Classically, research has examined different metrics such as cosine similarity that operate on traces of user activity. For example, many systems rely on vectors of user ratings [14, 31], which may be factorized using a technique like singular value decomposition (SVD) [20]. Other systems rely on *implicit* activity data [4, 16] such as clicks or purchases as the source material for computing similarity scores.

Content-based approaches use domain-specific features (e.g., genres, keywords) to compute item similarity scores. These approaches, unlike those based on user activity, do not have to deal with the *cold-start problem* [29] that exists before sufficient activity is logged about new items or users. Content-based recommender systems may use relatively simple information retrieval methods, such as keyword matching [11, 28], or the vector space model with basic TF-IDF weighting [24]. User-applied tags have been explored in this context [35]; the tag genome algorithm [38] encodes an item in a vector space based on its relationship to a common set of tags. Word2vec [26], along with the related doc2vec [21] and other "2vec" algorithms (e.g., [6]), represent another set of methods of encoding free text as numeric vectors.

Less work has involved users in judging the output of item similarity computations. In the domain of music, researchers have collected datasets of human judgments of similarity between songs using volunteers [15, 22]. To develop a "ground truth" for similarity in music, one effort engaged audio experts to provide similarity judgments [37]. One study of crowdsourced music similarity judgments found that aggregating responses from multiple judges has a large impact on accuracy: aggregating three judgments improved RMSE by 55% over using a single judgment [27].

Most related to this paper, Colucci et al. [11] ask study participants ($N = 14$) to provide yes/no answers about whether pairs of movies are "similar", collecting 3,803 binary labels across 1,416 movies. They used this dataset to evaluate different similarity algorithms, finding that cosine similarity on 5-star ratings outperformed both pearson correlation on 5-star ratings and a TF-IDF method based on metadata (all of which underperformed the output from a third-party website). They also used this dataset of similarity judgments in follow-up work [39] to train a content-based recommender system that outperformed a ratings-based method.

In this research, we contribute the most extensive and detailed data collection of user similarity judgments in our domain (movies) that we know of. We have openly released our dataset of survey responses and similarity judgments to facilitate more research in this area (available at http://hdl.handle.net/11299/198736).

2.2 Related Item Recommendation

Related item recommenders rank items in the context of a particular seed item — typically, the item the user is currently viewing. They are used in many different domains [32]. YouTube has published a number of papers on their approaches to "up next" video recommendation [12, 13, 41]. Other research has looked at related item recommendations in the context of e-commerce [17, 23, 36] and web portals or news aggregators [1].

In this research, we seek to contribute data on how users reason about item similarity and related item recommendations. The studies of related item recommendations (that we know of) report on performance-oriented metrics (e.g., click-through rate). We are not aware of any published user-centric research of related item recommendations.

3 METHODS

Our research goal is to learn from users about which types of algorithms best match user expectations about similarity and recommendation quality (RQ-ALG), and how related item recommendations should be designed (RQ-UX). We study these questions in the context of movies, an area with an extensive history of recommendation technology research [9] and industrial adoption of related item recommendations (e.g., Netflix, Amazon Prime). Movies are taste-based (different people like different movies) and complex (there are many features that describe them), properties that allow us to study our research questions.

We conducted this work on MovieLens (movielens.org), a movie recommendation website with thousands of active monthly users. MovieLens provides us with a rich dataset of user contributions, including ratings, tags, and browsing logs, as well as a large potential pool of participants who are interested in movies. MovieLens does not collect user review data; for the text processing pieces of this research we look at user reviews publicly available on imdb.com.

Our data collection methods have three main phases, which we describe in detail below:

(1) We use a stratified random sampling to choose a test set of 100 seed movies.
(2) We develop and train six related item algorithms, and run each of them to retrieve the top ten neighbors for each seed.
(3) We deploy a user survey to collect quality judgments and responses to a variety of questions about related item recommendations.

3.1 Sampling Strategy

Fundamental to all phases of this research, we choose a test set of 100 seed movies for our algorithms to process. Since our research questions rely on the presence of users who are familiar with pairs of movies, we select seed movies into the test set based on their popularity (measured by the total number of user ratings).

Specifically, we only consider seed movies that are in the top 2,500 most popular movies in the MovieLens database. While the MovieLens database currently has information about more than 58,000 movies, the average user has never heard of most of these. The movies in MovieLens follow a power law distribution in terms of ratings — a few movies account for many ratings, while the long tail contains many movies with few ratings. We choose to sample from the top 2,500, since these movies account for more than 80% of user ratings in the system, striking a balance between recognizability and coverage.

Further, to ensure some degree of evenness in the spread of popularity throughout the test set, we use a stratified random sampling method. Specifically, we divide the top-2500 movies into 10 tiers (strata) of size 250. We draw 10 movies at random from each of these strata. Therefore, we guarantee that we have exactly 10 movies in the top-250 most popular movies, exactly 10 movies that rank from 251-500 in popularity, and so on.

3.2 Algorithms

We test six algorithms for computing item similarity in a movie recommender system. Each algorithm generates top-N lists of *neighbors* simply by ranking items by similarity score. These algorithms operate on a range of input data, similarity metrics, and algorithms. We choose this set of algorithms because it covers some of the most common techniques from ratings-, clickstream-, and content-based approaches. See Table 1 for an overview of the algorithms.

Table 1: Overview of the six algorithms.

name	type [18]	sim. metric	core algorithm
baseline	N.A.	N.A.	random
rating-svd	user-rating	cosine	SVD [30]
click-arm	co-occurrence	conviction	ARM [2]
click-vecs	co-occurrence	cosine	item2vec [6]
tag-genome	text-retrieval	cosine	tag genome [38]
review-vecs	text-retrieval	cosine	doc2vec [21]

We optimized each algorithm by hand using the offline metrics described below in the results section. Here, we describe the six algorithms alongside the most important configuration parameters and data pre-processing steps.

baseline selects neighbors at random. Although this is a poor strategy, the presence of a random baseline allows us to calibrate user similarity judgments and the margin of improvement provided by each of the other five algorithms.

rating-svd applies singular value decomposition (SVD) [30] to the user-item ratings matrix, then uses cosine similarity to compute similarity scores between pairs of items [31]. We choose this strategy due the popularity of item-based collaborative filtering, a personalized method for predicting ratings. Ratings are from the MovieLens database, where values range from 0.5 to 5 in half-star increments. We configure the SVD algorithm with 100 features, we preprocess the input matrix with mean subtraction, and we normalize each row of output matrix to the unit vector.

click-arm applies association rule mining (ARM) [2] to a dataset of per-session clickstream data, then uses the conviction metric [8] to measure pairwise similarity scores. We include this strategy because it is a commonly used data mining technique in e-commerce for making related item recommendations. We use a simplified definition of per-session clickstream: the list of movie detail pages that a user views; we therefore can represent a *transaction* as a set of movie identifiers. MovieLens divides user activity into separate sessions when the period of inactivity is one hour or more. We use conviction (a measure of the degree of implication of an association rule relative to its independence) as our similarity metric, since it has been found to perform well in this context [5].

click-vecs applies item2vec [6] to a dataset of per-session clickstream data (described above) to yield vector embeddings of items, then uses cosine similarity to compute similarity scores. Item2vec is closely related to the better-known word2vec [26], an algorithm that achieves state-of-the-art results on linguistics tasks. Item2vec substitutes items for words (in our case, clicked-on items), and discards the spatial information of the items in a basket. We train the algorithm for 50 epochs, we set the negative sampling value to 15, and we set the dimension parameter to 100.

tag-genome computes item similarities by first predicting relevance scores between a set of common tags and each movie, then by computing the cosine similarities between these vectors of tag

Table 2: Examples of algorithm output: top-3 similar movies for two seed movies in the test set.

		Lion King, The (1994)	
	#1	#2	#3
baseline	Big Red One, The (1980)	Most Dangerous Game, The (1932)	Bo Burnham: Make Happy (2016)
rating-svd	Pocahontas (1995)	Mulan (1998)	Little Mermaid, The (1989)
click-arm	Aladdin (1992)	Beauty and the Beast (1991)	Toy Story (1995)
click-vecs	Mulan (1998)	Mask, The (1994)	Up (2009)
tag-genome	Aladdin (1992)	Tarzan (1999)	Beauty and the Beast (1991)
review-vecs	Lion King II: Simba's Pride (1998)	Lion King $1\frac{1}{2}$, The (2004)	Aladdin (1992)
		Mad Max: Fury Road (2015)	
	#1	#2	#3
baseline	Tale of Ham and Passion, A (1992)	Entity, The (1981)	Hysterical Blindness (2002)
rating-svd	Get Out (2017)	Baby Driver (2017)	Ex Machina (2015)
click-arm	Ex Machina (2015)	Interstellar (2014)	Whiplash (2014)
click-vecs	Star Wars: Episode VII (2015)	Ex Machina (2015)	Deadpool (2016)
tag-genome	Road Warrior, The (1981)	Logan (2017)	Dredd (2012)
review-vecs	Road Warrior, The (1981)	Mad Max Beyond Thund…(1985)	Terminator Salvation (2009)

relevance scores [38]. This approach uses supervised learning to predict relevance scores based on a combination of data, including user-applied tags [34] and unstructured text from user-written movie reviews. It is the current method used by MovieLens to show "similar movies" in the user interface, and is configured as described in Vig et al.'s paper [38].

review-vecs applies doc2vec [21] to unstructured text from user-written movie reviews to generate vector embeddings of each movie, then uses cosine similarity to compute similarity scores. We selected doc2vec because it represents a modern and powerful unsupervised learning approach, and because it can operate entirely on unstructured text. We used user-contributed reviews (2.5 million reviews; 2.3 GB of data) of movies to train the model; we used all reviews of a particular movie to learn its doc2vec embedding. We pre-processed the reviews to retain only named entities (e.g., people, or organization names), adverbs, adjectives, and compound modifiers. We learn the doc2vec model using Gensim [42], using the distributed bag of words (PV-DBOW) algorithm. We set the dimensionality of the feature vectors to 100, and we train the algorithm for 50 epochs.

To generate the *evaluation set* of seed-neighbor pairs, we ran each of the six algorithms to generate the top-10 neighbors for each of the 100 movies from the test set. We limit the algorithms to consider only the top 10,000 most popular movies in MovieLens (combined, these movies represent more than 98% of the system's ratings) to improve the runtime and slightly increase the potential for human evaluations. Therefore, we have an evaluation set of 1,000 seed-neighbor pairs for each algorithm (6,000 total pairs), which is the basis for subsequent analysis. See Figure 2 for examples.

3.3 User Survey

We conduct a user study to collect data about user preferences, and to develop a set of user quality evaluations that inform our understanding of related item algorithm performance. This study was approved by our university's institutional review board.

We recruited participants for our user study by email. Between March 21 and March 28, 2018, we emailed 24,111 randomly-selected MovieLens users who had valid email addresses, who had logged in during the past 2.5 years, and who had at least 15 ratings. 1,726 users (7.2%) clicked through to the survey, and 998 users (4.1%) consented to participate. We did not provide any incentives for participation. We collected data through April 16, 2018.

The first part of the survey asks participants their opinions about the importance of related item recommendations, about how they expect related item recommendation to work, and about the factors they use in evaluating the similarity of movies. We share the exact wording of questions below in Table 6.

The second part of the survey aims to collect quality judgments about specific pairs of movies. Although each of the six algorithms generated 1,000 seed-neighbor pairs, there is overlap in the output of the six algorithms, yielding a total of 4,912 *unique* pairs that we potentially show to users.

To evaluate a pair, we require that a user has rated both the seed and the neighbor (typically, in MovieLens, users rate only movies they have watched). In order to increase the possible number of pairs a participant can evaluate, we include a "rate more" interface as the second part of the survey. In this interface, we present users with lists of movies to rate. The first list we show contains seed (test set) movies, sorted by inverse popularity to boost the potential for collecting data on obscure pairs. The second list we show contains neighbors, sorted randomly, that, if rated, would provide a new possible pair evaluation. With each list, we present 72 movies in a grid, and ask the user to rate as many as possible.

Finally, as shown in Figure 1, we present users with a series of pairs of movies to evaluate. We allow users to answer as many of these as they wish (limited by the number of pairs where they have rated both the seed and the neighbor); we first ask for 10 in order to complete the survey, then allow users to keep going if they wish. Because we anticipate that it will be more difficult to collect judgments for more obscure movies, we order the sequence of pairs to compensate: given the set of pairs the user is qualified to evaluate, we order by the inverse popularity of the seed movie. This has the effect of sometimes showing multiple pairs in sequence with

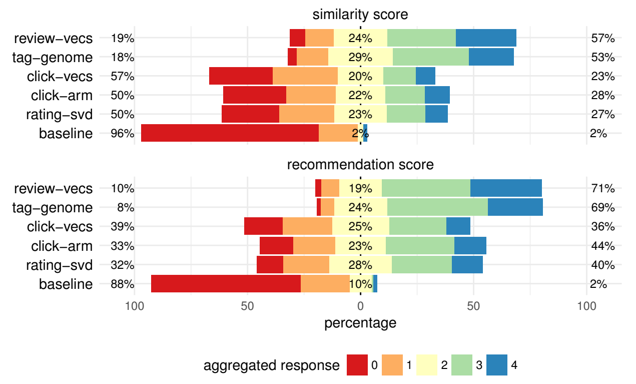


Figure 2: Distributions of aggregated similarity and recommendation score by algorithm. For example, 57% of seed-neighbor pairs from review-vecs received an aggregated similarity score of 3 or 4 ("moderately" or "extremely" similar), while 96% of the pairs from the baseline algorithm received a score of 0 or 1.

the same seed movie — in this case, we order the pairs randomly. We chose this design to help users move through the survey more quickly, since this allows them to keep an impression of the seed movie in their head as they compare it to multiple neighbors. We show the seed (left) and neighbor (right) side-by-side, and ask two questions. The first question asks about how "similar" the two movies are, and the second asks about the "recommendation quality" of the neighbor for someone who likes the seed movie.

4 RESULTS: ALGORITHM COMPARISON

We collected 23,017 seed-neighbor pair evaluations from 636 users ($min = 1$, $median = 25$, $max = 285$), each yielding two metrics: *similarity score* ("In your opinion, how similar are these two movies"), and *recommendation score* ("How likely would you be to recommend X to someone who likes Y"). These scores are ordinal, ranging from 0–4. To reduce the effect of user bias in the analysis [27], we discard data about pairs with fewer than three evaluations, which leaves 21,187 pairs. These pairs yield the following coverage (out of the 1000 possible seed-movie pairs per algorithm): baseline (12.1%), rating-svd (54.3%), click-arm (76.4%), click-vecs (45.8%), tag-genome (46.2%), and review-vecs (34.6%); lower coverage scores are associated with algorithms that tend to generate more obscure neighbors.

4.1 User Evaluation

See Figure 2 for a visualization of the distributions of similarity and recommendation scores for each of the six algorithms. To generate this figure, we developed *aggregated* similarity and recommendation scores for each seed-neighbor pair by taking the median of all user judgments (discarding pairs with fewer than three scores). We see that review-vecs generates neighbors that earned the greatest number of high scores (≥ 3) across both metrics.

To test if the apparent differences between algorithms are statistically significant, we employ ordinal regression using a cumulative link mixed model (CLMM) [10], treating algorithm as the fixed effect and user ID as the random intercept. We build two models, predicting aggregated similarity and recommendation scores. In both models, the ranking in terms of effect size is *review-vecs* > *tag-genome* > *rating-svd* > *click-arm* > *click-vecs* > *baseline* (all algorithms are statistically significant predictors, $p < 0.001$).

Table 3: Overlap of the five non-baseline algorithms, using Jaccard index of the seed-neighbor pairs. Baseline comparisons are not shown; overlap scores between the baseline and the other five algorithms is ≤ 0.003.

	rating-svd	click-arm	click-vecs	tag-gen.
click-arm	0.086			
click-vecs	0.083	0.072		
tag-genome	0.091	0.153	0.059	
review-vecs	0.056	0.086	0.039	0.139

To compare the similarity and diversity of neighbors generated by different algorithms for seed movies, we measure the overlap of evaluation set of seed-neighbor pairs between algorithms using Jaccard index. We see that the six algorithms are largely distinct in the neighbors they emit in their top ten lists – the two most similar algorithms using this metric (click-arm and tag-genome) have a Jaccard index of just 0.15. See Table 3.

Finally, we find that our two user-generated metrics are highly correlated. The Spearman rank-order correlation coefficient of similarity scores and recommendation quality scores is 0.80 ($p < 0.01$).

4.2 Offline Evaluation

To discover potential reasons for the differences in similarity and recommendation scores between algorithms, we analyze the pairs on eight factors from the MovieLens database (see Table 4). The first three factors measure the absolute value of differences between the seed and the neighbor (labelled "diff" in the table) in terms of popularity (number of ratings in the last year), release year, and average rating. Smaller diff values represent higher similarity — e.g., a small mean popularity diff means that an algorithm tends to generate neighbors with a similar level of popularity to the seed item. The remaining five factors measure the overlap between each seed-neighbor pair in terms of genres, directors, cast, collection (e.g., the Star Wars movies are in the same collection), and title. We use Jaccard index for all of these overlap factors except title, where we instead use a string similarity metric (using Python difflib) in the range of 0–1. Higher overlap indicates higher similarity — e.g., a high genre overlap means that an algorithm tends to generate neighbors with a similar set of genres to the seed item.

Tag-genome generates neighbors that are most similar with the seed in terms of genre overlap, while review-vecs generates neighbors with the highest director, cast, collection, and title similarity. Rating-svd has the most similarity between seeds and neighbors in terms of release year, popularity, and average rating.

We fit an ordinal regression model to predict the user-generated similarity and recommendation scores, using the eight factors discussed above as independent variables. The coefficients are shown in Table 5. The direction of the coefficients matches our intuition:

Table 4: Offline metrics describing the relationship between seed and neighbor. Each cell represents mean ± std. dev.

factor	baseline	rating-svd	click-arm	click-vecs	tag-genome	review-vecs
popularity diff	598.93 ± 824.96	**347.93 ± 568.00**	1170.52 ± 1532.85	361.06 ± 559.13	512.63 ± 706.77	525.86 ± 745.49
release year diff	16.76 ± 15.94	**3.31 ± 4.18**	6.24 ± 8.95	5.16 ± 6.78	9.16 ± 9.87	9.42 ± 11.07
avg. rating diff	0.56 ± 0.42	**0.18 ± 0.15**	0.33 ± 0.31	0.25 ± 0.23	0.22 ± 0.17	0.36 ± 0.31
genre overlap	0.16 ± 0.21	0.42 ± 0.28	0.44 ± 0.29	0.42 ± 0.28	**0.56 ± 0.28**	0.50 ± 0.28
director overlap	0.00 ± 0.04	0.05 ± 0.21	0.09 ± 0.28	0.04 ± 0.19	0.07 ± 0.25	**0.10 ± 0.29**
cast overlap	0.00 ± 0.01	0.01 ± 0.04	0.01 ± 0.05	0.01 ± 0.03	0.01 ± 0.05	**0.02 ± 0.05**
collection overlap	0.00 ± 0.03	0.02 ± 0.14	0.04 ± 0.19	0.02 ± 0.15	0.04 ± 0.20	**0.06 ± 0.23**
title similarity	0.00 ± 0.00	0.00 ± 0.03	0.01 ± 0.07	0.00 ± 0.00	0.00 ± 0.06	**0.01 ± 0.10**

Table 5: Ordinal regression models predicting similarity and recommendation. Values are coefficients, with standard error in parentheses.

predictor	similarity	recommendation
release year diff	$-0.007\,(0.001)^{***}$	$0.002\,(0.001)$
log(popularity diff)	$-0.086\,(0.008)^{***}$	$-0.035\,(0.008)^{***}$
rating diff	$-0.576\,(0.042)^{***}$	$-0.623\,(0.040)^{***}$
genre overlap	$2.429\,(0.048)^{***}$	$2.058\,(0.047)^{***}$
director overlap	$0.972\,(0.056)^{***}$	$0.957\,(0.056)^{***}$
cast overlap	$10.730\,(0.952)^{***}$	$10.862\,(0.982)^{***}$
collection overlap	$2.170\,(0.141)^{***}$	$1.465\,(0.134)^{***}$
title similarity	$2.985\,(0.240)^{***}$	$1.649\,(0.208)^{***}$

$^{*}p < 0.1$; $^{**}p < 0.05$; $^{***}p < 0.01$

smaller diffs and more overlap between a seed and a neighbor predict higher scores. The two models mostly agree, though popularity, genre, collection, and title have a stronger positive predicted effect for similarity than recommendation.

5 RESULTS: USER SURVEY

We collected 734 survey responses to our questions of how people reason about related movie recommendations. In this section, we describe these responses, grouped into four themes: importance, user experience, judging similarity, and open-ended responses.

5.1 Importance

Participants report that related item recommendations are very important to how they find movies. 52.2% of participants claimed to"often" or "always" seek a similar movie after watching a movie they enjoy (see Table 6 Q1). 47.0% of participants ranked "similar movies" as the most important recommendation feature in MovieLens, beating "top picks" (26.0%) and "recommendations by genre" (19.2%) (see Figure 3).

5.2 User Experience

We asked several questions related to the user experience in MovieLens around related item recommendations. See Table 6 (Q2–Q4) for details about the questions and the distributions of responses pertinent to this section.

Table 6: Survey questions and responses

Q1. *After watching a movie that you enjoyed, how often do you seek out a similar movie to watch next?* (N=734)		
9	(1.2%)	never
83	(11.3%)	rarely
262	(35.3%)	sometimes
334	(45.5%)	often
49	(6.7%)	always
Q2. *When showing similar movies, MovieLens should ...* (N=730)		
194	(26.6%)	display the most similar movies
277	(37.9%)	prioritize similar movies, as long as they somewhat fit my tastes
168	(23.0%)	prioritize movies that fit my tastes, as long as they are somewhat similar
91	(12.5%)	prioritize movies that fit my tastes, even if they are not very similar
Q3. *When showing similar movies, MovieLens should display movies that I have already rated.* (N=728)		
308	(42.3%)	yes
420	(57.7%)	no
Q4. *MovieLens should ensure that the similar movies section contains a variety of types of movies.* (N=734)		
32	(4.4%)	strongly disagree
101	(13.8%)	disagree
264	(36.0%)	neutral
238	(32.4%)	agree
99	(13.5%)	strongly agree

When displaying related item recommendations, it is not possible to optimize both item similarity and user relevance simultaneously. Using a four point scale (see Table 6 Q2), we asked participants whether they would prioritize similarity, relevancy, or a point in-between. 64.5% of participants chose one of the two options skewing more towards similarity than towards relevancy. Fully prioritizing relevance was the least popular choice, with 12.5% of responses.

It is not certain how users think about the utility of related items. While some users might view them purely as recommendations,

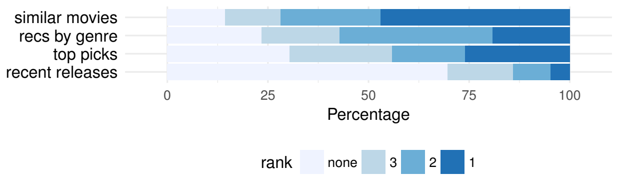


Figure 3: Responses to the prompt "how important are the following MovieLens recommendation features to you?", where we asked participants to rank the "most important", "2nd most important (if any)" and "3nd most important (if any)" (N = 734). For example, 345 (47.0%) participants ranked "similar movies" as the most important feature.

others might view them as items that facilitate exploration. We asked users if they wish to view movies they have already rated in the list of similar movies (see Table 6 Q3). The slight majority (57.7%) chose "no", that they would prefer only unrated (i.e., not-yet-viewed) items in these lists.

Finally, we measure how much users appear to value diversity in related item recommendations. We asked users on a standard 5-point likert scale to state their degree of agreement with a statement that it is important to see a variety of items in lists of similar movies (see Table 6 Q4). Few participants disagreed or strongly disagreed with this statement (18.1%). The most common response was neutral (36.0%), followed by agree (32.4%).

5.3 Recommendation Factors

We asked users to rate the importance of 15 different factors that possibly can help find a similar movie to watch next using a 5-point likert scale of agreement (see Figure 4). These factors range from straightforward (e.g., genre) to complex (e.g., mood). Some factors are computable from usage data (e.g., average user rating, popularity). Other factors rely on expert critical judgments (e.g., critical reception, awards).

According to users, the most universally-important feature is genre (75% strongly/agree). The next most important features are all complex movie-specific features — mood (62% strongly/agree), plot (57% strongly/agree), and theme (51% strongly/agree). With respect to activity-based features, users report average user rating (46% strongly/agree) to be much more important than popularity (18% strongly/agree).

We asked users to provide free-text responses to "other factors that you consider" to get a sense for the range of factors that participants use to judge similarity (N = 121). The most common "other" factor was director (N = 32), followed by variants of "liked by the same people" (N = 11), soundtrack (N = 7), and length (N = 6). Participants contributed a variety of other factors they consider important, including more details about cast and crew (producer, writer, composer), marketing materials (cover art, trailers), relationships to other media (prequel/sequel, based on a game or book), genre-specific features (level of violence, special effects, subtitles vs. dubbed foreign movies), and more movie qualities (availability, budget, production country, visual quality, style, tone, political orientation, quality, energy/pace, sub-genre/keywords).

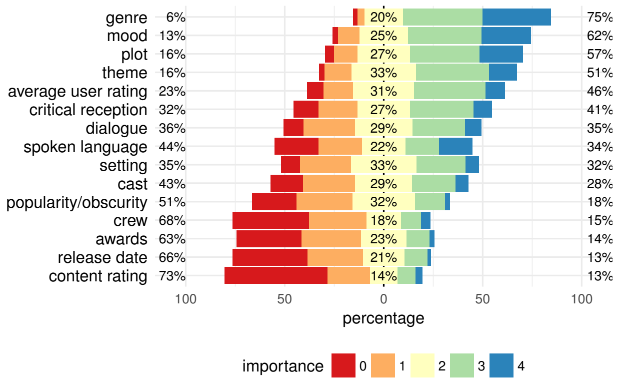


Figure 4: Responses to the prompt "rate the importance of the following factors in helping you find a similar movie to watch next". Answers range from 0 ("not important") to 4 ("very important"). For example, 75% of participants responded to "genre" with a score of 3 or 4, while 66% of participants responded to "release date" with a 0 or 1.

5.4 Open-Ended Responses

We provide an opportunity for participants to leave open-ended feedback with the prompt "If you have any other feedback about similar movie recommendations in MovieLens, please leave it here" (N = 50). Two common themes relating to the design of related item recommenders emerge from this feedback.

First, several participants discussed how the most similar items are not necessarily the most interesting items to display. In particular, some similar movies strike users as "obvious", and might be a poor choice for display. For instance:

> *obvious similarity and obvious relationship is not the point for me...the finding of inobvious [sic] relationships is the key feature*

Other subjects agreed, stating that MovieLens should "limit the amount of very obvious similar movies", that the interface "should not include sequels", and that "I [...] get annoyed when the similarities are superficial". One participant frames this point in terms of diversity, stating "I prefer seeing a selection of lesser-known similar movies alongside the more obvious choices."

Second, several participants indicated a desire for having control over the related item interface. For example, perhaps it is not always desirable to find the most similar movies *overall*, but rather based on a specific aspect of the item:

> *Sometimes there's an aspect of a movie I like that (for example, setting) which most movies considered "similar" don't share. It would be nice to be able to pick a tag of the movie I like and base the "similar movies" on that tag. For example, if I liked "Brave" because it's set in Scotland, I could select that tag and find "Trainspotting" and "Local Hero" instead of "Toy Story"*

Other participants suggested that "I want to be able to specify the tags on which to key recommendations" and that "I would like interactive controls — more/less diversity, filter by cast/crew, dialog by grade level, etc." Other suggestions included the ability to show or hide already-rated movies, or to change the sort order from most similar to a blend of similar and relevant.

6 DISCUSSION

RQ-ALG. Which related item algorithms best match user perceptions of relatedness and recommendation quality? In this experiment, content-based algorithms are a better match for user expectations than ratings- and clickstream-based (i.e., activity-based) algorithms. Content-based algorithms were the top performers on both user metrics (similarity and recommendation). The content-based approaches also performed best according to several offline metrics — including genre and movie collection overlap — that rank as important factors in the survey results. One clear implication of this work is that builders of related item recommenders should *consider a content-based approach*. In this domain, free text appears to work slightly better than tags, overall.

The two user-generated outcome metrics that we developed have a high correlation (0.80) — it is not clear if this correlation is an indication of a generalizable relationship or an artifact of the survey design. There were 49 seed-neighbor pairs where the aggregate similarity score was at least three points higher than the aggregate recommendation score. About half of these pairs consist of prequels (e.g., *Rocky 2* is similar to, but not a good recommendation for, *Rocky 3*) and remakes (e.g., the 2005 Keira Knightley *Pride and Prejudice* is similar to, but not a good recommendation for the 1995 Jennifer Ehle *Pride and Prejudice*). We think there is interesting future work in better understanding cases like these where similarity scores and recommendation scores disagree.

RQ-UX. How should related item algorithms be designed to improve the user experience? Fundamentally, we found that related item recommendations are important. Participants in our survey ranked related item recommendations to be more important than per-genre recommendations or overall recommendations. Further, more than half of the participants claim to "often" or "always" look for similar movies to watch after watching a movie they have enjoyed. Therefore, one implication of this work is that *related item recommendations play an important role in a recommender system.*

Users do not want related item recommenders to simply display the most similar items. When we asked survey participants explicitly about how they would prioritize displaying the most similar items versus the items best fitting their taste, more than 60% chose a compromise. That is, users don't want just the most similar items, and they don't want just the top-rated items, they want something in-between. Further, many users desire some degree of diversity in related item recommendations. Finally, several users left open-ended feedback that "obviously similar" items are a mark of bad recommendations. Therefore, an implication of this work is that *related item recommendations should blend item similarity with other factors such as user relevance, serendipity, and diversity.*

When we asked participants which features matter in "finding a similar movie to watch next", the highest rated features were all content-specific (genre, mood, plot, and theme) followed by features related to quality (average rating and critical reception). Beyond these, preferences seem to be highly personal: all factors in our survey question attracted at least a few "strongly agree" responses, and we collected a wide range of open-ended responses, from budget to political orientation. The design implication from this data is to *prioritize content relevance and quality in related item recommendations*, and perhaps to allow some form of direct control over the recommendations, as was suggested by several participants.

7 LIMITATIONS AND FUTURE WORK

This research examines related item algorithms in the context of movies. Some of our findings may be domain-specific, and we hope future user-centric work looks at other domains.

We built and evaluated algorithms that compute similarity, but we have not tested personalized ranking algorithms for related item recommendation. In our survey, users reported that they desire similar recommendations that fit their tastes to some degree. Therefore, we think it is likely that personalized algorithms will be able to deliver better recommendations than the non-personalized algorithms tested here; that remains interesting future work.

It is very difficult and labor intensive to collect a dataset of similarity and recommendation quality judgments from users. In particular, our methods rely extensively on users with a breadth of domain knowledge, so that they are familiar with both the seed and the neighbor. We primarily studied the most popular (top 2,500 seed movies, top 10,000 neighbors, in terms of popularity) movies. The long tail has substantially less user activity to describe items, and often has less metadata as well. However, users indicated in our study that they are looking for non-obvious recommendations. This points to interesting future work in mining the long tail for serendipitous related item recommendations, and in understanding coverage of current techniques.

8 CONCLUSION

In a six-way, human-judged comparison of similar item algorithms, we find that algorithms based on content (free text and tags) outperform algorithms based on ratings and clickstream data, both in terms of similarity and recommendation quality. We conducted a survey of more than 700 users of a movie recommender, finding that people think of content descriptions (e.g., genre, plot) and quality features (e.g., average rating) as most important to whether a related movie is a good recommendation. We also find that that recommender systems should avoid surfacing recommendations that are too "obvious". Finally, we find that users consider related item recommendations to be extremely important relative to other recommendation features.

We have openly released our dataset of survey responses and similarity judgments to facilitate more research in this area. We hope this inspires more user-centric research exploring item similarity algorithms and their relationship to recommendation quality.

9 ACKNOWLEDGMENTS

We thank Saikrishna Sunkam for his technical contributions, members of GroupLens research for their support and feedback, and MovieLens members for their participation. This material is based on work supported by grants from Amazon and Google.

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Raghu Ramakrishnan. 2013. Content Recommendation on Web Portals. *Commun. ACM* 56, 6 (June 2013), 92–101. https://doi.org/10.1145/2461256.2461277

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93)*. ACM, New York, NY, USA, 207–216. https://doi.org/10.1145/170035.170072

[3] Hyung Jun Ahn. 2008. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences* 178, 1 (Jan. 2008), 37–51. https://doi.org/10.1016/j.ins.2007.07.024

[4] Kamal Ali and Wijnand van Stam. 2004. TiVo: Making Show Recommendations Using a Distributed Collaborative Filtering Architecture. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*. ACM, New York, NY, USA, 394–401. https://doi.org/10.1145/1014052.1014097

[5] Paulo J. Azevedo and Alípio M. Jorge. 2007. Comparing Rule Measures for Predictive Association Rules. In *Machine Learning: ECML 2007 (Lecture Notes in Computer Science)*. Springer, Berlin, Heidelberg, 510–517. https://doi.org/10.1007/978-3-540-74958-5_47

[6] O. Barkan and N. Koenigstein. 2016. ITEM2VEC: Neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. 1–6. https://doi.org/10.1109/MLSP.2016.7738886

[7] Bert Boyce. 1982. Beyond topicality: A two stage view of relevance and the retrieval process. *Information Processing & Management* 18, 3 (Jan. 1982), 105–109. https://doi.org/10.1016/0306-4573(82)90033-4

[8] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. 1997. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD '97)*. ACM, New York, NY, USA, 255–264. https://doi.org/10.1145/253260.253325

[9] Robin Burke and Maryam Ramezani. 2011. Matching Recommendation Technologies and Domains. In *Recommender Systems Handbook*. Springer, Boston, MA, 367–386. https://doi.org/10.1007/978-0-387-85820-3_11

[10] R. H. B. Christensen. 2018. ordinal—Regression Models for Ordinal Data. https://CRAN.R-project.org/package=ordinal R package version 2018.4-19.

[11] Lucas Colucci, Prachi Doshi, Kun-Lin Lee, Jiajie Liang, Yin Lin, Ishan Vashishtha, Jia Zhang, and Alvin Jude. 2016. Evaluating Item-Item Similarity Algorithms for Movies. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. ACM, New York, NY, USA, 2141–2147. https://doi.org/10.1145/2851581.2892362

[12] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 191–198. https://doi.org/10.1145/2959100.2959190

[13] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube Video Recommendation System. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 293–296. https://doi.org/10.1145/1864708.1864770

[14] Christian Desrosiers and George Karypis. 2011. A Comprehensive Survey of Neighborhood-based Recommendation Methods. In *Recommender Systems Handbook*. Springer, Boston, MA, 107–144. https://doi.org/10.1007/978-0-387-85820-3_4

[15] J. Stephen Downie, Andreas F. Ehmann, Mert Bay, and M. Cameron Jones. 2010. The Music Information Retrieval Evaluation eXchange: Some Observations and Insights. In *Advances in Music Information Retrieval*. Springer, Berlin, Heidelberg, 93–115. https://doi.org/10.1007/978-3-642-11674-2_5

[16] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*. 263–272. https://doi.org/10.1109/ICDM.2008.22

[17] J. Katukuri, T. Könik, R. Mukherjee, and S. Kolay. 2014. Recommending similar items in large-scale online marketplaces. In *2014 IEEE International Conference on Big Data (Big Data)*. 868–876. https://doi.org/10.1109/BigData.2014.7004317

[18] Peter Knees and Markus Schedl. 2013. A Survey of Music Similarity and Recommendation from Music Context Data. *ACM Trans. Multimedia Comput. Commun. Appl.* 10, 1 (Dec. 2013), 2:1–2:21. https://doi.org/10.1145/2542205.2542206

[19] Bart P. Knijnenburg, Martijn C. Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the User Experience of Recommender Systems. *User Modeling and User-Adapted Interaction* 22, 4-5 (Oct. 2012), 441–504. https://doi.org/10.1007/s11257-011-9118-4

[20] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37. https://doi.org/10.1109/MC.2009.263

[21] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *arXiv:1405.4053 [cs]* (May 2014). arXiv: 1405.4053.

[22] Jin Ha Lee. 2010. Crowdsourcing music similarity judgments using mechanical turk. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*.

[23] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.Com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (Jan. 2003), 76–80. https://doi.org/10.1109/MIC.2003.1167344

[24] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook*. Springer, Boston, MA, 73–105. https://doi.org/10.1007/978-0-387-85820-3_3

[25] Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06)*. ACM, New York, NY, USA, 1097–1101. https://doi.org/10.1145/1125451.1125659

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*. Curran Associates Inc., USA, 3111–3119.

[27] Peter Organisciak and J. Stephen Downie. 2015. Improving Consistency of Crowdsourced Multimedia Similarity for Evaluation. In *Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '15)*. ACM, New York, NY, USA, 115–118. https://doi.org/10.1145/2756406.2756942

[28] Parivash Pirasteh, Jason J. Jung, and Dosam Hwang. 2014. Item-Based Collaborative Filtering with Attribute Correlation: A Case Study on Movie Recommendation. In *Intelligent Information and Database Systems (Lecture Notes in Computer Science)*. Springer, Cham, 245–252. https://doi.org/10.1007/978-3-319-05458-2_26

[29] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. 2002. Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI '02)*. ACM, New York, NY, USA, 127–134. https://doi.org/10.1145/502716.502737

[30] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2000. *Application of Dimensionality Reduction in Recommender System - A Case Study.* Technical Report TR-00-043. University of Minnesota.

[31] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the Tenth International World Wide Web Conference*. Hong Kong.

[32] J. Ben Schafer, Joseph Konstan, and John Riedl. 1999. Recommender Systems in e-Commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce (EC '99)*. ACM, New York, NY, USA, 158–166. https://doi.org/10.1145/336992.337035

[33] Markus Schedl, Peter Knees, and Fabien Gouyon. 2017. New Paths in Music Recommender Systems Research. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, New York, NY, USA, 392–393. https://doi.org/10.1145/3109859.3109934

[34] Shilad Sen, Jesse Vig, and John Riedl. 2009. Learning to Recognize Valuable Tags. In *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI '09)*. ACM, New York, NY, USA, 87–96. https://doi.org/10.1145/1502650.1502666

[35] Shilad Sen, Jesse Vig, and John Riedl. 2009. Tagommenders: Connecting Users to Items Through Tags. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*. ACM, New York, NY, USA, 671–680. https://doi.org/10.1145/1526709.1526800

[36] B. Smith and G. Linden. 2017. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing* 21, 3 (June 2017), 12–18. doi.ieeecomputersociety.org/10.1109/MIC.2017.72

[37] R. Typke, R. C. Veltkamp, and F. Wiering. 2006. A Measure for Evaluating Retrieval Techniques based on Partially Ordered Ground Truth Lists. In *2006 IEEE International Conference on Multimedia and Expo*. 1793–1796. https://doi.org/10.1109/ICME.2006.262900

[38] Jesse Vig, Shilad Sen, and John Riedl. 2012. The Tag Genome: Encoding Community Knowledge to Support Novel Interaction. *ACM Trans. Interact. Intell. Syst.* 2, 3 (Sept. 2012), 13:1–13:44. https://doi.org/10.1145/2362394.2362395

[39] C. Wang, A. Agrawal, X. Li, T. Makkad, E. Veljee, O. Mengshoel, and A. Jude. 2017. Content-based top-N recommendations with perceived similarity. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 1052–1057. https://doi.org/10.1109/SMC.2017.8122750

[40] Chun Zeng, Chun-Xiao Xing, and Li-Zhu Zhou. 2003. Similarity Measure and Instance Selection for Collaborative Filtering. In *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*. ACM, New York, NY, USA, 652–658. https://doi.org/10.1145/775152.775243

[41] Renjie Zhou, Samamon Khemmarat, and Lixin Gao. 2010. The Impact of YouTube Recommendation System on Video Views. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*. ACM, New York, NY, USA, 404–410. https://doi.org/10.1145/1879141.1879193

[42] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50.