A Framework for Dataset Benchmarking and its Application to a New Movie Rating Dataset

SIMON DOOMS, Ghent University, Belgium
ALEJANDRO BELLOGIN, Centrum Wiskunde & Informatica, The Netherlands
TOON DE PESSEMIER, iMinds-Ghent University, Belgium
LUC MARTENS, iMinds-Ghent University, Belgium

Rating datasets are of paramount importance in recommender systems research. They serve as input for recommendation algorithms, as simulation data, or for evaluation purposes. In the past, public accessible rating datasets were not abundantly available, leaving researchers no choice but to work with old and static datasets like MovieLens and Netflix. More recently, however, emerging trends as social media and smartphones are found to provide rich data sources which can be turned into valuable research datasets. While dataset availability is growing, a structured way for introducing and comparing new datasets is currently still lacking. In this work, we propose a five-step framework to introduce and benchmark new datasets in the recommender systems domain. We illustrate our framework on a new movie rating dataset – called MovieTweetings – collected from Twitter. Following our framework, we detail the origin of the dataset, provide basic descriptive statistics, investigate external validity, report the results of a number of reproducible benchmarks, and conclude by discussing some interesting advantages and appropriate research use cases.

Categories and Subject Descriptors: H.1.2 [**Information Systems**]: Models and Principles—*User/Machine Systems, Human Information Processing*

General Terms: Algorithms, Experimentation, Human Factors

Additional Key Words and Phrases: benchmark, dataset, evaluation, reproducibility, MovieTweetings, IMDb, Twitter, MovieLens

ACM Reference Format:
Simon Dooms, Alejandro Bellogín, Toon de Pessemier, and Luc Martens, 2014. A Framework for Dataset Benchmarking and its Application to a New Movie Rating Dataset. *ACM Trans. Embedd. Comput. Syst.* V, N, Article A (January YYYY), 29 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Recommender systems need input data to drive their decision making process and generate recommendations for users. The (perceived) quality of the recommendations in the end does not solely rely on the recommendation algorithms, but also greatly depends on such provided input data, in combination with other factors such as the

This work is funded by a PhD grant to Simon Dooms of the Agency for Innovation by Science and Technology (IWT Vlaanderen) and the Spanish Ministry of Science and Innovation (TIN2013-47090-C3-2). Part of this work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme, funded by European Comission FP7 grant agreement no.246016. The experiments in this work were carried out using the Stevin Supercomputer Infrastructure at Ghent University, funded by Ghent University, the Hercules Foundation and the Flemish Government - department EWI.
Author's addresses:

DOI: http://dx.doi.org/10.1145/0000000.0000000

interface, the explanations provided to the user, and trust in the technology [Knijnenburg et al. 2012]. Without input, a recommender is just an engine without fuel and so the selection of the input data is of paramount importance to any recommender system. Because input data (e.g., typically user ratings) are hard to find and difficult to collect, most research on recommender systems relies on publicly available datasets.

Two of the most popular datasets are the MovieLens [Herlocker et al. 1999; Bobadilla et al. 2010; Peralta 2007] and the Netflix [Bennett and Lanning 2007; Töscher et al. 2009; Piotte and Chabbert 2009; Koren 2009] datasets, both focusing on the movie domain. The first MovieLens dataset (ML 100K) was released at the end of the 90's and integrated 100,000 ratings originating from users of the MovieLens system covering a seven-month period (September 19th, 1997 through April 22nd, 1998). This dataset quickly became very popular because of its simplicity (i.e., only explicit ratings and very basic item and user data were available) and the lack of other datasets at that time. Later in 2006, the Netflix dataset originated from the well-known Netflix prize[1], where 1 million dollar was promised to the first team able to improve the in-house Netflix recommendation algorithm by more than 10%. In the last few years, dozens of other datasets have become available focusing sometimes on very divergent item domains (e.g., jokes, like the Jester dataset[2]), specific meta-data availability (e.g., contextual information, like the datasets produced in the different editions of the CAMRA challenge[3]) or social network information (e.g., cross-domain ratings from Twitter [Dooms et al. 2014]).

With more datasets becoming available, the need rises for a structured way to introduce a new dataset. Currently datasets are mostly published alongside some minimal documentation that details only the data itself and its structural properties. Ideally the introduction of a new dataset should allow researchers to quickly assess the value of the new data and interpret its comparability to other existing datasets.

In this work, we propose a five-step framework to introduce and benchmark new datasets in the recommender systems research domain. We illustrate this framework by applying it to a new movie rating dataset called MovieTweetings. This dataset contains explicit movie ratings, originating from the *Internet Movie Database (IMDb)*[4], provided on a 10-star rating scale and basic movie information data (i.e., genres) in a similar format as the MovieLens dataset. An extended version of the dataset has been the topic of the ACM RecSys Challenge 2014[5] which focused on user engagement as evaluation. One of the unique properties of the MovieTweetings dataset is that its ratings are frequently updated and therefore guaranteed to include the most recent and popular movies. It may be hard however to convince researchers to adopt such a new dataset instead of an already available and well-known dataset. With the application of our framework, we aim to illustrate how datasets can be introduced while presenting researchers with a complete analysis and some insight into its useful applications and scenarios.

The remainder of this work is structured as follows. Section 2 details our framework focusing on the introduction and benchmarking of new datasets. In Section 3 we apply such framework to the MovieTweetings dataset and provide insights on the origin of the dataset and its external validity; then, in Section 4 we present benchmarking results, and in Section 5 we discuss some of the most appropriate use cases for this

[1]More information at http://www.netflixprize.com.
[2]Available at http://shadow.ieor.berkeley.edu/humor.
[3]More information at http://2012.recsyschallenge.com/tracks/camra/.
[4]See http://www.imdb.com.
[5]See http://2014.recsyschallenge.com.

dataset. Finally, the results obtained in this work and their relevance towards future research are presented in Section 6.

2. A DATASET BENCHMARKING FRAMEWORK

Here we propose a five-step framework for introducing and benchmarking new datasets; this framework is especially tailored to and tested in the recommender systems research domain, but we expect it to be useful in related areas – such as user modeling, personalized search, etc. –, and hence, it could show a wider applicability in the future. In the following, we provide for each step some explanation and recommended procedures for further analysis.

(1) **Dataset Story.** This step introduces the item domain, the origin of the dataset, and its collection procedure. Understanding the way in which the dataset was collected can be very important for future researchers who e.g., discover biases in the dataset. For example, an e-commerce website may generate more data during the holiday season, or at some point during the data collection, a user interface may have changed, resulting in different observed user behavior (and collected user data). It is therefore important to carefully document the item domain (and its known characteristics) and the dataset collection procedure.

(2) **Descriptive Statistics.** Here we list a number of basic statistics of the dataset. This provides researchers with a sense of dataset size, its sparsity, etc. Statistics from other similar datasets can also be presented to allow comparison. For datasets relevant to the recommender systems domain we propose to document at least the following statistics: number of users, number of items, number of ratings, sparsity (or density). These statistics are best compared with existing similar datasets in the same domain (if available) and can be complemented with other metrics that are relevant to illustrate the difference (or similarity). See Table I for an example of descriptive statistics applied to the MovieTweetings dataset.

(3) **External Validity.** The goal of this step is to inspect the external validity of the dataset. It can start out with clearly defining a number of hypothesized biases that can then be confirmed or rejected. These biases can be analyzed in various ways; for the MovieTweetings dataset we employ a correlation analysis methodology: the average ratings are correlated with rating statistics from IMDb and compared with those from the MovieLens dataset; nonetheless, a similar analysis could be derived if no ratings are available. A popularity study can further help in identifying the major differences and similarities with other datasets. Comparing the dataset with other similar datasets may seem trivial at first, but the difficulty lies in uncovering the relevant differences in rating behavior, or user interactions in general. A simple count of the number of users, items and ratings may not be sufficient. We used rating histograms and plotted various rating distributions to compare the fundamental differences in item popularity between MovieTweetings and MovieLens; in the general case where ratings are not available, histograms based on (user, item) occurrences may reveal behavior trends in a similar way as the rating histograms.

(4) **Reproducible Benchmarks.** In this step, well-known recommendation algorithms would run using publicly available software libraries. Results under varying conditions (e.g., data splitting methodologies and evaluation metrics) would then be reported and compared against those found for other datasets. The goal here is to evaluate the baseline performance of a particular dataset, by testing it using well-known recommendation algorithms. Iteratively repeating the same experiments with increasingly growing slices of the dataset furthermore allows to inspect the stability and consistency of the results. Performance results can be measured by a wide variety of evaluation metrics commonly adopted in the recom-

mender systems literature [Shani and Gunawardana 2011] and considering additional contexts [Campos et al. 2014]. Also the evaluation methodology is a variable that can be tested, although the focus should be on consistency, and hence the same methodology should be applied to all the experiments, otherwise the results will not be comparable [Said and Bellogín 2014]. If possible, such methodology should be as close to the real life scenario where the dataset will be used as possible.
An important aspect here is that publicly available software libraries are used, and their parameters are clearly reported to allow for experimental reproducibility [Bellogín et al. 2013]. In this work, we use MyMediaLite in combination with the *trec_eval* program, but there are many other options available (e.g., the Rival toolkit[6] for recommender systems evaluation [Said and Bellogín 2014]). We experimented with a wide variety of experimental conditions: 5 recommendation algorithms, 2 data splitting methodologies and multiple evaluation metrics (both rank-based and error-based). The results should be compared with other similar datasets in the same item domain to find out if the analyzed dataset is as difficult (or easy) as the other datasets. We believe this aspect can remain unnoticed even after the aforementioned analyses, since the datasets may be similar from a global point of view, but in a user (or item) basis, the scarcity (or abundance) of data, or their distributions, may complicate (or favor) some recommendation methods over others, evidencing an inherent difference in the data patterns. An extreme example would be a dataset where an item average recommender achieves a very good performance, indicating that the most important signal is the item's average rating, no matter how different the dataset is to previous datasets in terms of distributions and popularity correlations. The nature of these differences may arise, in fact, from outside of the dataset, external factors such as the interface of the original system the ratings were collected from or the main goal the user is aiming to satisfy with the system.

(5) **Use Cases & Advantages.** Finally, it would be a good practice to indicate the intended use cases for the dataset and its advantages against other already available datasets. The use cases can be supported by a number of experiments that illustrate why the dataset may be a better choice in certain situations.

3. INTRODUCING THE MOVIETWEETINGS DATASET

We now illustrate our proposed benchmarking framework on a new movie rating dataset called MovieTweetings. In this section, we detail how it was collected, present some descriptive statistics and discuss external validity.

3.1. Dataset Story

While searching the Internet for structured movie preference information, to our own surprise, we discovered a vast number of explicit movie ratings from IMDb being posted on Twitter by means of a social *sharing* feature. This notion of *social sharing* is increasingly becoming more popular on the Internet. Websites offer promote buttons that aid users in posting interesting content (i.e., often the page the user is currently browsing) directly to their social network. When such a button is clicked, a user may add some additional comments before finally submitting. The content provider usually already provides a suggestion (template) for what a user might post, for instance, the title of the page together with the url and some reference to the social network account of the website itself (illustrated in Fig. 1).

[6]Available at http://rival.recommenders.net

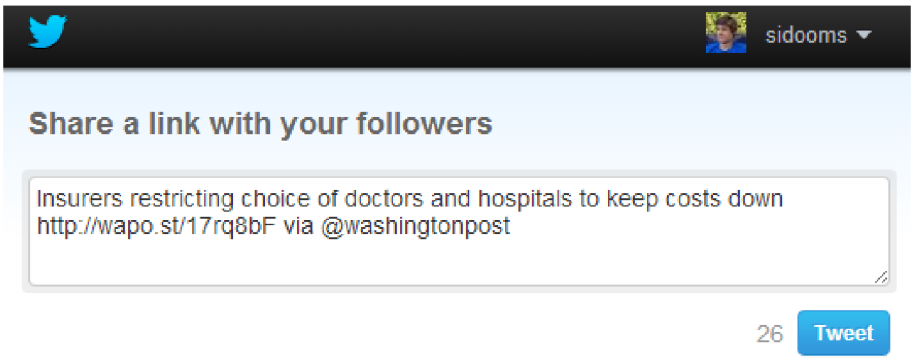

Fig. 1. Illustration of the social sharing feature that is included in many websites (here: *washingtonpost.com*). By clicking a *share* button, the content provider provides a suggestion as to what the user might post to its social network (here: *twitter.com*).

We have found that one of the mobile apps from the IMDb platform[7], after a movie has been rated, offers a well-structured template to post to Twitter (Fig. 2). When a user rates a movie on the iOS mobile app, an option is available to 'Share my rating'. When enabled, the user is taken to a screen that proposes to post the following text (for the movie 'Man of Steel'):

```
I rated Man of Steel 10/10 #IMDb
```

This pre-formatted tweet is well-structured and therefore apt for automated extraction. In the tweet we find the title of the movie, the rating, and a website-specific hashtag. The hashtag allows for easy filtering tweets originating from IMDb. When the tweet is posted, a link to the IMDb page of the involved movie is inserted as follows:

```
I rated Man of Steel http://www.imdb.com/title/tt0770828 10/10 #IMDb
```

From this link, the IMDb id of the movie can be extracted which allows us to unambiguously identify the movie that is rated in the tweet (which is not always possible using only the movie title). While searching Twitter can lead to many ambiguous results, we can use the proposed fixed format of the tweet to our advantage. Instead of searching for the movie title, we use the query 'I rated #IMDb' and then apply string matching techniques to extract the relevant fields from the returned tweets. Specifically, we extract the following fields from the tweets:

— Twitter user id
— IMDb movie id
— Rating
— Timestamp

Furthermore, we also extract additional genre data from the IMDb page of the movie rated in the tweet. Such additional genre data can be exploited by content-based recommenders who would use movie attributes in the recommendation process [Lops et al. 2011]. Even more content attributes can be downloaded by extracting them directly from the corresponding IMDb page (whose URL can be reconstructed directly from the IMDb movie id by adding "`http://www.imdb.com/title/tt`"). Sometimes users tend to extend the default tweet (up to the limit of 140 characters imposed

[7]Available at http://www.imdb.com/apps.

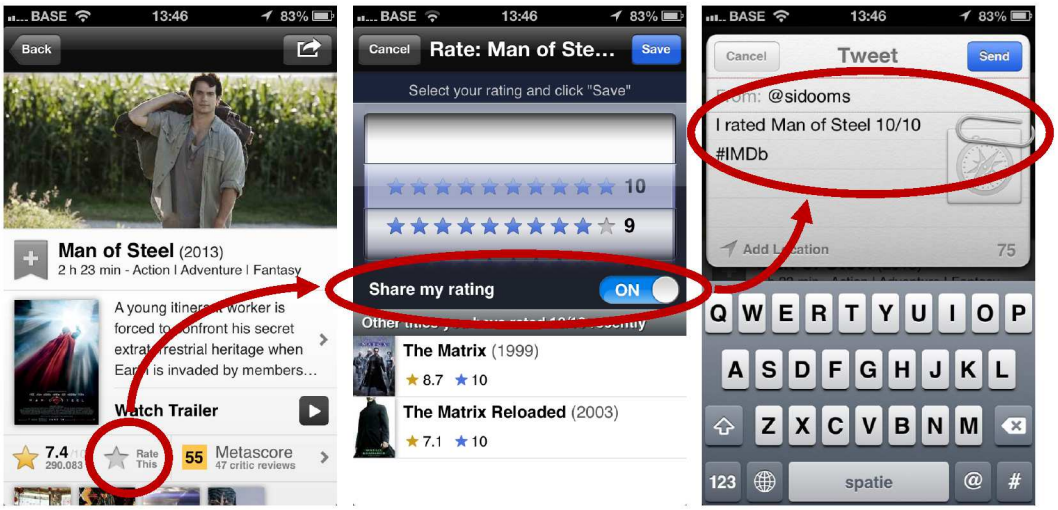

Fig. 2. Screenshots from the iPhone mobile IMDb app illustrating the process of rating a movie and sharing it on Twitter.

by Twitter), proposed by the IMDb app, with their own personal opinions or additional comments. For our dataset, however, we do not integrate these additional comments which would require Natural Language Processing [Yi et al. 2003] and might introduce more noise than information. Instead, we focus on the available explicit feedback, i.e., the numerical rating. Starting March 7, 2013 we queried the Twitter search API on a daily basis and extracted ratings from relevant tweets into our dataset. The dataset itself consists of three files: *ratings.dat*, *users.dat* and *movies.dat*, which are formatted similarly as the popular MovieLens dataset to facilitate integration in existing implementations:

— **ratings.dat** contains the ratings as tuples, together with the user, movie and corresponding timestamp. The user id is an internal numerical identifier, while the movie id is the IMDb id. Ratings range from 1 to 10 in the analyzed dataset[8].
— **users.dat** provides a link between the internal user ids and the true Twitter id of the user, allowing for additional data enrichment.
— **movies.dat** lists the movies that were rated at least once, together with the movie year and genre data, from IMDb.

The dataset offers two repositories[9]: latest data and snapshots. Latest data always contains all the data in the dataset, including the most recently added data. This repository will therefore be subject to frequent updates. The snapshots, on the other hand, offer fixed portions of the dataset in various sizes (e.g., 10K, 100K, 150K) to stimulate and facilitate experimental reproducibility. The dataset license is MIT[10], which allows both academic researchers and industry to benefit from the data.

[8]The dataset does include a few 0/10 ratings, but they are originating from users that manually changed the rating value to 0 in the template tweet. These rating values will be ignored in this work.
[9]Available at https://github.com/sidooms/MovieTweetings.
[10]Available at http://opensource.org/licenses/MIT.

Table I. Basic characteristics dataset comparison

Metric	**MT 100K**	ML 100K	ML 10M100K*
# Users	16,554	943	2,059
# Users with at least 5 ratings	4,692	943	1,597
# Users with at least 10 ratings	2,583	943	1,244
# Users with at least 20 ratings	1,154	943	932
# Items	10,506	1,682	6,930
# Ratings	100,000	100,000	100,000
Density	0.06 %	6.3 %	0.7 %
Minimum # ratings per user	1	20	1
Average # ratings per user	6	106	49
Maximum # ratings per user	320	737	1,563
Minimum # ratings per item	1	1	1
Average # ratings per item	10	59	14
Maximum # ratings per item	1,812	583	529
Earliest rating	Feb 28, 2013	Sep 20, 1997	Nov 22, 2008
Latest rating	Sep 1, 2013	Apr 22, 1998	Jan 5, 2009
Time span	186 days	214 days	44 days

3.2. Descriptive Statistics

We now present some descriptive statistics of the MovieTweetings dataset. In Table I we list some basic numerical characteristics and compare them with other known datasets for reference. As reference dataset we use MovieLens because it is the most popular movie rating dataset in the recommender systems domain and is still publicly available[11] (the Netflix dataset is not anymore due to privacy issues). Both MovieTweetings (MT) and MovieLens (ML) have a dataset snapshot containing 100K ratings, so these are obvious candidates for a comparison. MovieTweetings is however a natural, unfiltered dataset, i.e., no users and items have been omitted, while in the MovieLens 100K dataset only users with at least 20 ratings are included. To increase the comparability of the datasets, we generated a new MovieLens dataset based on the data of MovieLens 10M. We sliced the (chronologically) last 100K ratings of this bigger dataset, taking into account the rating timestamps. While in MovieLens 10M all users have rated at least 20 movies, for its 100K subset, this will no longer be true, making the comparison with MovieTweetings more fair. We will refer to this dataset as the *ML 10M100K** dataset, in order to distinguish it from the alternative name of the Movielens 10M dataset (ML 10M100K) that considers the fact that such dataset contains around 100K tag assignments. Wherever possible, we also include the standard MovieLens 100K in the comparative experiments as another baseline.

The density metric is calculated according to the following equation:

$$rating\ density = 100 \times \frac{\#\ available\ ratings}{\#\ all\ possible\ ratings} = 100 \times \frac{\#\ available\ ratings}{(\#\ users) \times (\#\ items)} \quad (1)$$

The numbers in Table I clearly indicate one of the major aspects that differentiates the MovieTweetings dataset: its low density. The number of items and users is much higher (almost by a factor of 10) for MovieTweetings compared to the other datasets. While most rating datasets are restricted to users of the particular closed system (e.g., the Netflix or MovieLens system), the MovieTweetings dataset integrates data from users of the IMDb platform, which is very popular and open to anyone on the Internet. Also in terms of number of items, the IMDb catalog is not restricted to movies that can be rented but rather includes almost all existing movies. Integrating the IMDb

[11] At http://grouplens.org/datasets/movielens.

platform as rating source, therefore leads to very high numbers of distinct users and items, leading to a density value closer to 0 than for the other datasets.

Table I also lists time information regarding the included ratings for each of the datasets. We can clearly see how the MovieLens 100K is the oldest dataset with ratings at least 15 years older than the MovieTweetings ratings. The artificially created MovieLens ML 10M100K* dataset contains more recent ratings (2008-2009), but is still older than MT. In terms of time span however, ML 100K and MT 100K are roughly the same. The main conclusion we can draw from the presented descriptive statistics is that although datasets are hard to compare because of their many differences, we can increase comparability by using multiple (samples of) datasets and if necessary create artificial (or derived) ones.

3.3. External Validity

In this subsection we explore the external validity of the MovieTweetings dataset by investigating some possible biases and more thoroughly comparing the dataset with other known rating datasets in the domain.

The MovieTweetings dataset may be subject to biases introduced by the manner in which it is collected and in the sampling inherent to the Twitter API [Morstatter et al. 2013]. To understand the consequence of working with the MovieTweetings dataset, it is important to first understand these biases and study the extent of their impact. The ratings in the dataset come from users of the IMDb platform that rate movies using the mobile IMDb app for iOS and post the rating (publicly) to Twitter. One obvious bias to consider here is the effect of the ratings being collected through Twitter and not directly from IMDb itself. Only data from Twitter users with a public profile that chose to share their IMDb rating is included. Maybe these users are sharing their high and low ratings, but consider the mid-range ratings – such as 6/10 – not interesting enough to be posted to their social network. Another more structural bias comes from the fact that the dataset focuses on the IMDb platform. This platform offers a broad selection of movie information, but not the movies itself. Apart from the ability to watch the trailer of a movie, users can not watch movies on the IMDb site before rating them, something possible in a system like Netflix. Moreover, the rating scale may also affect the users [Gena et al. 2011], since the IMDb platform allows to rate movies using a 10-star feedback system, instead of the more common 5-star feedback system (like MovieLens).

3.3.1. Twitter Bias. To study the possible rating bias introduced by including only Twitter ratings, we would need to compare the ratings of the entire IMDb platform with the subset of the ratings extracted through Twitter. For this purpose a correlation study seems appropriate. The complete set of IMDb ratings (per user) is not available but the platform does provide the aggregated average movie scores. Unfortunately, IMDb does not disclose its exact averaging formula. On their website they claim to apply various filters to eliminate the effect of fake ratings by individuals who are trying to distort the aggregated rating of a movie. Therefore in our correlation analysis we have to take at least a small amount of noise into account (i.e., perfect correlation will not be possible).

For every movie that was included in the MovieTweetings (100K snapshot) dataset, we used the OMDb API[12] to obtain the aggregated average IMDb rating. We then compared this average value with the average of the MovieTweetings ratings for that movie in a correlation analysis. Fig. 3 (left) shows the resulting scatter plot, visualizing every movie with its aggregated IMDb rating (on the Y-axis) and the corresponding average MovieTweetings rating (on the X-axis). The resulting Pearson correlation value

[12]Available at http://www.omdbapi.com.

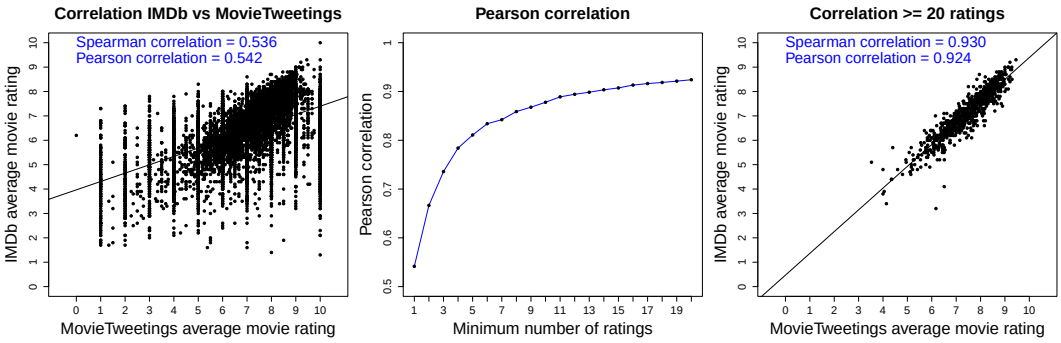

Fig. 3. Plots illustrating the correlation of the aggregated IMDb rating and the averaged MovieTweetings rating per movie. The figure on the left shows the correlation for all movies in the MovieTweetings dataset. In the middle, the Pearson correlation values are shown for movie subsets having a minimum of 1-20 ratings. The scatter plot on the right illustrates the absence of noise for movies with a minimum of 20 ratings.

is 0.542, which indicates a positive correlation. When we inspect the scatter plot, we find this confirmed, in particular for the higher rating values (higher than 6). The figure however also shows a significant amount of noise, which we hypothesize originates from movies with a low number of ratings.

To experiment with the effect of the rating frequency of the movies, we repeated the correlation analysis for different subsets of the MovieTweetings movies: we illustrate in the middle plot of Fig. 3 the increasing Pearson correlation for the movie subsets having a minimum of 2, 3, ..., 20 ratings. As expected, the harder the constraint, the more the influence of the noise is reduced, which leads to a stronger linear correlation (when 20 ratings are considered). Fig. 3 (right) plots the correlation values including only movies with a minimum of 20 ratings. Note that the correlation seems to converge to a value below the perfect correlation (i.e., of 1.0), this may be the result of not using the exact averaging formula of IMDb.

Complementary to this correlation analysis, we further compare the MovieTweetings dataset with the IMDb ratings by performing a popularity analysis for both datasets. A well-known IMDb popularity list is the top 250 movie list[13]. This list shows the best rated movies according to a Bayesian estimate defined in Equation 1:

$$weighted\ rating\ (WR) = \frac{v}{v+m} * R + \frac{m}{v+m} * C \tag{2}$$

where:

— R = average for the movie (mean) = (Rating)
— v = number of votes for the movie = (votes)
— m = minimum votes required to be listed in the Top 250 (currently 25,000)
— C = the mean vote across the whole report (currently 7.0)

So the overall popularity score takes into account the rating values, as well as the number of ratings itself to rank the movies. We adopted this as the definition of popularity and implemented the formula to rank the movies in the MovieTweetings dataset. For the same set of movies we also calculated the IMDb popularity value using the IMDb average rating values. Having calculated both the IMDb popularity and the MovieTweetings popularity value, the two lists can be ranked according to each

[13]Accessible at http://www.imdb.com/chart/top?ref_=nv_ch_250_4.

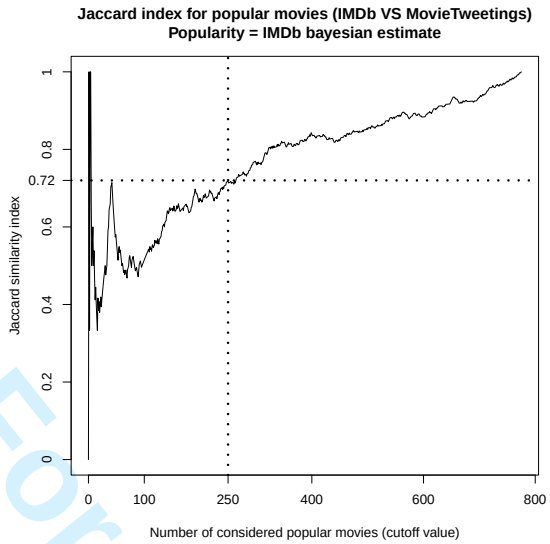

Fig. 4. The Jaccard similarity index, indicating the similarity between the top popular IMDb and MovieTweetings movies, for different cutoff lengths. Only movies with a minimum of 20 ratings are taken into account.

value and compared. For the comparison, we employ a similar method as the one used in [Bellogín et al. 2013] to compare lists of popular artists, where the Jaccard similarity index was used for different cutoff values. This is needed because standard correlation indices cannot be used, since a top-K list does not necessarily share the same items, which would make the computation of Spearman's or Pearson's correlation unfeasible. Fig. 4 shows the Jaccard index for all movies with a minimum of 20 ratings. While the top 10 popular movies are almost identical, the two datasets tend to diverge when less popular movies are considered (with another small spike at around top 50) and slowly converge to a Jaccard value of 1 at the end.

Table II shows the list of top 10 movies for both MovieTweetings and IMDb and their resulting Jaccard index values. The lists are very similar except for some small changes in the ordering. Apart from the similarity amongst these top popular movies, data show that more recent movies tend to obtain a higher overall popularity score in the MovieTweetings dataset. This is to be expected, considering how MovieTweetings collects its data from Twitter. Therefore the dataset contains more recent and currently popular movies. We expect however that with more data being collected regularly, the effect of the recentness of the movies will gradually decrease.

While the MovieTweetings dataset may favor recent movies, we hypothesize that, in general, similar trends can be noticed for IMDb. One of such trends is the type of movies that are popular, i.e., the genre. A set of genres is available for each movie allowing for a new correlation analysis, this time comparing the similarity amongst genres of popular movies for both MovieTweetings and IMDb. For this experiment we focus on the top 250 popularity list. For the 250 most popular movies we summed up the number of times each genre occurred (that is, we computed the term frequency of each genre [Baeza-Yates and Ribeiro-Neto 2011]), doing this for both datasets. Fig. 5 shows the correlation between the genre counts of the most popular MovieTweetings movies versus the ones of IMDb.

Table II. List of top 10 popular movies from IMDb vs MovieTweetings

Ranking	IMDb	MovieTweetings	Jaccard
1	The Shawshank Redemption	The Shawshank Redemption	1.00
2	The Godfather	The Dark Knight	0.33
3	The Dark Knight	The Godfather	1.00
4	Pulp Fiction	Pulp Fiction	1.00
5	The Godfather: Part II	LOTR: The Return of the King	0.67
6	The Good, the Bad and the Ugly	Terminator 2: Judgment Day	0.50
7	LOTR: The Return of the King	Schindler's List	0.56
8	Schindler's List	Forrest Gump	0.60
9	12 Angry Men	Saving Private Ryan	0.50
10	Inception	Inception	0.54

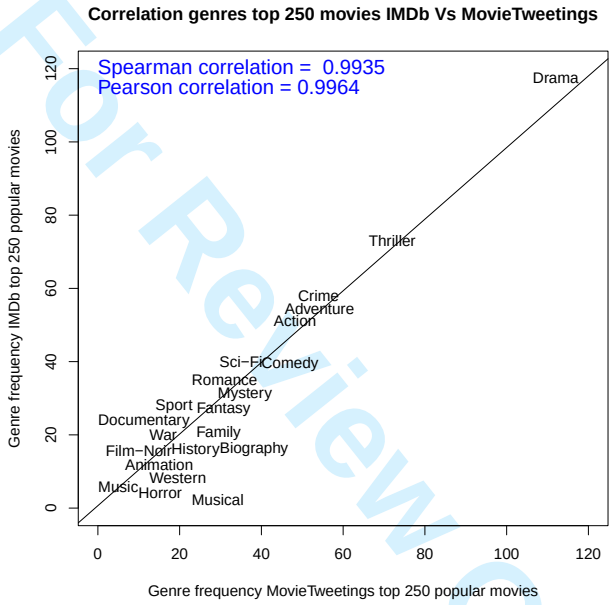

Fig. 5. The correlation of how many times each genre occurred in the top 250 movie list for MovieTweetings and IMDb.

While the Jaccard index for the top 250 popular movies was 0.72 (see Fig. 4), Fig. 5 clearly shows how highly correlated the movie genres are for both lists (Spearman correlation is 0.99). This close-to-perfect correlation indicates that although the movie lists may not be strictly identical, movies are being replaced by similar (probably more recent) movies with similar genres. For instance, in both datasets *Drama* seems to be the most popular genre, followed by *Thriller* and *Crime*.

3.3.2. IMDb bias. Another interesting approach towards investigating biases in the MovieTweetings dataset is to compare it with other popular datasets in the domain. This would allow us to verify how inherently different the ratings originating from the IMDb platform are from other movie platforms like Netflix and MovieLens. We already presented some basic statistics about the MovieTweetings dataset and compared them with the MovieLens dataset in Table I. We now provide a more thorough analysis and comparison of the dataset properties.

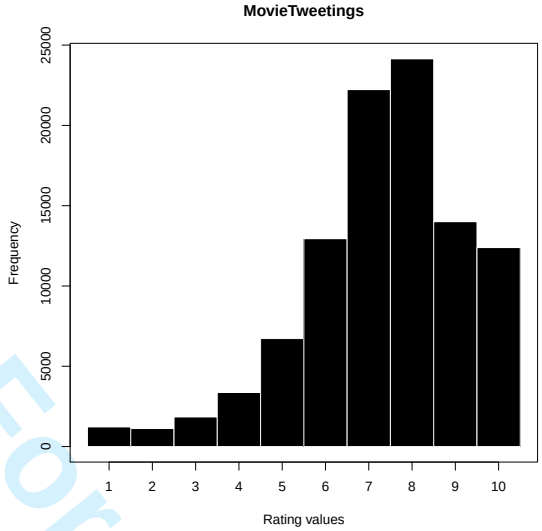

Fig. 6. Rating value histogram for the MovieTweetings dataset. The rating values are based on the IMDb 10-star rating scale.

In terms of number of ratings, the three datasets were equal (i.e., 100K ratings), nonetheless it is worth comparing the distribution of their rating values to uncover different rating behaviors. In Fig. 6 we plotted the histogram of the rating values for MovieTweetings. These values are based on the IMDb 10-star rating scale, whereas MovieLens users rated on a 5-star rating scale. The difference in rating scales makes it hard to compare the rating frequencies. Therefore, to ease the comparison with the MovieLens datasets, we paired the rating values for MovieTweetings in Fig. 7.

Every histogram displays the rating values on the X-axis and the number of times each rating value occurred in the dataset (i.e., frequency) on the Y-axis. A general trend is that the more positive rating values are more frequent in any of the three datasets. This is a well-known observation in the recommender systems research domain referred to as *not random missing data* [Marlin and Zemel 2009]. Users mostly watch movies they assume to be interesting (based on, for instance, genre, trailer, or other movie information), and therefore most movies they rate (apart from the ones they wrongfully assumed interesting) will be rated positively, which explains why generally negative/low ratings are not present in these datasets.

Although all three datasets distributions are showing a skew towards more positive rating values, the trend is significantly stronger for MovieTweetings. Only 2% of its rating values are smaller than 5 (neutral rating), compared to the MovieLens datasets where 17% for ML 100K and 11% for ML 10M100K* are lower than 3 (the counterpart neutral value in these datasets). A possible explanation for this may be the fact that users are not explicitly asked to rate movies on the IMDb platform as is the case for the MovieLens system. IMDb only recently included (basic) recommendations on its website, hence there used to be no direct incentive to rate movies other than to contribute to the aggregated IMDb rating value for a specific movie. We hypothesize that therefore users are even more skewed towards only rating exceptionally good movies, resulting in higher rating values.

Aside from the rating value distribution, it is also important to compare the general rating distribution among the datasets. In Fig. 8 the distribution of the ratings across

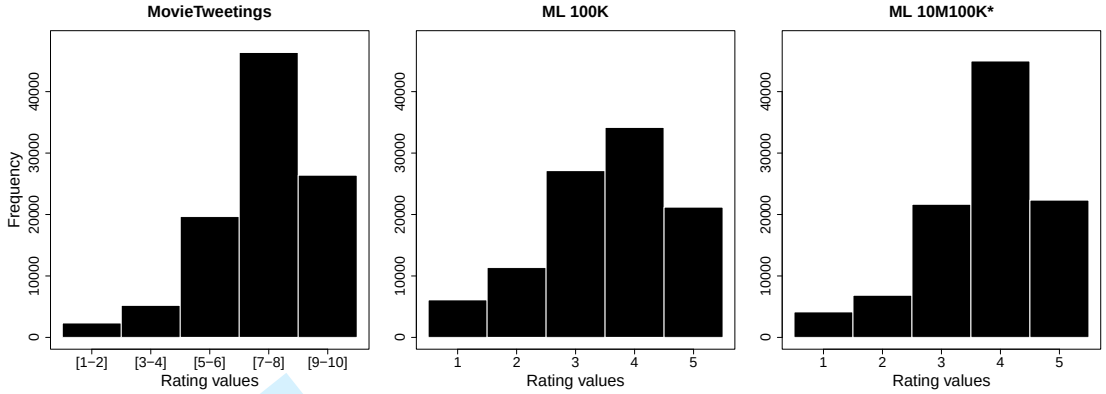

Fig. 7. Rating value histograms for the datasets MovieTweetings, ML 100K and ML 10M100K* illustrating a similar (yet stronger for MovieTweetings) shift towards positive rating values across the datasets.

the items is shown (similar to [Cremonesi et al. 2010] where Netflix and MovieLens were compared). Items are ordered by popularity (most popular at the bottom) and expressed as a percentage of the total number of items. From the figure we can see that in the case of MovieTweetings, 40% of the total amount of ratings is provided on only 1% of all the items (i.e., the 1% most popular), compared to MovieLens where this is 10%. Thus, in general the MovieTweetings dataset is less dense, but 40% of the ratings are mostly concentrating on a very small number of items which greatly increases the density for these items in particular (the top 105 most popular movies are each rated on average 380 times). The ML 10M100K* rating distribution is somewhat more similar to MovieTweetings than ML 100K is, but both are in fact still significantly different from the MovieTweetings rating distribution. Hence, MovieTweetings is more biased towards popular movies.

An important aspect about the MovieTweetings dataset is its recentness. Since ratings are mined from Twitter, there is no limitation as to how old or recent a rated movie should be. Data show however that recent movies are rated more, obviously because users tend to rate the movies they have just seen, and recent movies are more easily available (in cinemas) while being – probably – more interesting topics to share on a social network. Specifically, in Fig. 9 we plotted the year of every rated movie and its frequency for the three datasets. In general, similar patterns can be observed: a long tail with a peak at the end. The location of the peak indicates the most recent movies in the dataset, which is 2013 for the MovieTweetings dataset and towards 1998 and 2009 for the MovieLens-based datasets. The histograms here indicate how old the MovieLens datasets truly are. The MovieLens datasets include more ratings from older movies, which may still be relevant data for some use cases (e.g., recommending classic movie titles to older users). In most cases, however, users will prefer recommendations for modern and recent movies and for those situations the MovieTweetings dataset may offer an ideal way of bootstrapping a recommender system and avoid cold start issues for new users or unrated items.

Similarly as we compared the IMDb ratings with MovieTweetings, we now compare MovieTweetings with MovieLens by means of a rating correlation analysis. To be able to easily compare the rating values we rescaled the MovieTweetings ratings to a 5-star scale. For movies in MovieLens also present in the MovieTweetings dataset, we calculated the average movie rating and correlated the results. We present the results for the ML 10M100K* dataset, we found similar results for movies from ML 100K and we shall comment on this later.

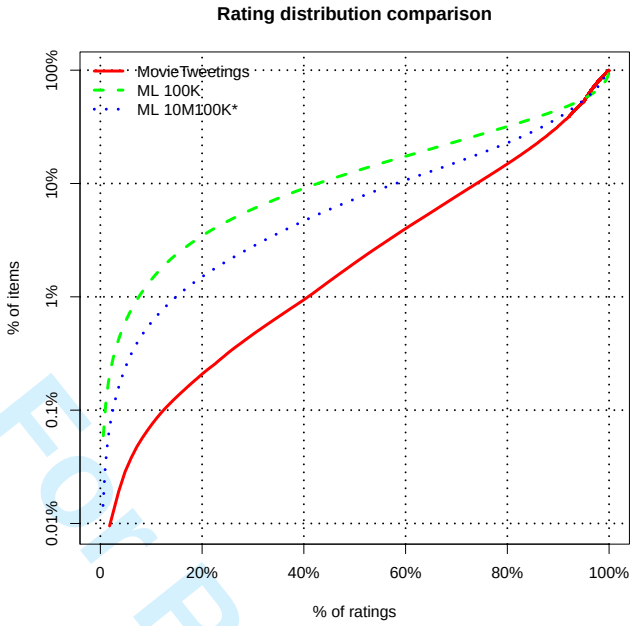

Fig. 8. Rating distribution comparison, linking the number of ratings with the number of rated items. Items are sorted according to popularity i.e., most rated, with the most popular items at the bottom. Note that the Y-axis is a log scale.

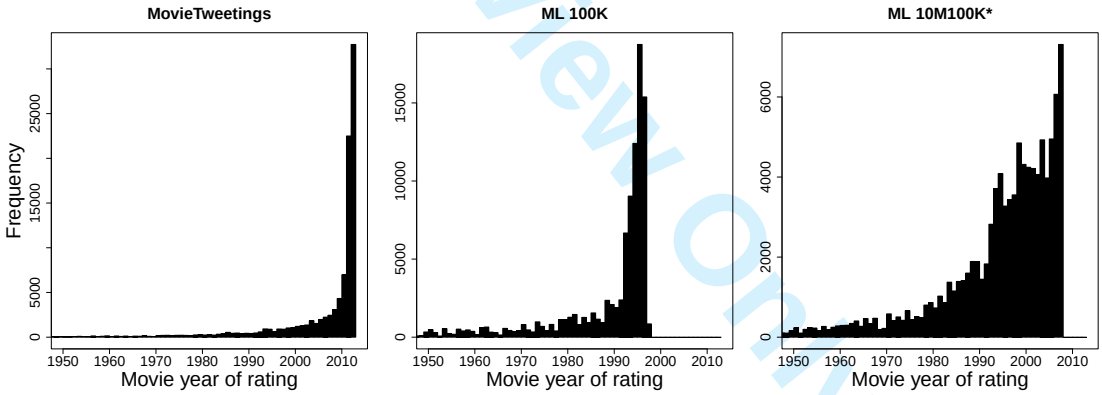

Fig. 9. Histograms illustrating how frequent movies of a given year (on the X-axis) were rated for MovieTweetings, ML 100K, and ML 10M100K*.

Fig. 10 shows the results for movies which have been rated at least 1, 2, 3, 4, 5 and 20 times. The exact Spearman and Pearson correlation values are listed in Table III. These figures show trends similar to the comparison of MovieTweetings and IMDb data. For all movies (i.e., figure for movies with >= 1 ratings) a general trend of positive correlation can be noted, while some movies show diverging rating values (i.e., dots arranged vertically on the figure). This effect decreases when we restrict the movie set to movies with a minimum of 2, 3, etc. ratings, which again confirms the diverging rating values to originate from movies which have been rated only a few times. Correlation values get stronger when increasing the minimal rating threshold

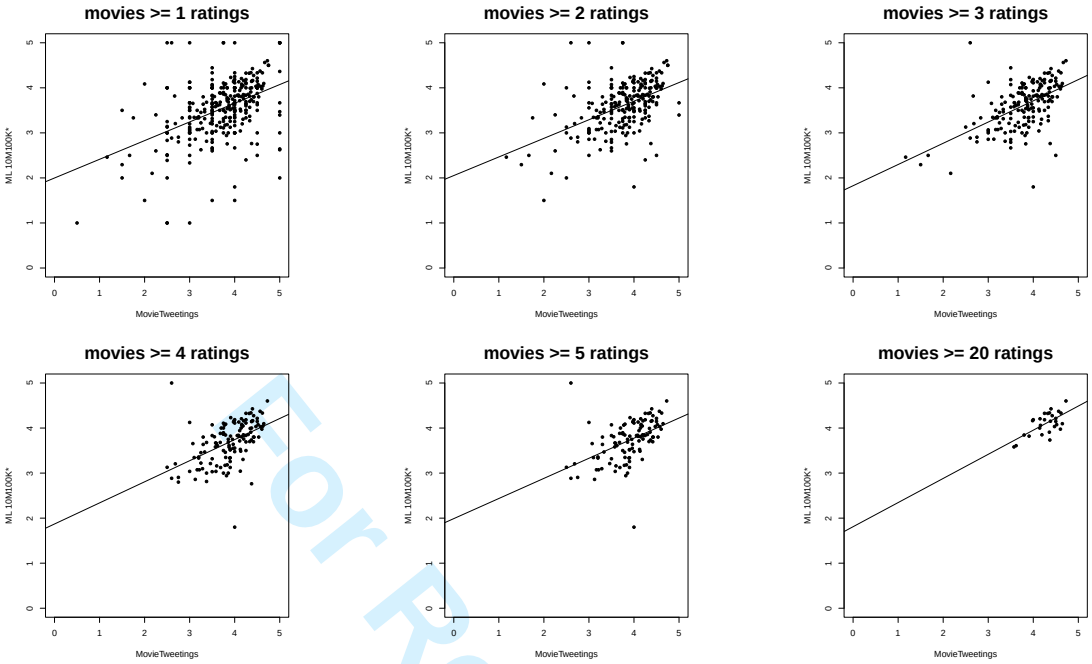

Fig. 10. Scatter plot illustrating the correlation of the average MovieTweetings rating per movie and the ML 10M100K* dataset for different subsets of movies which have been rated at least 1, 2, 3, 4, 5, 20 times. The MovieTweetings results have been rescaled to a 5-star rating scale to make the ratings more comparable.

Table III. Correlation values for average movie ratings of MovieTweetings and ML 10M100K*

Configuration	Spearman	Pearson
movies >= 1 ratings	0.425	0.449
movies >= 2 ratings	0.481	0.483
movies >= 3 ratings	0.538	0.545
movies >= 4 ratings	0.567	0.489
movies >= 5 ratings	0.580	0.475
movies >= 20 ratings	0.497	0.651

per movie, except in the last case (the subset of movies with at least 20 ratings), where there are too few movies (i.e., less than 30) with at least 20 ratings that occur at the same time in the MovieTweetings and the ML 10M100K* dataset to make a proper analysis.

Similar results were obtained for the ML 100K dataset, more specifically, ML 100K presented a slightly larger overlap with MovieTweetings in terms of movies (i.e., 80 movies with at least 20 ratings), but the overlap between these datasets was still too limited to perform a thorough popularity comparison, as we did in the previous subsection.

In summary, in this section we have addressed the external validity of the MovieTweetings rating data by investigating biases, studying its properties and comparing it against other similar datasets. With a sufficient number of minimum ratings per movie, the dataset correlates strongly with ratings found on the IMDb website, as confirmed by the popularity analysis. The dataset does however show a bias towards

very recent and popular movies, but similar trends could be noted for the MovieLens datasets.

4. BENCHMARKING THE MOVIETWEETINGS DATASET

We now analyze the MovieTweetings dataset under several conditions (data splitting, performance of recommendation methods, and evaluation metrics) and compare these results with those obtained using other datasets. By doing this, we aim to provide a (reproducible) benchmark against which other experimental variations – out of the scope of this work – can be, in the future but also retroactively, compared.

4.1. Experimental Setup

We again use two versions of the MovieLens data (i.e., *ML 100K* and *ML 10M100K**) against which we compare the MovieTweetings (MT) 100K snapshot. Note that we selected these datasets because of their similarity in both domain and properties with the MovieTweetings dataset.

We follow the evaluation methodology presented in [Koren 2008] (denoted as RPN in [Said and Bellogín 2014]), where for each user a set of not relevant items (unrated by this user in the training and testing splits) is randomly selected (100 in our case), and then, for each highly relevant item in the testing split (i.e., those rated as 5 in MovieLens or as 10 in MovieTweetings), a ranking is generated by predicting a score for this item and the other (not relevant) items. Then, the performance of this ranking is measured using the *trec_eval* program[14]. In this way, standard retrieval metrics such as precision, normalized Discounted Cumulative Gain (nDCG) or Mean Average Precision (MAP) could be used [Baeza-Yates and Ribeiro-Neto 2011]. Additionally, and for the sake of comparison with previous works, we also measured error-based metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), pervasive in the recommender systems literature [Shani and Gunawardana 2011]. We have to note that the dataset framework we present here can be instantiated using any other evaluation strategy available in the literature; in fact when the TestItems evaluation methodology from [Bellogín et al. 2011] or TrainItems from [Said and Bellogín 2014] were tested in some preliminary experiments, we found comparable results but decided to stick with RPN because it is insensitive to the sparsity bias [Bellogín 2012] – which allows proper comparisons of algorithms when the sparsity changes – even though these strategies could be understood to be closer to real life than RPN.

We have tested five recommendation algorithms as implemented in the MyMediaLite Recommender System library (version 3.10)[15]. All of them only use ratings as the basis of the predictions, two are non-personalized (user and item average) methods, and the rest do use information about the target user – either as memory-based collaborative filtering algorithms (user and item nearest neighbor) or as a model-based (matrix factorization) recommender. Table IV shows a description of these approaches, along with the default parameters for MyMediaLite version 3.10 used in our experiments. Note that some of these specifications are not standard (e.g., the cosine function as similarity, or using a baseline predictor in the nearest neighbor methods), but since they are applied to all the datasets impartially, the overall conclusions should be fair and not sensitive to this aspect. In any case, we did a preliminary test with Pearson's correlation as similarity function and the general trend remained the same, the only change was that the performance of UserKNN and ItemKNN improved slightly (but uniformly in the three datasets).

[14]Available at http://trec.nist.gov/trec_eval.
[15]Available for download at http://mymedialite.net.

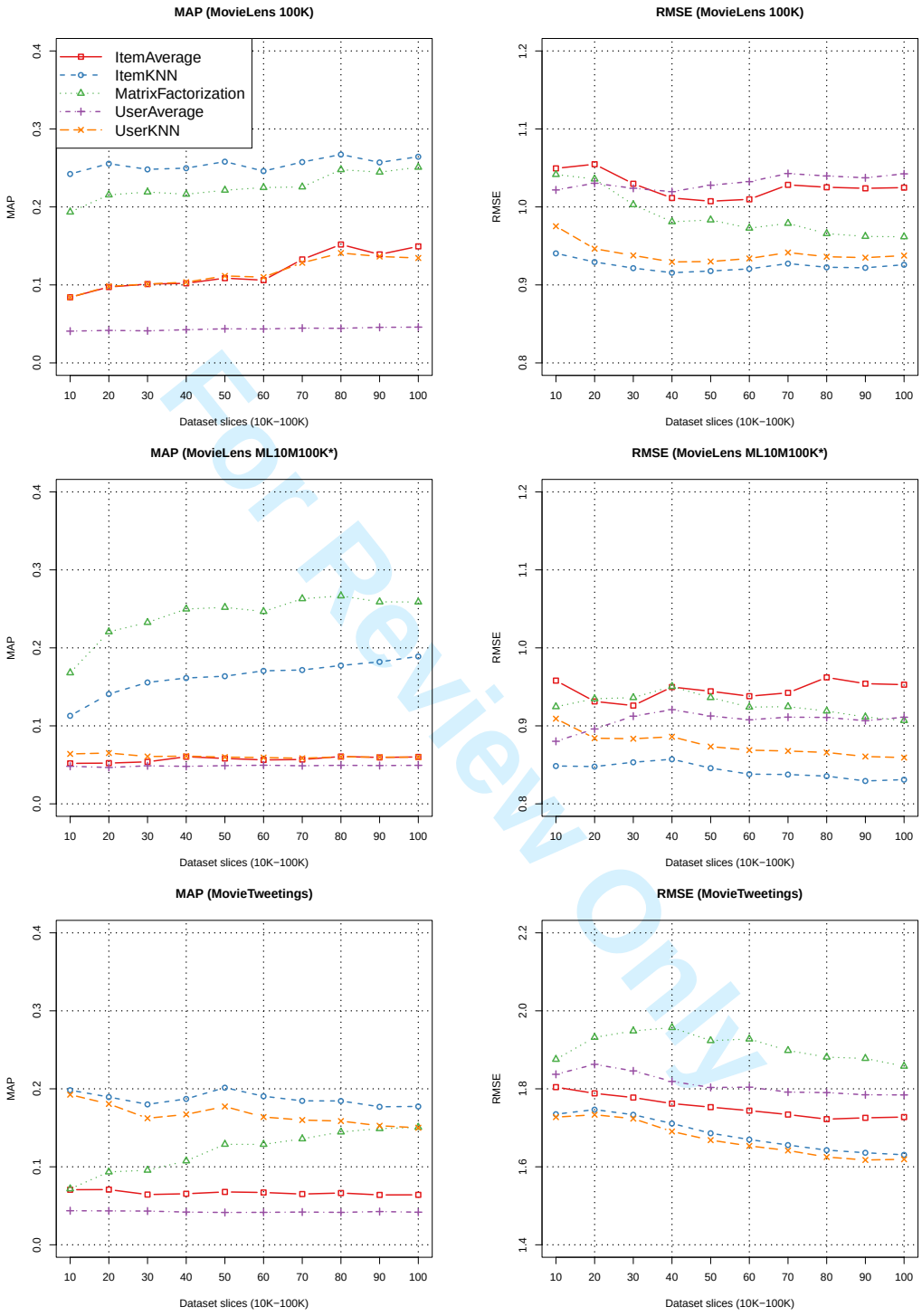

Fig. 11. Mean Average Precision (MAP, the higher the better) and Root Mean Squared Error (RMSE, the lower the better) metrics computed using a cross-validation 5-fold splitting strategy, using a varying number of ratings for each split (from 10K to 100K).

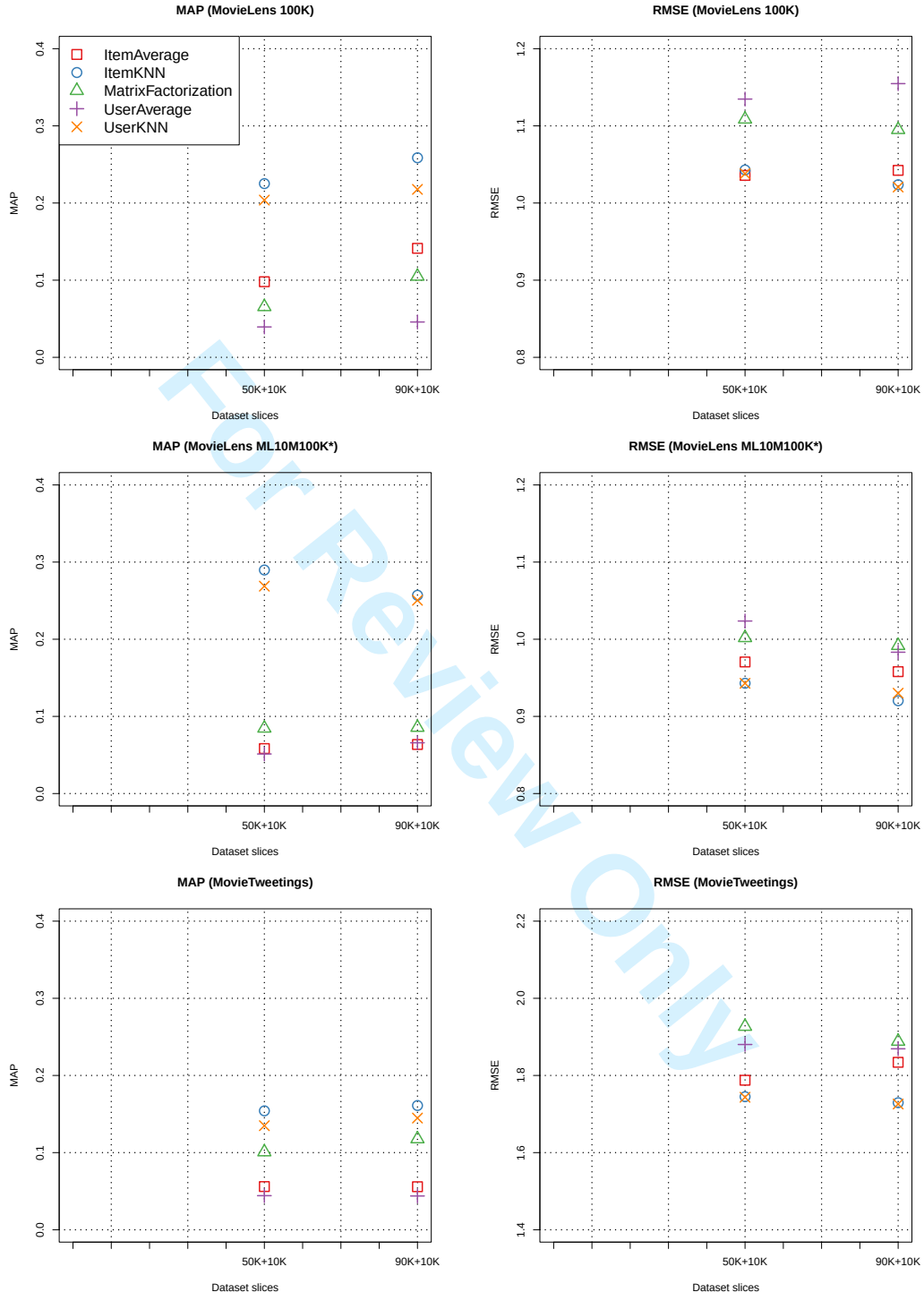

Fig. 12. MAP and RMSE metrics computed using a temporal splitting strategy.

Table IV. Description of the recommendation algorithms evaluated and the values of the parameters used

Name	Description	Parameters
ItemAverage	Target item's average rating	–
UserAverage	Target user's average rating	–
ItemKNN	Item-based nearest neighbor	k=80, correlation=BinaryCosine, reg_u=15, reg_i=10, num_iter=10
UserKNN	User-based nearest neighbor	k=80, correlation=BinaryCosine, reg_u=15, reg_i=10, num_iter=10
MatrixFactorization	Factorization of rating matrix using stochastic gradient descent	num_factors=10, regularization=0.015, learn_rate=0.01, learn_rate_decay=1, num_iter=30

4.2. Benchmarking Results

We now present the results obtained when comparing the algorithms listed in Table IV for the three datasets introduced before (whose statistics are summarized in Table I). We analyze these results (presented in Fig. 11 and Fig. 12) according to three dimensions: evaluation metrics, data splitting, and recommendation performance. Besides, we also experiment with different *data slices* or subsets of the original dataset that contain the first D_R ratings (where D_R varies from 10K to 100K, in 10K increments). For the evaluation metrics, we focus on Mean Average Precision (MAP) [Baeza-Yates and Ribeiro-Neto 2011] and Root Mean Squared Error (RMSE) [Shani and Gunawardana 2011], although other ranking-based metrics like nDCG, precision, and recall produced similar results as those obtained for MAP, likewise for MAE with RMSE. For data splitting we experiment first with a standard way for randomly generating training and testing splits: a cross-validation splitting strategy that generates non-overlapping subsets (to be used as training and testing splits) where every (user, item, rating) tuple is evaluated once – i.e., it only appears in one testing split, and it is guaranteed that there is one split containing such tuple. The results we report here have been averaged over 5 folds, although similar results were found with 10 folds. The second splitting strategy evaluated is based on a temporal splitting, where the testing split occurs after the training split. We use ten thousand ratings for testing, and experiment with a window of the previous $50,000$ or $90,000$ ratings as training split. These settings correspond with the following evaluation conditions, as presented in [Campos et al. 2014]: community-centered base set, time-dependent rating order, and fixed size (10K ratings). We apply these conditions to the whole dataset and to the subset of the first 60K ratings.

From Fig. 11 and Fig. 12 we observe that the performance of the recommender algorithms is heavily influenced by the **evaluation metric** (RMSE[16] or MAP) used to decide which recommenders perform better. Specifically, whereas the ranking-based metric (MAP) is very stable – in terms of dataset snapshots and splitting strategies –, preserving the trend in best/worst recommenders, the error-based metric (RMSE) has more fluctuations. Furthermore, the RMSE metric is not useful to discriminate which recommender is performing best because its values are very close to each other, and, in general, it is not consistent that the best method with RMSE (lowest value) achieves the best value with MAP (highest value) or viceversa[17]; in particular, the worst method according to RMSE in MovieTweetings is the matrix factorization algorithm, which has a medium-to-high performance in terms of MAP. We argue some of these differences between MAP and RMSE may be due to the different levels of

[16]Note that for RMSE, the range of ratings in MT is different (from 1 to 10) than the one from ML (from 1 to 5); thus, the range of possible error is higher in the former.

[17]This conclusion confirms works such as [McNee et al. 2006; Cremonesi et al. 2010; Bellogín et al. 2011] where error-based metrics show different behaviors, usually not linked with the final experience of the user.

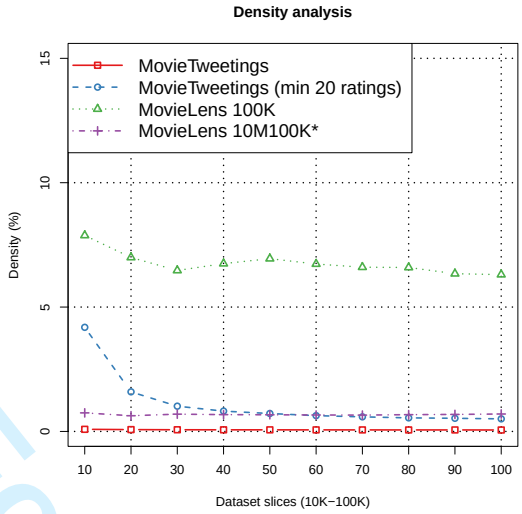

Fig. 13. Comparison of density values for each snapshot of the datasets analyzed in the paper.

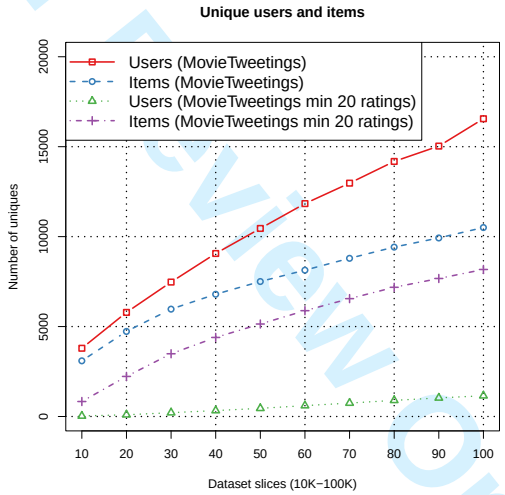

Fig. 14. Unique users and items at each MovieTweetings snapshot.

density (as defined in Equation 1) presented in each of these datasets, since in such scenario it is more likely that more users or items may have no training information (or very little) after splitting the dataset, which seems to have a stronger effect on error-based metrics and leads to very similar performance values for very different recommendation methods. Fig. 13 compares the density in these datasets every ten thousand ratings; in this context it is clear that the sparsity in the original MovieTweetings dataset is higher (i.e., density is lower), mainly because it contains a much larger number of users and items than the other datasets, but keeps the same number of ratings. This aspect of the dataset may also affect the fact that MAP is lower in MT than in ML datasets, because it is a more difficult (less dense) dataset.

To further analyze the differences in behavior when other assumptions in the dataset are considered, we generated a subset of MovieTweetings where only users with at least 20 ratings are kept, as the ML 100K dataset was originally released. As

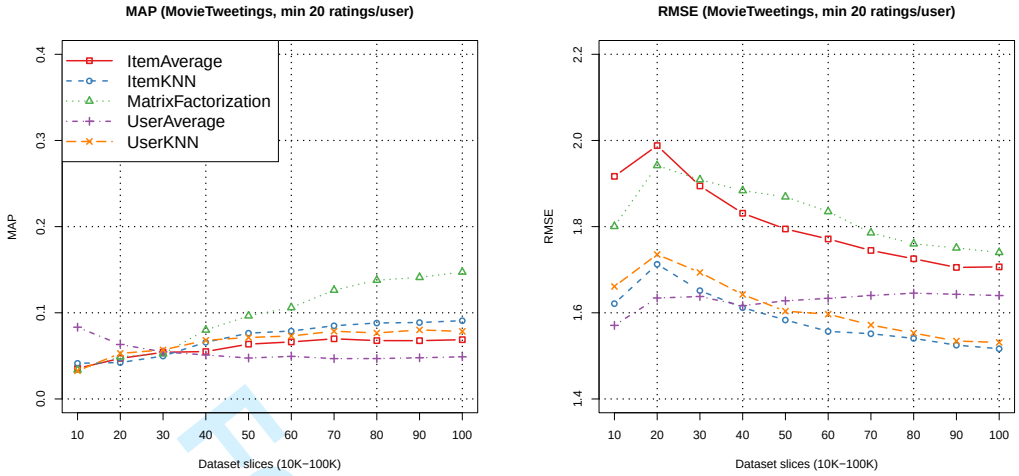

Fig. 15. MAP and RMSE metrics computed using a cross-validation 5-fold splitting strategy for the users in the MovieTweetings dataset with at least 20 ratings.

we see in Fig. 13, this artificially generated dataset is less sparse than the original MT dataset, although it gets closer to the original dataset when more ratings (larger dataset slices) are considered; this can be explained by looking at the dynamics of the number of users and items available at each slice point (Fig. 14). Furthermore, the recommendation performance (illustrated in Fig. 15) has now changed and the relative performance between the recommendation methods changes more often than in the previous case; also the range between RMSE values is larger for some of these algorithms and, especially for MAP, there are different best methods at each point, like the matrix factorization method, that outperforms the other algorithms after the 40K slice.

Regarding the **performance of recommendation algorithms**, each dataset has a different optimal method, but in general the matrix factorization recommender is among the top performing recommenders, in agreement with previous research in rating-based recommendation [Koren et al. 2009; Cremonesi et al. 2011], and at the same time, the non-personalized recommenders (user and item average) have a very low performance. It should be noted that the user-based nearest neighbor and the item average recommenders are equivalent in the MovieLens datasets in terms of MAP (meaning that their rankings are effectively the same), whereas in MovieTweetings they perform differently. Besides, the two neighbor-based recommenders (UserKNN and ItemKNN) outperform the rest of the algorithms in MovieTweetings, both in terms of MAP and RMSE.

The relative ranking-based performance, in most of the cases, does not change too much from the first slice (10K ratings) to the last one (100K ratings). One exception to this is the matrix factorization algorithm in the MT dataset. As already observed before in the literature [De Pessemier et al. 2010; Campos et al. 2011], recommendation performance increases in the MovieLens datasets when more ratings are available, but this is not the case with the MovieTweetings dataset, where all the recommenders, except the matrix factorization method, decrease or maintain their performance. A similar result was observed in [Said et al. 2009], where a dataset with several new items was used, which lowered the recommendation precision. These results are completely reversed when the subset of users with more than 20 ratings (Fig. 15) is analyzed, since here the performance increases with more data (except for the UserAverage method).

Note also that the best recommender in terms of MAP (the matrix factorization method with a value of 0.15 using all ratings) is the worst according to RMSE (value of 1.73, also using the whole dataset). A possible explanation to these effects is the amount of users and items presented in this constrained dataset (Fig. 14), where a smaller number of items and, especially, of users is available in the dataset, which makes it easier for the recommenders to obtain a higher performance.

Error-based metrics, as discussed before, are not very useful to decide which is the best recommender. In the same way, it is very difficult to decide in which of the slices the algorithms perform better. These facts make the error-based metrics not so interesting to predict which recommenders will perform better in the future, given a particular dataset slice. These predictions, however, could be easily drawn from the results with the ranking-based metrics.

Finally, it is interesting to note that the results obtained in each of the **data splitting** techniques evaluated are very consistent for the MT dataset, whereas this is not the case for the ML datasets. Specifically, the best and worst recommenders remain the same for ML 100K, but the ones in the middle vary drastically their relative performance; even worse, for ML 10M100K* the best recommender using cross-validation (matrix factorization) is one of the worst using a temporal split (the best here is ItemKNN). This consistency in the evaluation for the MovieTweetings dataset is a positive characteristic, since it shows a direct correspondence between the standard offline evaluation using repeatable tests (cross-validation) and the more realistic evaluation scenarios (temporal split).

In summary, we have evaluated different aspects of the MovieTweetings dataset from a practical perspective. We have found that it is not very different to the two versions of the MovieLens dataset we have experimented with when using a specific splitting strategy, but it is more consistent across data splitting strategies. Nonetheless, we have observed that its very high sparsity may produce lower performance scores in general, and that the RMSE scores are not very useful to discriminate the recommendation methods, although this was also true – to a lower extent – for the other datasets.

5. DISCUSSION: APPROPRIATE USE CASES AND ADVANTAGES

Finally, our benchmarking framework proposes a discussion of appropriate use cases for the application of the dataset and listing its advantages towards other currently available (similar) datasets. We start with the latter, i.e., a listing of the advantages of the MovieTweetings dataset.

— **Realistic user behavior**. The MovieTweetings dataset is unfiltered and therefore a natural dataset. No users or items are excluded from the dataset for not having a sufficient amount of ratings. This has important consequences for running experiments but also has the advantage of allowing to simulate realistic user behavior. A real-life recommender system will have to deal with non-active users and poorly rated items, and so the MovieTweetings dataset can offer a means to experiment with and simulate these scenarios.
— **Meta-data easily expandable**. The meta-data contained in the dataset consists of movie genres (as available in the MovieLens dataset), but can be very easily supplemented with other movie information. Because the dataset links every movie to the unique IMDb identifier, additional meta-data can be collected by using tools as the OMDb API or by scraping the original IMDb website itself.
— **Non-anonymized users**. Because rating data is collected from publicly available information, the users contained in the MovieTweetings dataset did not need to be anonymized. So the user identifiers used in the dataset can be linked to the

real Twitter users. Additional user data can easily be collected using the Twitter API and by analyzing online user behavior. Furthermore, the integration of real approachable users in the dataset offers an interesting user-base for researchers that may spark a new generation of low-effort online/live user-centric evaluation experiments.

— **Frequently updated**. The most important advantage and difference towards other datasets is the fact that the dataset is updated regularly. Instead of offering data from a fixed period in time, new ratings extracted from Twitter are added frequently. As long as the updating of the MovieTweetings dataset is maintained by its administrators, it will therefore continue to contain the most recent and popular movies, which makes it an interesting seed dataset for user-centric experiments.

Different datasets have different properties and their suitability therefore largely depends on the scenario they are deployed in. We now list some of the typical scenarios in which public rating datasets are deployed and compare the applicability of MovieTweetings against other public datasets such as MovieLens.

— **Use MovieTweetings for user-centric experiments.**
User-centered experiments are one of the scenarios in which very often public rating datasets are imported into recommender systems. In such experiments the goal is to have real users interact with a recommender system to learn about their satisfaction and behavioral patterns. Evaluation may be focused on various aspects of e.g. the recommendation algorithm or the user interface. To be able to generalize results from user-centric experiments, the experiments must be performed on a realistic deployment of the recommender system. Realistic deployment means that the system properties should be as close as possible to those of the final system (if it would be deployed for real) including available content items and a sufficient amount of users. Many examples can be found in the literature where the MovieLens dataset is used to drive such user-centric experiments (e.g., [Hurrell and Smeaton 2013; Said et al. 2013]). While in the past MovieLens may have been an appropriate sample of realistic rating data, nowadays the dataset is outdated and may even negatively influence the user experience. To illustrate this, we run a small experiment where we visually inspect (as real users would do) the recommendations generated for a random user, using either the MovieLens (100K) or MovieTweetings (100K) dataset as input data. The MyMediaLite MatrixFactorization implementation was used to generate the recommendations. Table V compares the top 10 results for a random user of both datasets.

Table V. Comparing MatrixFactorization recommendation results for the MovieLens and MovieTweetings datasets

MovieTweetings	MovieLens
Wrong Turn 5: Bloodlines (2012)	Pulp Fiction (1994)
Sound City (2013)	The Godfather (1972)
The Cloth (2012)	GoodFellas (1990)
Mud (2012)	Paths of Glory (1957)
Now You See Me (2013)	Apocalypse Now (1979)
Ted (2012)	Secrets & Lies (1996)
The Host (2013)	Citizen Kane (1941)
Mr. Popper's Penguins (2011)	A Space Odyssey (1968)
The Hunger Games (2012)	The Third Man (1949)
Warrior (2011)	Chinatown (1974)

The results illustrate how the recentness of the recommended movies based on MovieLens is limited, while MovieTweetings has much more recent data and thus is able to recommend more recent movies. Even the best recommendation algorithm may be evaluated negatively if it can only recommend from an outdated item catalog. **For all experimentation that involves actual users, we therefore recommend to use the MovieTweetings dataset instead of the more commonly used MovieLens dataset**. We have shown that the MovieTweetings dataset can be regarded as a recent update of the MovieLens dataset showing much of the same properties albeit with higher amounts of items and users. More items may actually be an additional advantage for user-centric experiments since the item catalog may be larger, more diverse and therefore has a higher chance of containing interesting items.

In particular for user-centric experiments MovieTweetings has a unique feature that no other public rating dataset offers: non-anonymized users. Users in the dataset are actual Twitter users and can publicly be interacted with. This opens up the Twitter platform for all sorts of novel types of evaluation experiments like e.g. deploying a recommender system as an automated Twitter account.

— **Use MovieTweetings for realistic offline simulation.**

Simulation is another type of scenario in which public rating datasets are often imported into recommender systems. When a recommendation algorithm has been implemented, its designer may be curious about the performance in terms of some offline calculable metrics such as *RMSE*. Again literature often employs the MovieLens dataset to do this, but as noted by [Gunawardana and Shani 2009], those results are biased towards users with a large number of ratings. Since MovieLens is a filtered dataset containing only users with a minimum of 20 ratings, obtained results may not be generalizable to users with 19, 18, or even less, ratings. The MovieTweetings dataset allows us to verify this claim.

For the 5 algorithms introduced in Section 4.1, we calculate the average RMSE value for different configurations of the dataset, each time containing only users with minimum x ratings, with x ranging from 1 to 20. Fig. 16, shows the results of the experiment. As expected, a significant variance can be detected between the different configurations. The results clearly show how a metric such as RMSE can fluctuate when changing sets of users are taken into account and, therefore, no overall RMSE truly represents all users in a system.

Since MovieTweetings is an unfiltered dataset, RMSE values can be calculated for all sorts of users, which allows for a more realistic simulation of offline recommendation quality. An example of a situation where such an analysis is useful, is the decision of the minimal required ratings for users in a system [Kluver and Konstan 2014]. Recommender systems usually do not recommend items to new users but instead require them to rate a sufficient number of items first. This number is often chosen arbitrarily, but the MovieTweetings dataset may support such a decision by making it possible to simulate recommendation quality metrics for different groups of users. For instance, based on the results in Fig. 16 we could decide that, for our recommender, a minimum of 8 ratings should be required for new users, since having more than 8 ratings seems to no longer lead to substantially better RMSE values.

Another example of offline simulation made possible by the MovieTweetings dataset is measuring which algorithms work best on different kinds of users. In Fig. 17, we show the RMSE results for again the same 5 recommendation algorithms but this time restricting the dataset in the different configurations (on the X-axis) to contain only users with an exact (instead of minimum) number of ratings x, with x ranging from 1 to 20. The results show how the *ItemAverage, UserKNN,* and *ItemKNN* seem

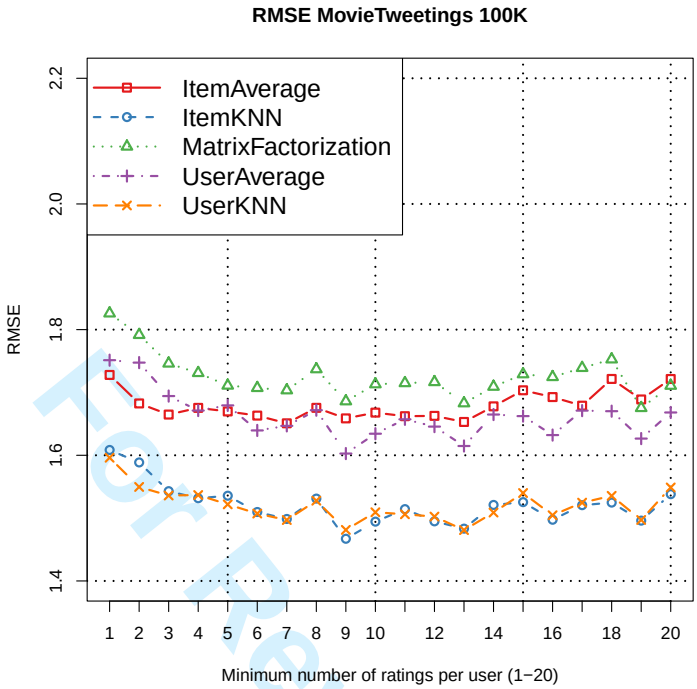

Fig. 16. The RMSE values averaged over all users for changing configurations including only users with a minimum of 1 rating, 2 ratings, 3 ratings, and so on.

good (i.e., RMSE is lower) candidate recommendation algorithms for users with a low number of ratings, while *UserAverage, UserKNN,* and *ItemKNN* are good approaches for users with a number of ratings close to 20. Interestingly, the simple *ItemAverage* and *UserAverage* algorithms are showing almost identical recommendation quality as the more advanced (but computationally heavier) *ItemKNN* and *UserKNN* methods.

The MovieLens dataset does not support analyzing patterns as those presented in the previous examples. Researchers may attempt to simulate behavioral patterns of users with less than 20 ratings by creating artificial MovieLens users (with e.g., randomly removed ratings), but the generalization power and correctness of such results would be equally artificial.

— **Use MovieLens (and MovieTweetings) for comparative offline experiments.** Aside from serving as bootstrap data for either user-centric or simulation experiments, public rating datasets are also often used in comparative offline experiments (e.g. [de Castro et al. 2007]). Academic researchers having implemented a new recommendation method, often publish the details of their algorithm with some accompanying results of how the algorithm compares to other state of the art (or well-known) recommendation algorithms. Since the goal here is to benchmark two (or more) algorithms relative to each other, the dataset in itself is not that important. What is important though is that the same dataset and other evaluation dimensions (metric, splitting strategy, etc.) are used equivalently for each benchmark situation [Said and Bellogín 2014]. The MovieLens 100K has served as a well-accepted benchmarking dataset in the recommender systems domain for many years, and

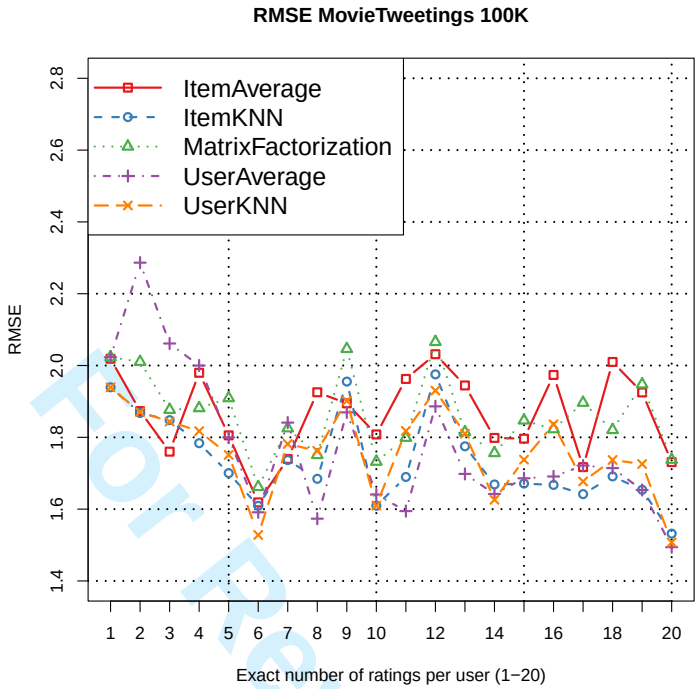

Fig. 17. The RMSE values averaged over all users for changing configurations including only users with exactly 1 rating, 2 ratings, 3 ratings, and so on.

so there is no reason to change that. Researchers are familiar with the dataset structure, its properties, and typical values for the popular evaluation metrics on default recommendation algorithms. **We thus recommend the continued use of the MovieLens dataset for comparative offline experiments.**

Nonetheless, we note that as time progresses, the MovieLens dataset will become less relevant and eventually researchers will have to switch to more recent datasets even for offline comparative scenarios. We therefore recommend researchers to future-proof their work by reporting algorithmic results measured on both the MovieLens as well as the MovieTweetings dataset. Since the data formats of both datasets are identical, in most cases the implementation effort to do this will be very low.

6. CONCLUSIONS

In this work, we proposed a framework for introducing and benchmarking new datasets in the research domain. The five-step framework was illustrated on a movie ratings dataset called MovieTweetings. In step 1, the origin of the dataset was detailed. The MovieTweetings dataset is collected dynamically from Twitter and originates from structured tweets posted through the IMDb platform. Step 2 presented basic descriptive statistics about the dataset while comparing them with other known (and similar) datasets in the domain. We found that even though equal dataset sizes were taken into account, the dataset showed to have some significantly different properties in comparison with the MovieLens dataset. The extreme low density (or high sparsity) resulting from a much higher number of items and users was one of these observations. In step 3,

we focused on external validity. Some hypothesized data biases were investigated (e.g., how similar are the MovieTweetings ratings to the IMDb ratings) through a number of correlation analyses and the dataset was compared with other similar datasets in the movie domain. We showed that with a sufficient minimum number of ratings per movie, the dataset correlated strongly with the rating patterns as found on the IMDb website. Although the MovieTweetings items are more biased towards more recent and popular movies, similar rating patterns to those of the MovieLens dataset could be observed. In step 4, we reported some results on a number of reproducible benchmarks exploring multiple dimensions as recommendation algorithms, evaluation metrics, and data splitting strategies. Lastly in step 5, the advantages of the dataset (against other datasets) were listed and scenarios in which the dataset could be successfully deployed were discussed. We showed how MovieTweetings was especially interesting for user-centric and simulation focused experiments and could be used complementary to the MovieLens dataset for offline comparative experiments.

By illustrating our analysis of the MovieTweetings dataset and its properties, we hope to inspire future researchers to publish and benchmark datasets in a similarly structured fashion such that the added value, properties, and optimal use cases of future datasets can easily be compared and interpreted.

Several open problems with benchmarking and comparing datasets still remain. Probably the most important one is how to benchmark a dataset with unique properties or even from a new domain ('are the results obtained using MovieTweetings transferable to a domain like Facebook?'), since we base most of our analysis on comparisons against other datasets from similar domains. Another related problem would be how to deal with datasets that combine more than one domain, so that cross-domain recommendations can be tested. Because of the limited dimensions of the MovieTweetings dataset (compared to other available datasets) we were unable to analyze the effect of sampling size beyond 100K ratings. Since the dataset is still growing on a daily basis, we look to future work for reporting such additional analyses.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful and constructive comments.

REFERENCES

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. 2011. *Modern Information Retrieval - the concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England.

Alejandro Bellogín. 2012. *Performance prediction and evaluation in Recommender Systems: an Information Retrieval perspective*. Ph.D. Dissertation. Universidad Autónoma de Madrid.

Alejandro Bellogín, Pablo Castells, and Iván Cantador. 2011. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *RecSys*, Bamshad Mobasher, Robin D. Burke, Dietmar Jannach, and Gediminas Adomavicius (Eds.). ACM, 333–336.

Alejandro Bellogín, Pablo Castells, Alan Said, and Domonkos Tikk. 2013. Workshop on reproducibility and replication in recommender systems evaluation: RepSys. In *RecSys*, Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis (Eds.). ACM, 485–486.

Alejandro Bellogín, Arjen de Vries, and Jiyin He. 2013. Artist popularity: do web and social music services agree. In *Int. Conf. on Weblogs and Social Media (ICWSM), Boston*.

James Bennett and Stan Lanning. 2007. The Netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. 35.

Jesús Bobadilla, Francisco Serradilla, and Jesus Bernal. 2010. A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems* 23, 6 (2010), 520–528.

Pedro G. Campos, Fernando Díez, and Iván Cantador. 2014. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-adapted Interaction* 24, 1-2 (2014), 67–119.

Pedro G. Campos, Fernando Díez, and Manuel Sánchez-Montañés. 2011. Towards a more realistic evaluation: testing the ability to predict future tastes of matrix factorization-based recommenders. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 309–312. DOI:http://dx.doi.org/10.1145/2043932.2043990

Paolo Cremonesi, Franca Garzotto, Sara Negro, Alessandro Papadopoulos, and Roberto Turrin. 2011. Comparative evaluation of recommender system quality. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11)*. ACM, New York, NY, USA, 1927–1932. DOI:http://dx.doi.org/10.1145/1979742.1979896

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 39–46. DOI:http://dx.doi.org/10.1145/1864708.1864721

Pablo AD de Castro, Fabrício Olivetti de França, Hamilton M Ferreira, and Fernando J Von Zuben. 2007. Applying biclustering to perform collaborative filtering. In *Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on*. IEEE, 421–426.

Toon De Pessemier, Simon Dooms, Tom Deryckere, and Luc Martens. 2010. Time dependency of data quality for collaborative filtering algorithms. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 281–284. DOI:http://dx.doi.org/10.1145/1864708.1864767

Simon Dooms, Toon De Pessemier, and Luc Martens. 2014. Cross-domain rating datasets from structured data on Twitter. In *Workshop on Modeling Social Media: Mining Big Data in Social Media and the Web (MSM), at WWW 2014*.

Cristina Gena, Roberto Brogi, Federica Cena, and Fabiana Vernero. 2011. The impact of rating scales on users rating behavior. In *User Modeling, Adaption and Personalization*. Springer, 123–134.

Asela Gunawardana and Guy Shani. 2009. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research* 10 (2009), 2935–2962.

Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 230–237.

Eoin Hurrell and Alan F Smeaton. 2013. A conversational collaborative filtering approach to recommendation. In *Advances in Visual Informatics*. Springer, 13–24.

Daniel Kluver and Joseph A. Konstan. 2014. Evaluating recommender behavior for new users. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*. 121–128. DOI:http://dx.doi.org/10.1145/2645710.2645742

Bart P. Knijnenburg, Martijn C. Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the user experience of recommender systems. *User Model. User-Adapt. Interact.* 22, 4-5 (2012), 441–504.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08)*. ACM, New York, NY, USA, 426–434. DOI:http://dx.doi.org/10.1145/1401890.1401944

Yehuda Koren. 2009. The Bellkor solution to the Netflix grand prize. *Netflix prize documentation* (2009).

Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* 42, 8 (2009), 30–37.

Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: state of the art and trends. In *Recommender Systems Handbook*. 73–105.

Benjamin M Marlin and Richard S Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*. ACM, 5–12.

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. DOI:http://dx.doi.org/10.1145/1125451.1125659

Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen M. Carley. 2013. Is the sample good enough? Comparing data from Twitter's streaming API with Twitter's Firehose. In *ICWSM*, Emre Kiciman, Nicole B. Ellison, Bernie Hogan, Paul Resnick, and Ian Soboroff (Eds.). The AAAI Press.

Verónika Peralta. 2007. *Extraction and Integration of MovieLens and IMDb Data*. Technical Report. Technical Report, Laboratoire PRiSM, Université de Versailles, France.

Martin Piotte and Martin Chabbert. 2009. The pragmatic theory solution to the Netflix grand prize. *Netflix prize documentation* (2009).

Alan Said and Alejandro Bellogín. 2014. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*. 129–136. DOI:http://dx.doi.org/10.1145/2645710.2645746

Alan Said, Ben Fields, Brijnesh J Jain, and Sahin Albayrak. 2013. User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 1399–1408.

Alan Said, Robert Wetzker, Winfried Umbrath, and Leonhard Hennig. 2009. A hybrid PLSA approach for warmer cold start in folksonomy recommendation. In *Proceedings of the RecSys'09 Workshop on Recommender Systems & The Social Web*. CEUR-WS Vol. 532, 87–90.

Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender Systems Handbook*. 257–297.

Andreas Töscher, Michael Jahrer, and Robert M Bell. 2009. The BigChaos solution to the Netflix grand prize. *Netflix prize documentation* (2009).

Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 427–434.