Investigating the Impact of Recommender Systems on User-based and Item-based Popularity Bias

Mehdi Elahi[a], Danial Khosh Kholgh, Mohammad Sina Kiarostami[b], Sorush Saghari, Shiva Parsa Rad, Marko Tkalčič[c]

[a] *University of Bergen, Bergen, Norway*
[b] *University of Oulu, Oulu, Finland*
[c] *University of Primorska, Koper, Slovenia*

Abstract

Recommender Systems are decision support tools that adopt advanced algorithms in order to help users to find less-explored items that can be interesting for them. While recommender systems may offer a range of attractive benefits, they may also intensify undesired effects, such as the *Popularity Bias*, where a few popular users/items get more popular and many unpopular users/items get more unpopular.

In this paper, we study the impact of different recommender algorithms on the popularity bias in different application domains and recommendation scenarios. We have designed a comprehensive evaluation methodology by considering two different recommendation scenarios, i.e., the user-based scenario (e.g., recommending users to users to follow), and the item-based scenario (e.g., recommending items to users to consume). We have used two large datasets, Twitter and Movielens, and compared a wide range of classical and modern recommender algorithms by considering a diverse range of metrics, such as PR-AUC, RCE, Gini index, and Entropy Score.

The results have shown a substantial difference between different scenarios

Email addresses: Mehdi.elahi@uib.no (Mehdi Elahi), dkhoshkholgh@acm.org (Danial Khosh Kholgh), Mohammad.Kiarostami@student.oulu.fi (Mohammad Sina Kiarostami), sorush9saghari@gmail.com (Sorush Saghari), sh.parsarad@gmail.com (Shiva Parsa Rad), marko.tkalcic@gmail.com (Marko Tkalčič)

URL: https://www.uib.no (Mehdi Elahi), https://www.oulu.fi (Mohammad Sina Kiarostami), https://www.upr.si (Marko Tkalčič)

and different recommendation domains. According to our observations, while the recommendation of users to users may increase the popularity bias in the system, the recommendation of items to users may indeed decrease it. Moreover, while we have measured a different level of popularity bias in different languages (i.e., English, Spanish, Portuguese, and Japaneses), the above-noted phenomena has been consistently observed in all of these languages.

Keywords: Recommender Systems, Popularity Bias, personalization, Twitter, social media

---

1. Introduction

The spread of recommender systems in the daily life of users is causing that more and more decisions are affected by recommendations [1, 2]. Recommender systems adopt a wide range of algorithms [3, 4] in order to learn from the user preferences, elicited in various forms [5], and to generate a small set of *recommended items*, which have high utility for a user, from a larger set of items (the *input dataset*) [6, 7]. The recommended set of items might be biased towards a group of items (e.g. popular items constitute the majority of recommendations), which, in turn, may have a substantial impact on users' decisions and consumption behaviour, for examples on purchases or information consumption.

This algorithmic confounding is a more general problem in artificial intelligence and it can be found in many forms [8]. For example, it has been found that the number of friends on Facebook is not a reflection of a user's popularity or extroversion, as one would expect, but of the bias of the recommendation algorithm in Facebook [9]. While this observation does not seem to have strong implication, other cases of algorithmic confounding show a different picture. In a study carried out by Chaney et al. [10], the authors found that recommendation algorithms increase homogeneity of user behaviour and decrease the satisfaction of users. According to Baeza-Yates [11], having a recommender system biased towards popular items might undermine the consumption or sales of items that are not popular, hence preventing them to become popular. If

the majority of a company's revenue is generated by a few popular items this is not a problem from the company's point of view. However, if a company's business is generated by a majority of less-popular items then having such a recommender system undercuts the company's revenues.

Algorithmic confounding is not the only source of bias in recommender systems [8]. Due to the selection bias, i.e. data collection that is not properly randomized, datasets come with inherent biases even before algorithms are being applied. In [12], Baeza-Yates claims that the access and usage of the WWW is correlated with educational, economic, and technological user characteristic. For example, it is estimated that the majority, around 50% of web pages are in English while the active English-speaking population is only 5% [12]. It is easy to imagine a recommender system that would recommend disproportionately more English web pages than non-English ones.

It has been argued recently that, if the input dataset is already biased, the recommendation algorithm may intensify that existing bias [13]. Some researchers, such as Adomavicious et al. [14], assumed that the recommendation algorithm strengthens the bias and proposed an approach to de-bias the recommended set. More recent research investigated the extent to which the recommended algorithms intensify the bias. For example, Mansoury et al. [13] showed that, on the Movielens dataset, the recommendation algorithm actually does intensify the pre-existing bias from the input dataset.

However, the assumption that a recommender algorithm does intensify the bias is arguable. It is often true that the bias in the recommended set is different than the bias in the initial dataset. However, we argue that there are three possible scenarios:

(a) The bias in the recommended set is higher than the bias in the initial dataset. For example, on a user level, in the output set less users/authors generate a bigger portion of all items. Another example, on an item level, is that the majority of engagements with items is done on a smaller set of items.

(b) The bias in the recommended set is roughly the same as the bias in the initial dataset.

(c) The bias in the recommended set is lower than the bias in the initial dataset. For example, in the output set more users/authors generate a bigger portion of all items or the majority of engagements with items is done on a larger set of items.

In this paper we investigate whether recommendation algorithms increase the bias (option a), keep the bias unchanged (option b), or reduce the bias (option c). We explore different domains (movies and micro-blogging), different languages (in the micro-blogging domain: English, Japanese, Spanish, and Portuguese), and we analyze the results from two different perspectives (from the user- and item-perspective). In particular, we address the following research questions:

- **RQ1**: Does the intensification/reduction of bias in the recommended set occur both on a user- and item-basis?
- **RQ2**: Does the intensification/reduction of bias in the recommended set occur in datasets from different domains, in particular movies (Movielens) and micro-blogging (Twitter)?
- **RQ3**: Are there any differences in the intensification/reduction of bias between different languages in the Twitter domain?

In summary, the present paper makes the following contributions

- We compare the bias reinforcement on two separate datasets (MovieLens and Twitter) and show that bias is reinforced differently on different datasets;
- We demonstrate that recommendation algorithms increase the bias at the user level (i.e. in the output set less users generate a bigger portion of all tweets) whereas there is a reduction of bias on the tweet level (i.e. the majority of the engagement is spread across more tweets than in the initial set of tweets);

- We show that there are differences in how the recommendation process affects the bias on the language basis;
- We evaluate the extent of reinforcement of recommendation-driven bias on several algorithms.

The remainder of the paper is organized as follows. In section 2, we survey the related work on biases in recommender systems. In section 3, we describe the details of our experimental methodology. Finally, in section 4, we report the results of the experiments and in section 5, we provide the conclusion and plans for the future work.

2. Related Work

Here we review the related work, going from more general to more specific. First we will survey the bias in information systems in general, then the works on bias in recommender systems and finally the work on popularity biases in recommender systems.

2.1. Bias in Information Systems

Bias is a concept that has its root in different disciplines and hence can be seen from a diverse perspectives. Mehrabi et al. [15] surveyed existing biases in the broader fields of artificial intelligence and information systems. They distinguish between data biases and algorithm-induced biases. One of the reported biases, related to our work is the activity bias, to which the bias is datasets can be partially attributed. The authors in [16], found substantial activity bias in user-generated content online. They have shown that only 2% of Twitter users in 2009 created nearly 50% of all tweets. Additionally, they found that 1.1% of all tweets have been written and posted by users with a very low number of followers.

A study by Wu et al. [17] explored how users followed other users on Twitter. They found that 0.05% of the most popular users have attracted almost 50% of all followers. In other words, nearly half of the Twitter users follow only a small

set of top celebrities. In a similar study, the authors in [12] explored Facebook data and found that 7% of all users produced 50% of all posted content.

2.2. Bias in Recommender Systems

A number of prior works have studies the potential bias in recommender systems, i.e. the bias in the user interaction or the bias based on the self-explored items. This may depend very much on the core algorithm employed by a recommender system to generate recommendations and how the collected history of user interaction is used by the algorithm [18, 19]. If a recommender system algorithm focuses on exploiting only interaction data, the users may only see what she *want* to see (i.e., obvious recommendations). This may keep the users stay inside a closed world (known as the *filter bubble*), cut out of new (or diverse) items that might also have utility for the users [20].

In general, the bias in recommendation output can originate from the pre-existing bias in the input data. Such bias could be intensified by the algorithm by further propagating the existing bias from the input data. Such an effect could strengthened over time as the users interact more and more with the recommendations generated based on the data by the recommender algorithm from the previous steps.

This is why several studies have focused on this phenomena and showed that what happens in reality is more complicated than the above-described process. An example of such studies is the one carried out by Mansoury et al. [13], where the authors studied the effect of the feedback loop on the bias amplification in recommender systems through an offline simulation. The paper formally and empirically showed that different recommendation algorithms amplify the existing bias through various iterations of user interaction. Additionally, for two user groups of males and females, they observed that the bias amplification for the females, which happens to be in minority group based on their population and their number of ratings, was higher than males. This means that the impact of the feedback loop is generally stronger for users who belong to a minority group.

Some studies have shown consistent and robust evidence that consumers' ratings are biased toward system-generated recommendations. In [21], the authors measured the bias induced by the display context as the mean difference between people's conclusions when a high and low value was presented. In their study, different display designs, adopted by recommender systems for communicating personalized recommendations to the user, were assessed. The authors found substantial bias level and reported that none of the rating display options completely removed the biases generated by the customized recommendations. They reported that some interface displays were more advantageous than others for reducing biases. In particular, graphical recommendation formats led to significantly lower biases in users' post-consumption preference ratings than their equivalent numerical forms (either as precise numbers or numeric ranges). However, providing recommendations as ranges of predicted rating values rather than as accurate values (either under graphic or numeric conditions) did not impact the appearance of biases.

2.3. Popularity Bias in Recommender Systems

A well-known type of bias in recommender systems is the *popularity bias*. This bias is intrinsically inherited by any dataset, which includes a small number of popular items (typically referred to as the *short head*) and a big number of unpopular items (typically referred to as the *long tail*) [22]. Since the short head items are already popular (e.g., highly-liked tweets or highly-watched movies), the recommender systems are typically presented as smart tools which can promote more the unpopular long tail items hence increasing their popularity. It is believed that this could provide various benefits to users, as they could find interesting less-explored items, as well as enhancing the profit of the platforms (e.g., Twitter and YouTube). The latter happens since the overall user engagement and content consumption increases.

An interesting observation, reported in some of the related works, is that even the user ratings (and hence overall average rating) of items is strongly influenced by the recommendations generated by the system. Although popular

items are generally *good* recommendations, they are also likely to be well-known. So delivering only popular items will not enhance new item discovery and will ignore users' interests with niche tastes. According to [23, 24], while ∼20% of items account for popular items, cumulatively, more than ∼80% of the user ratings are provided to them. The long tail receives so few ratings that meaningful cross-user comparison of their ratings becomes unreliable. Since short-head items are likely to be well-known to many users, the ability to recommend items outside of this band of popularity will determine if a recommender system can introduce new products and experiences.

Twitter is one of the most effective tools in propagating information in real time, and the propagation effectiveness of a tweet is related to the number of times the tweet has been retweeted. Different models have been proposed in the literature to understand the retweet proneness of a tweet (tendency or inclination of a tweet to be retweeted). For example, in [25], a model for obtaining the indications about the probable number of retweets a particular tweet may obtain from the social network was proposed.

In summary, almost all of the aforementioned related work demonstrated that recommender systems either introduce bias (to an unbiased initial dataset) or increases the bias in the initially biased dataset. However, our work complements this knowledge with several contributions, among others, by showing that in the micro-blogging domain (Twitter) a recommender system may actually generate both of these effects, depending on the recommendation and experimental scenario. In other words, the majority of the prior works have focused on a single scenario, either the item-based scenario or the user-based scenario. Our work is one of the very few works that covers both scenarios. Hence, we investigate the potential bias in the item recommendation scenario (e.g., recommending tweets or movies) as well as investigating the potential bias in the user recommendation scenario (e.g., recommending twitters or movie directors).

Our extensive experiments, considering item-based and user-based scenarios, have shown a substantial difference between bias in recommendation when

focusing on items compared to when focusing on users. We have shown that the bias may very much depend on the characteristics of the data as well as the specifications of the core recommender algorithm. We have adopted advanced recommender algorithms (including deep learning models) and compared them in terms of relevant metrics, such as the Gini index and the Entropy score.

Moreover, none of the previous works have investigated the potential impact of languages on bias in recommender systems. To the best of our knowledge, this is the first study that has investigated such an impact.

3. Methodology

In addressing the formulated research questions, we have designed and conducted a number of experiments. In the first set of experiments, we performed an exploratory analysis to better understand the data's characteristics. This has been followed by the performance evaluation of different recommender algorithms and the measurement of the quality of recommendation in terms of two metrics, adopted by Twitter in RecSys Challenge 2020 [26]: the Area Under the Precision-Recall Curve (PRAUC) and the Relative Cross-Entropy (RCE). In the second set of experiments, we measured the impact of different algorithms on the popularity bias in terms of the Gini index and the Entropy score. Finally, we have investigated the potential impact of the language of Twitter users, when measuring the popularity bias, generated by recomemnder algorithms.

3.1. Datasets

Two different datasets have been used for the experiments: the Twitter and the Movielens dataset. The Twitter dataset has been provided by Twitter for the RecSys Challenge 2020 [26]. The final version of the Twitter dataset includes 126 million user engagements (i.e., like, reply, retweet and retweet with comment) with 23.4 million users and 60.8 million tweets. The dataset has been collected over a period of two weeks and it contains public user interactions. Furthermore, we have used the Movielens 1M dataset, which contains one million anonymous ratings provided by 6040 users to 4000 movies.

3.2. Evaluation Methodology

We have split both of the datasets into the train set (85%) and test set (15%). For the Twitter data, the train set contained 107 million tweet-user pairs and the test set contained 19 million tweet-user pairs. We have used the train set to build different recommendation models and to predict the data within test set. This allowed us to assess the quality of the recommendation models predicting different types of the user engagements within the test set. For the Movielens dataset, the train set contained nearly 850000 ratings and the test set contained 150000 ratings. The train set has been used to train the models to predict the ratings in the test set. We have measured the quality of the predictions using the Relative Cross-Entropy (RCE) and Area Under the Precision-Recall Curve (PR-AUC) [26].

The RCE metric corresponds to the improvement of a prediction relative to the straw man, or the naive prediction, measured by cross-entropy (CE). The naive prediction corresponds to the case that does not consider the user and tweet features, i.e. it always predicts the average (observed) CTR of the train set. Suppose the average CE of the naive prediction is CE_{naive} and the average CE of the prediction to be evaluated is CE_{pred}, then RCE is defined as

$$RCE = 100\frac{(CE_{naive} - CE_{pred})}{CE_{naive}}$$

.

The benefit of using RCE is that we can obtain a confident estimate of whether the model is under- or over-performing the naive prediction [26]. The higher the RCE, the higher the prediction quality, and hence the better the performance of the recommender system.

The PR-AUC metric is equivalent to the true positive rate (or sensitivity) in a classification problem, while precision is the same as the positive predictive value. Reviewing both precision and recall is particularly useful when there is an imbalance in the observations between two classes. The PR-AUC is a commonly used evaluation metric and is more sensitive than AUC on skewed

data. The higher this metric the better the performance of the recommender system.

We measured the level of bias in the raw data and compared it with the output of the recommender algorithms to understand whether the bias has been intensified or not by these algorithms. For that we analyzed the Twitter dataset and sorted the tweets according to their popularity (from the most popular to the least popular). The popularity of a tweet has been measured in terms of the number of times the users have interacted with (e.g., liked, replied, retweeted, or retweeted with comment) that tweet. We called the rank of an individual tweet in the sorted list the tweet rank. Accordingly, we computed the cumulative frequency of the user engagement for different tweets at a given rank. Similarly, we computed the user rank representing the rank of a Twitter user when all users are sorted according to their popularity. The popularity of a user has been measured according to the number of times the tweets of that user have been interacted with the other users. This is followed by computing the cumulative frequency of the user engagement for different users at a given rank. Then we have compared the cumulative frequency of recommended items according to the number of times they appeared in the recommendation lists generated for different users. This allowed us to understand whether a recommender algorithm intensifies the level of the pre-existing popularity bias or not. This methodology has been repeated for the Movielens dataset.

In addition to that, we computed the Gini index and the Shannon Entropy in our evaluation. The Gini index is a common metric that represents the level of equality. The Gini index is originally based on the Lorenz curve and a lower Gini index indicates higher equality [27]. Let $L(u)$ be the Lorenz curve denoting the fraction of the sales generated by the lowest $100*u\%$ of the sold items u $\in$ [0,1], i.e., $L(u)$ for a data point $u \in [0,1]$ is the fraction of sales of products that are in 100*u% of the least sold items. A sample of a Lorenz curve is shown in Figure 1.

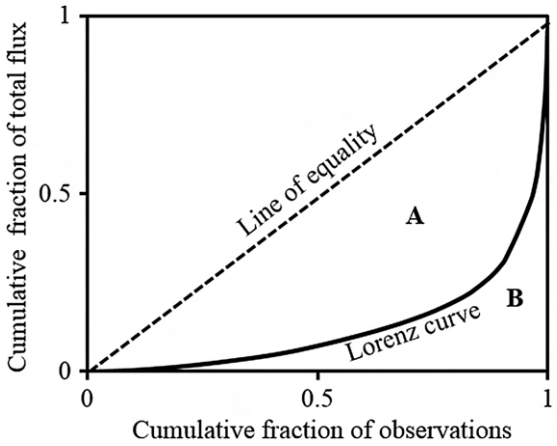

Figure 1: Lorenz curve (source: [28])

The Gini index is defined as:

$$G = \frac{A}{A+B}, \tag{1}$$

$$A = \int_0^1 (u - L(u))du, B = \frac{1}{2} - A$$

In Economics, a higher level of equality is defined by a situation where people's earnings are similar. This is illustrated by the Lorenz curve closer to the line of equality, i.e. the diagonal line in Figure 1 (Gini value of 0). The lower level of equality, on the other hand, is defined by the situation when people's earnings are very different. This is where the Lorenz curve gets closer to the horizontal axis (Gini value of 1) [1].

In addition to the Gini index, we computed the Shannon Entropy [29] according to the following formula:

$$H = -\sum_{i=1}^{n} p_i.log(p_i); \quad p_i = \frac{occ(item_i)}{count(choices)} \tag{2}$$

where n is the number of unique items that has been engaged with by the users, and $occ(item_i)$ denotes the number of times the i-th item was engaged. As the

[1] `https://www.investopedia.com/terms/l/lorenz-curve.asp`

maximum value of H depends on the number of items n, H is normalised by dividing it with $log(n)$.

In the follow-up experiments, we clustered the dataset based on four most popular languages used on Twitter, i.e., English, Japanese, Spanish, and Portuguese. Our goal was to investigate if different languages of Twitter users may have any impact on the level of bias introduced by recommender algorithms. We have performed two different analyses (described with details later on), i.e., item-based analysis and user-based analysis.

For the item-based analysis, we clustered the data based on the language label provided for every individual tweet. For the user-based analysis, we needed to identify the language of every user. For that, we determined the language label of every user based on the language of the majority of her tweets. After the clustering, we have computed the above-described metrics on every language cluster.

3.3. Feature Extraction

The appropriate features needed to be determined prior to implementing various recommendation algorithms. Therefore, as part of the preprocessing stage, features were extracted from both the Twitter dataset and the MovieLens dataset. For Twitter, this resulted in 38 different features, which can be further divided into the following groups:

- Categorical Data: Features that determine which categories the tweet or the user belongs to. These include:
  - Engaging user verified status.
  - Engaged with user verified status.
  - Whether engaged with user follows engaging user or not.
  - Type of the tweet (TopLevel, Retweet, or Quote).
  - Whether there is media content (Video, GIF, Photo) present in the tweet or not.

  - Whether there is a question mark in the tweet or not.

  These categorical features were either boolean coded or one-hot encoded depending on the number of categories.

- Previous Interaction Data: Features that act like "flags" and indicate whether the user has previously engaged under certain conditions or not. These consist of the following alternatives:

  - Previous interaction with the language of the tweet.
  - Previous interaction with any of the hashtags present in the tweet.
  - Previous interaction with any of the domains present in the tweet (in cases where there are hyperlinks in the tweet).
  - Previous interaction with any of the media types (Video, GIF, Photo) present in the tweet.

- Previous Engagement Rate: If there is any previous information in the train set about a user, their previous engagement rate for each engagement type (Like, Reply, Retweet & Retweet with Comment) is included as an input to the recommender. The reason for doing so is that a user's past behavior is an indicator of their future action. For instance, a user who has a high reply rate in the training data is much more likely to reply to a new tweet than another user with a low reply rate. In cases where there is no past user data to extract this rate, global engagement rate (average of all users) is used instead.

- Quartile Data: Some of the numerical features included in the original dataset were difficult to deal with primitively. For example, the number of followers has a wide range, and simply scaling it to a predefined range of 0 to 1 results in a poor representation of the data due to the distant outliers. So instead, a one-hot encoding was used to indicate which quartile the sample belonged to in these cases. The following features were included using this method:

  - Engaging user follower count.
  - Engaging user following count.
  - Engaged with user follower count.
  - Engaged with user following count.

There was less feature extraction required for the MovieLens dataset, as it is much smaller both in the number of samples and features. The final pre-processed dataset contained 46 features with the following composition:

- Categorical Data: these features were either boolean or one-hot encoded:
  - User gender.
  - User age (7 groups).
  - User occupation (20 categories).
  - Movie genre (18 categories).

3.4. Recommendation algorithms

We implemented a set of popular recommender algorithms from two big classes of approaches, i.e., Collaborative Filtering (CB) and Content-based Filtering (CBF). While CBF focuses on leveraging the content descriptions of the items (e.g., genre), CF focuses on exploiting solely the customer feedback (e.g., ratings). Modern recommendation approaches utilize algorithms based on Artificial Neural Networks that are capable of learning from both item content and user feedback. We considered a diverse set of algorithms from these approaches, ranging from a more classical algorithm (e.g., k-NN) to a more modern algorithm (deep learning). These algorithms have been tested on both the Twitter dataset [26] and MovieLens dataset [30] with minor adjustments made to each algorithm due to the differences between two datasets. The full list of considered algorithms is the following:

- **Random** As a simple, worst-case baseline, the random algorithm classifies samples according to a uniform distribution, with no regard to the

provided input. Therefore, there is an equal chance an engagement is predicted or not as the output.

- **Personal Average** One of the simplest recommenders implemented, personal average predicts each user's mean engagement rate from past experience (train set) for each user-item pair in the test set. In cases where there is no past user experience (which is only 2.23% for Twitter and 0.1% for MovieLens), the global average (average of all user engagements) is used instead.

- **KNN** k-Nearest Neighbors is a commonly used model in classification especially in problems where there are clusters of data points. KNN classification works by first finding the k closest data points in the train set to the target sample (In our case k=3) and then determines a target's class by majority voting, meaning the class with most votes among neighbors is selected. Due to the large size of the dataset, an exact KNN approach was not tractable and therefore an approximate one [31] was used instead.

- **XGBoost** XGBoost [32] is one of the more elaborate models introduced so far. It is based on gradient boosting decision trees, and is widely used for classification tasks. While not being quite state-of-the-art compared to other intricate models used today as recommender systems, XGBoost does perform fairly well in recommendatino scenarios.

  The objective function that we have to minimize for XGBoost is the following:

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t), f_t \in F$$

  Where f_t is a random tree in F, the space of regression trees, and Ω is the regularization function used to prevent over-fitting. As stated in [32], this function *cannot be optimized using traditional optimization methods in Euclidean space* (since it is a function of functions). Instead, it is

suggested to use second-order approximation for optimizing the objective in the general setting:

$$L^{(t)} \approx \sum_{i=1}^{n} l[y_i, \hat{y}_i^{(t-1)} + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

After hyper-parameter optimization we used the following values:

- Learning Rate = 0.01
- Max Tree Depth = 15
- Number of Estimators = 150
- Trained for as long as there was no improvement in 10 consecutive epochs

- **MLP** Multi-layer Perceptrons are the foundation of deep neural networks. The architecture of this model is made of n layers, where each layer has m neurons and in each of these neurons there is a vector of weights and a bias. In addition, each layer has an activation function. There are three main steps involved in a neural network: forward passing, computing the error and error back-propagation.

  1. Forward pass: We have two main steps in each layer for calculating weighted inputs:

     $$Z_t = W_t * X_{t-1} + b_t$$

     With W_t being vector of weights at layer t, X_{t-1} being input for layer (or output of previous layer), and b_t being bias for layer t.

     Then we pass the Z vector to an activation function:

     $$y_t = f_t(Z_t)$$

     which is the output of this layer and input for the next layer.

2. Error computation: After completing the forward pass, the network error is calculated by a loss function (in our case, a binary cross-entropy function was used as our task was binary classification)

$$E_{total} = L(Y, T)$$

where Y is our network's output vector, T is the target vector, and L is the loss function.

3. Error Backpropagation: To optimize the network weights and biases through gradient descent, the computed error for each layer is propagated to its previous layer by using the chain rule. This way, we can propagate the error from our loss function in the last layer all the way back to network's first layer. Therefore, the rule for updating the weights is:

$$w_t^{new} = w_t^{previous} - \alpha * \frac{\partial E_{total}}{\partial w_t^{previous}}$$

$$\frac{\partial E_{total}}{\partial w_t^{previous}} = x_t * \delta_t$$

where α is the learning rate and δ_t is the error at layer t. Similar calculations could be trivially done for updating biases which are not given here.

The aforementioned steps are repeated until the loss function converges to a minimum.

After hyper-parameter optimization we used the following values:

- Learning Rate = 0.001
- Optimizer: Adam
- Batch Size = 64
- Network Architecture: 12 Layers (ReLU) + 1 Layer (Sigmoid)
- Regularization: L2 Norm

- **SVD++** Singular Value Decomposition is an algorithm is equivalent to Probabilistic Matrix Factorization when baselines are not used. SVD is mostly used for recommender systems based on ratings.

  The SVD prediction r^{ui} is defined with

  $$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

  where p_u are user factors, q_i are item factors, b_u are user biases and b_i are item biases.

  To estimate all the unknowns we minimized the following regularized squared error

  $$\sum_{r^{u_i} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2)$$

  The SVD++ algorithm, an extension of SVD, uses also implicit ratings:

  $$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j)$$

  where the y_j terms are a new set of item factors that capture implicit ratings. Here, an implicit rating describes the fact that user u rated item j, regardless of the rating value. If user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. The same applies for item i with b_i, q_i and y_i.

  We used the following hyper-parameters:

  - Learning Rate = 0.001
  - Trained for 30 epochs

- **DeepFM** [4] combines Deep Neural Networks (DNN) and Factorization Machines (FM) [33] in order to model both low- and high-order interactions among features. DeepFM consists of 2 major parts:

- FM part, which is responsible for capturing low-order interactions: linear (order-1) and pairwise (order-2) interactions.
- DNN part, which is a deep feed-forward neural network that extracts high-level feature interactions.

DeepFM is similar to the Wide & Deep models [34], but the difference is that DeepFM uses a shared embeddings layer, which eliminates the need for manual feature extraction required for the FM module. The output of DeepFM is computed in the following way:

$$\hat{y} = sigmoid(y_{FM} + y_{DNN})$$

where y_{FM} is the output of the FM module and y_{DNN} is the output from the DNN module.

It should be noted that, we used the implementation of DeepFM provided by the authors in [4], using the following parameters:

- Learning Rate = 0.002
- Batch Size = 64
- Trained for 200 epochs

3.5. Threshold Predictions

Some of the methods used (e.g., personal average, MLP and SVD++) may require a classification threshold as they are, in their nature, regression models that return probability values as the output. However, the utilized datasets and the tasks that we tackled are primarily focused on classification. In order to determine the optimal classification threshold, we proceeded as follows. First, the rate of recommendations made in the train set was determined for each engagement type (i.e., Like, Reply, etc.). Next, the threshold was selected so that the recommendation rate of each engagement type in the test set would match the recommendation rate of the same engagement type in the train set. This way, it was assured that the distribution of recommendations in both the train and test set stays as close to each other as possible.

Finally, we have performed the experiments using a server machine running Ubuntu 16.04 equipped with an *Intel XEON E5 2697 V3 CPU* clocked at 2.6 GHz with 128 GB of DDR3 RAM.

4. Results

4.1. Experiment A: Exploratory and Performance Analysis

In the first set of experiments, we have explored the data in order to obtain an overall view on the characteristics of the data. This has been followed by the evaluation of the performance for different recommender algorithms. This helped us to compare these algorithms with respect to accuracy in different application domains (i.e., micro-blogging and movie domains).

Figure 2: Word Cloud in English (left), Word Cloud in Spanish (right)

Initially, we have adopted NLP [2] techniques, such as stop-word removal and *tf-idf*, in order to pre-process and visualize the content of the tweets. After these steps, we have computed the word cloud of the tweets in different languages. Figure 2 shows the word cloud for the two most popular languages, English and Spanish, where, for the sake of readability, we have translated the Spanish words into English. We have noticed that, despite the expected similarities, there are considerable differences among the most popular (and unique) words, used in the tweets within different languages.

[2]Natural Language Processing

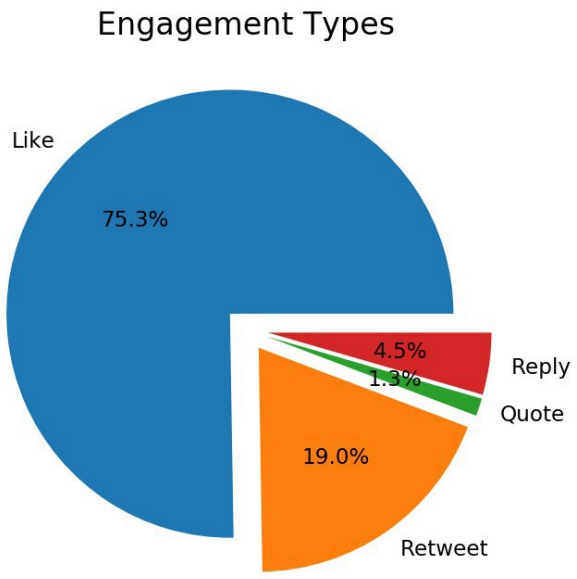

Figure 3: Pie chart that represents the histogram of the engagements

We have further analyzed the Twitter dataset in terms of user engagements, which includes four classes of actions, *Like*, *Reply*, *Retweet*, and *Quote*. The histogram of the engagement classes is plotted in Figure 3, illustrated as a pie chart. As it can be seen, there is a big imbalance among different classes of engagements. The largest portion of data belongs to Like class which accounts for 75.3 %. The next class is Retweet with 19.0%, Reply with 4.5% and Quote with 1.3%. Such a large imbalance may also justify the need for consideration of Relative Cross Entropy (RCE) as an important metric in the experiments.

In the next experiment, we have compared the performance of different recommender algorithms in terms of PR-AUC and RCE metrics. The results are presented in Table 1. As it can be seen, the best overall performance has been observed for XGBoost followed by one of our deep learning techniques (MLP). For the Twitter dataset, in terms of PR-AUC (i.e., Area Under the Precision-Recall Curve), the XGBoost algorithm obtained the value of 0.83 (predicting Likes), 0.62 (predicting Replies), and 0.62 (predicting Retweets). When pre-

Table 1: The comparison of the performances of different recommender algorithms. **TW**: Twitter dataset, **ML**: MovieLens dataset.

		PR-AUC					RCE				
	Data	**XGB**	**MLP**	**KNN**	**SVD++**	**DeepFM**	**XGB**	**MLP**	**KNN**	**SVD++**	**DeepFM**
TW	*Like*	**0.83**	0.71	0.70	0.43	0.70	-881.41	**0.00**	-1551.21	-0.65	-2695.98
	Reply	**0.62**	0.51	0.36	0.02	0.46	-416.44	**-10.93**	-669.62	-26.10	-26547.00
	Retweet	**0.62**	0.55	0.48	0.11	0.41	-700.37	**-0.11**	-967.79	-6.11	-8893.43
	Quote	0.48	**0.50**	0.21	0.01	0.24	-436.72	**-31.11**	-540.40	-32.44	-74613.12
ML	*Rating*	**0.28**	0.25	0.05	0.25	-	1.37	1.20	**7.01**	1.81	-

dicting quotes, MLP obtained a slightly better result, 0.50. In terms of RCE (i.e., Relative Cross Entropy), the MLP algorithm obtained the value of 0.00 (predicting Likes), -10.93 (predicting Replies), -0.11 (predicting Retweets) and -31.11 (predicting Quotes). For the Movielens dataset, the best results have been achieved by the XGBoost and KNN algorithms. While XGBoost obtained the best results for PR-AUC (0.28), KNN obtained the best value for RCE (7.01).

These results clearly indicate the superior accuracy of recommendation based on XGBoost (an optimized distributed gradient boosting method) and MLP (a deep learning technique). While we have evaluated the performances of these algorithms, the main focus of the paper will be reporting the results of the next experiments, measuring the impact of these recommender algorithms on the popularity bias.

4.2. Experiment B: User-based and Item-based Analysis

In the second set of experiments, we wanted to address the research questions RQ1 and RQ2. Hence, we were interested in investigating the level of bias originated from different recommender systems when operating in different application domains (micro-blogging and movie). In addressing the RQ1, we have considered two different recommendation scenarios, i.e., (i) when the systems recommends items to a target user compared to (ii) when the system recommends users to a target user. Examples of the former scenario can be Twitter recommending tweets to a target user to read or YouTube generating recommendation of videos for user to watch. Examples of the latter scenario

can be Twitter recommending user to user to follow or YouTube recommending channels of the video makers to users to subscribe.

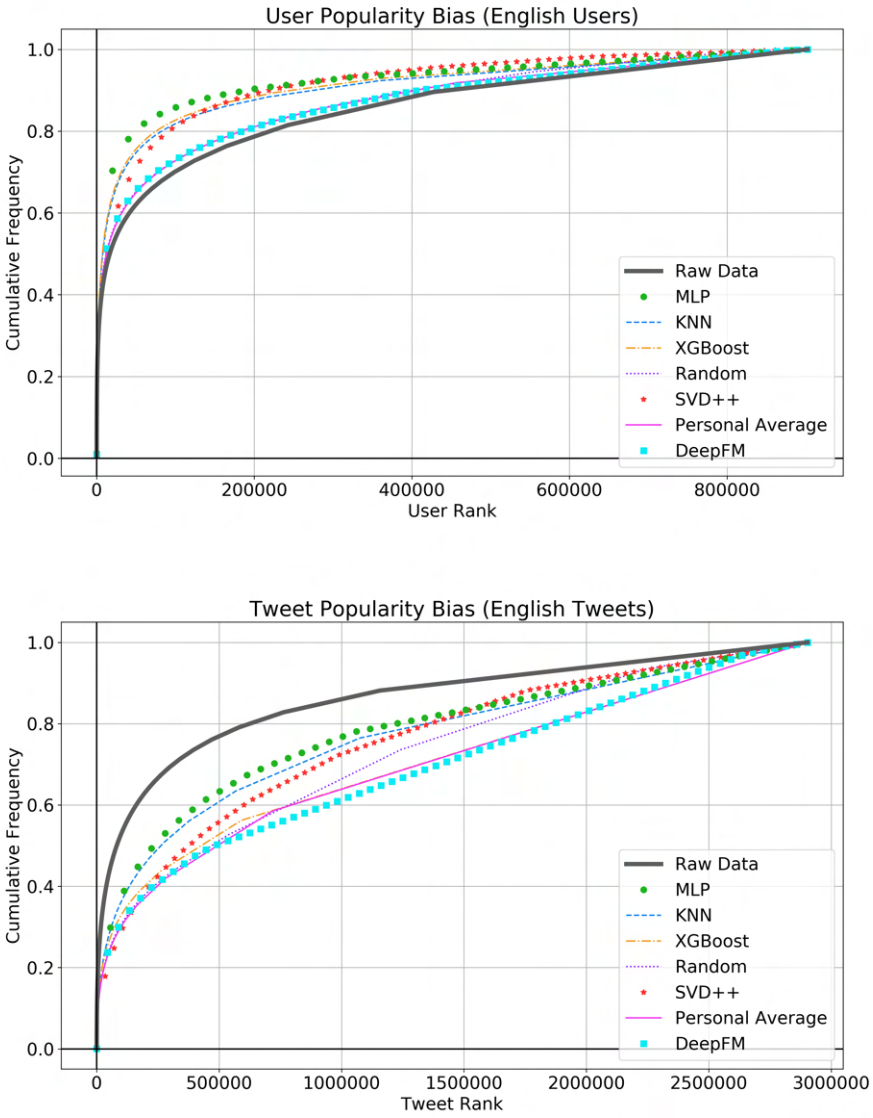

Figure 4: **Twitter** Dataset: User-based analysis of bias (top), Item-based analysis of bias (bottom)

The results are illustrated in Figures 4 and 5. The former figure has been made by computing the cumulative frequency of user engagement for different items (tweets) or users (Twitter users) at a given rank in the Twitter dataset. The latter figure has been made by computing the cumulative frequency of ratings for different items (movies) or users (movie directors) at a given rank in the Movielens dataset. In both of the figures the x-axis represents the rank of each item (or user) when sorted according to their popularity. Hence, the more popular items (or users) are found in the left side and the more unpopular items (or users) are found in the right side of the x-axis. The y-axis represents the cumulative prevalence of the items (or users) found in the data. It can be clearly seen in the figures that a few highly ranked items (or users) dominantly receive the largest portion of user engagements (in the Twitter dataset) or user ratings (in the Movielens dataset).

Considering the results of Twitter (Figure 4), as it can be seen, when a system recommends users to a target user, i.e., (e.g., recommending whom to follow), all different types of recommender algorithms may similarly impact the system by increasing the popularity bias compared to the level of bias in the raw data. This can be observed in Figure 4-top, where the raw data, provided by Twitter, is illustrated with thick gray color and the recommender algorithms are illustrated with different colors and styles. The figure clearly shows that all algorithms may increase the popularity bias. This basically means that the popular users, with a large number of followers, will become even more popular. The worst results have been obtained by recommendations based on the deep learning algorithm (MLP), depicted with green color.

In contrast to this, as it can be seen in Figure 4-bottom, when the system generates recommendations of items (tweets) for users, i.e., which tweets to read and engage with, the popularity bias gets decreased. Consequently, this basically means that the unpopular items (tweets) could gain more popularity and hence been seen by larger number of users. This surprising phenomenon can be observed for all algorithms depicted with different colors.

Considering the Movielens results, illustrated in Figure 5, the above men-

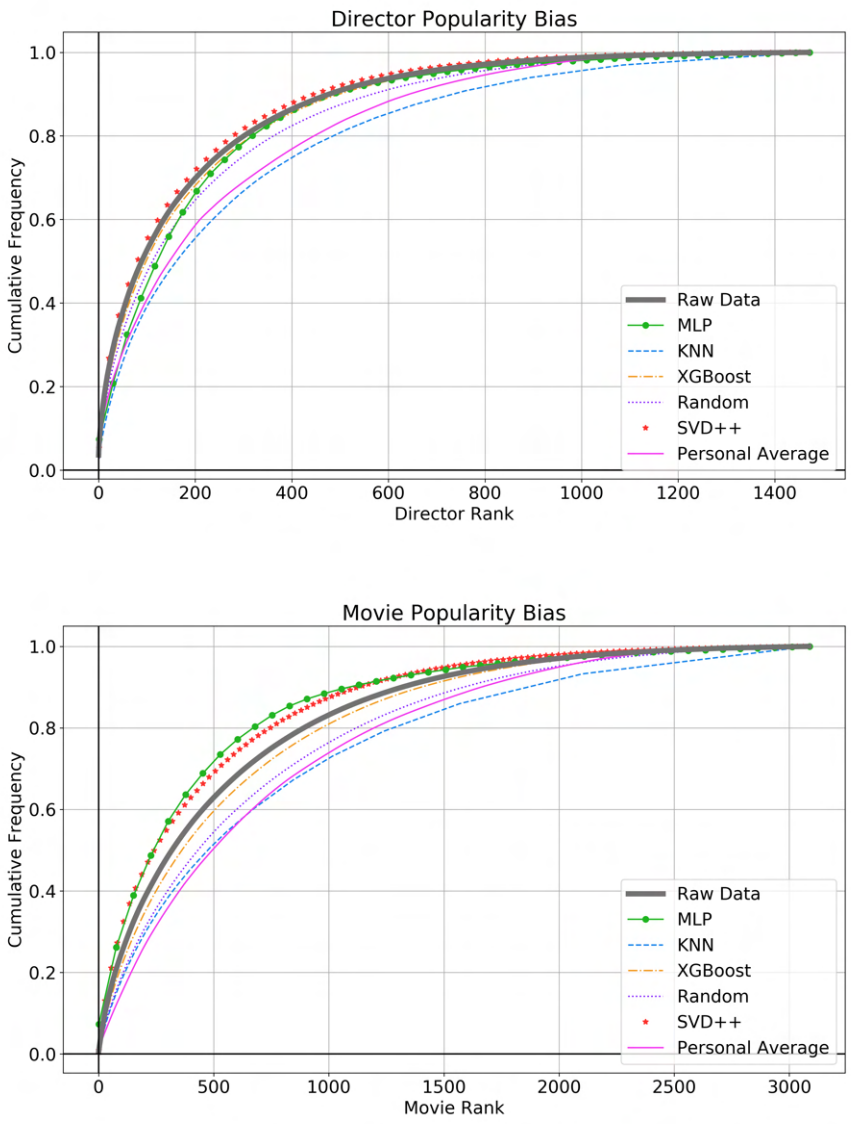

Figure 5: **Movielens** Dataset: User-based analysis of bias (top), Item-based analysis of bias (bottom)

tioned phenomenon has not been observed. Hence, the substantial differences between user-based recommendation and item-based recommendation scenarios

(reported for Twitter dataset) have not been observed in Movielens dataset. When considering recommendation of users to users (e.g., recommending a movie director to users to follow), shown in Figure 5-top, some of the algorithms may increase the popularity bias (e.g., SVD++), compared to the raw data, and some may decrease it (e.g., KNN). Similarly, considering the recommendation of items to users (e.g., recommending movies to users to watch), shown in Figure 5-bottom, again some of the algorithms may increase the popularity bias (e.g., MLP), compared to the raw data, and some may decrease (e.g., KNN). This is an interesting observation and it demonstrates that the impact if recommender algorithms on popularity bias can strongly depend on the characteristics of the collected data, recommendation scenario, and other particularities of the application domain.

4.3. Experiment C: Language Impact on Popularity Bias

In the third set of experiments, we addressed RQ3 by analysing the Twitter data in different languages and comparing them against each other. We were interested in investigating whether or not the recommender algorithms that use data in different languages may exhibit differences in the level of popularity bias.

In order to compare the popularity bias in different languages, we considered two well-known metrics, the Gini index and the Entropy score, which have been computed for the raw data in a particular language as well as for the recommendations generated by different algorithms using the data.

Figure 6 represents the results for the Gini index. First of all, as it can be seen, the raw data in different languages have different levels of impact on the Gini values. Considering the user-based experimental scenario (Figure 6-top), the level of popularity bias in the raw data is 72.04 for the English language while this is 68.01 for Portuguese, 65.43 for Spanish and 60.55 for Japanese. This indicates that the difference between the popularity of highly followed users and lowly followed users is higher for the English language than for the other languages. The lowest such difference has been observed for the Japanese

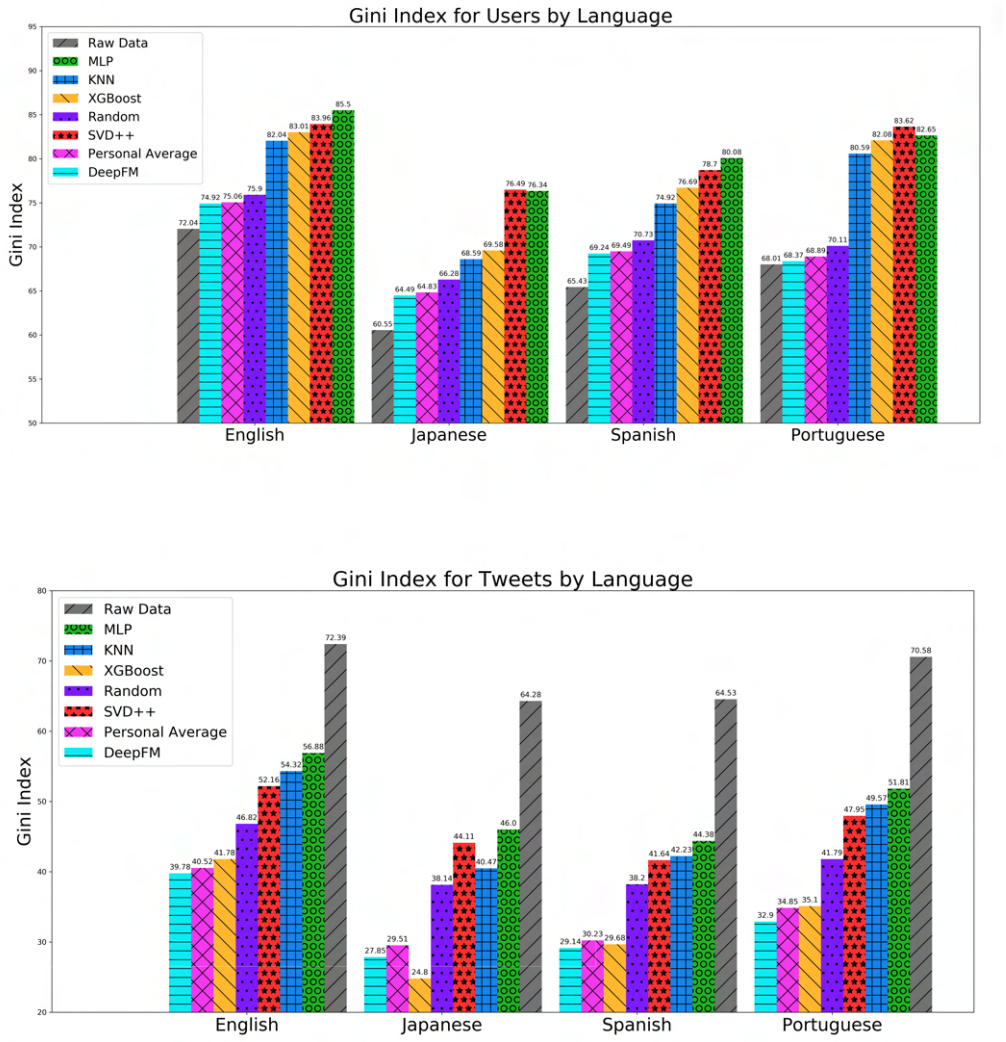

Figure 6: Comparison of **Gini** scores of different users who tweet in different languages (top), and different tweets in different (bottom)

language. The same trend can be observed when comparing the output of different recommender algorithms. Again, recommendation of users to users (e.g., recommending whom to follow) creates more popularity bias in English

language compared to the other languages.

Considering the item-based experimental scenario (Figure 6-bottom), the differences among the languages is slightly lower than in the user-based scenario while the order of the languages is maintained. Accordingly, the level of popularity bias in the raw data is 72.32 for English language while this is 70.58 for Portuguese, 64.53 for Spanish and 64.28 for Japanese. This is an indication that the difference between the popularity of highly engaged tweets (i.e., items) and lowly engaged tweets is much higher for the English language than for the other languages. The lowest such difference has been observed for the Japanese language.

This is an interesting observation and may need a discussion. First of all, it is important to note that the Twitter users who tweet in a single language may not necessarily come from one country. Hence, for some languages, such as English, the range of users who generate tweets in this language could be very wide.On the other hand, for some other languages, such as Japanese, the users who generate tweets may come from a smaller set of countries (mainly Japan). This might be an explanation why the observation for these languages differ. Moreover, there might be some other factors that causes such a difference in the observation. This may include societal and cultural aspects that may certainly influence the behaviour of the users in the online platforms such as Twitter. While this can be of high interest for further investigations it is beyond the scope of this research.

As a follow-up to the described experiments, we have also compared the differences in languages considering the entropy scores, computed for the raw data as well as for the recommendations generated by different algorithms. Figure 7 shows the results. Again, we observed difference among languages in terms of entropy scores. Figure 7-top shows the user-based experimental scenario and reports the level of entropy in the raw data to be 16.84 for English language, 17.14 for Japanese, 15.79 for Spanish, and 14.80 for Portuguese. In terms of item-based popularity bias (Figure 7-bottom), the raw data of English language exhibits the largest entropy score. This value is 19.01, while it is 18.55 for

Japanese, 17.74 for Spanish, and 16.71 for Portuguese. These are considerably higher entropy scores compared to the user-based analysis of these languages. Hence, the difference between the popular users (i.e., highly followed) and unpopular users (i.e., lowly followed) is higher than the difference between popular tweets (i.e., highly engaged) and unpopular tweets (i.e., lowly engaged).

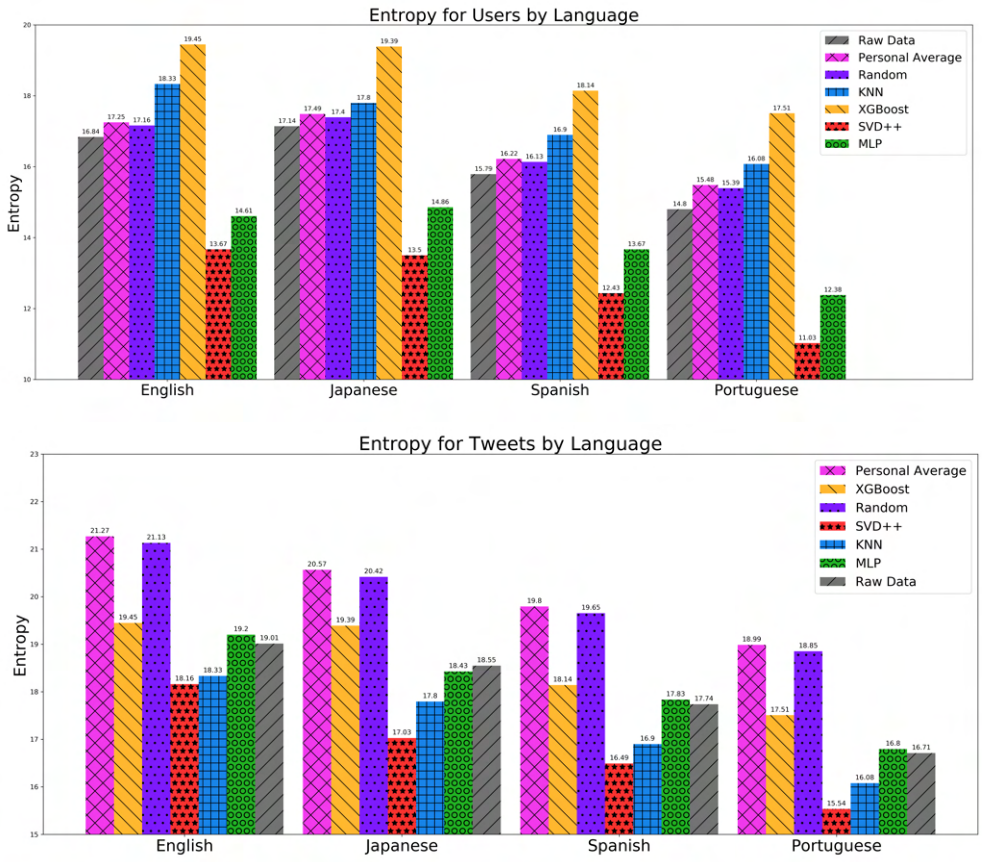

Figure 7: Comparison of **Entropy** scores of different users who tweet in different languages (top), and different tweets in different (bottom)

In these experiments we have observed a very similar phenomenon as the one discussed before: substantial difference between the user-based experimental scenario, i.e., recommending users to users to follow, in comparison to the item-based experimental scenario, recommending tweets to users to engage. In-

terestingly, such a difference is consistently observed in all the analyzed languages. This may confirm that our reported results do not depend on the raw data (tweets) collected in a particular language or particular users (twitters) whom generate the tweets in that language and might be observed when measuring different metrics.

4.4. Discussion

Previous research (e.g. [13]) showed that applying recommendation algorithms to datasets intensifies the existing bias. We were wondering if this conclusion is generalizable, hence the main research questions addressed by this work were two-dimensional: (i) does the application of recommendation algorithms intensify or reduce the pre-existing bias in the input datasets and (ii) how is this intensification/reduction dependant across users-vs-items, different domains (movies, micro-blogging), and languages. The short answer is that the bias is sometimes intensified and sometimes reduced by the recommender algorithm, hence making the claim of universal bias intensification not generalizable. When looking at our results in more depth we can observe the following: (a) in the micro-blogging domain, when recommending users to users (e.g. which users to follow), the recommender algorithms intensify the pre-existing bias, (b) in the micro-blogging domain, when recommending items to users (e.g. which tweets to read) the recommender algorithms decrease the pre-existing bias, (c) in the movies domain, there was no clear conclusion as some algorithms increased and some decreased the pre-existing biases, both in the user-user recommendation scenario and the item-user recommendation scenario, (d) in the micro-blogging domain, the pre-existing biases are different in different languages, (e) across multiple languages some algorithms increase and some decrease the bias.

Our main message, that the application of recommender systems sometimes intensifies and sometimes decreases the pre-existing bias, has many implications, among which we highlight two: (i) for practitioners, it is important not to assume that a recommendation algorithm will increase the bias but to verify it as we have shown that sometimes this is not the case and (ii) for researchers, it

is important to conduct further studies to investigate the reasons for what now appears to be unpredictable behaviour of recommender algorithms in terms of increasing/decreasing the bias and to devise methods for mitigating these changes.

5. Conclusion

In this paper we have investigated whether recommendation algorithms increase the bias, keep the bias unchanged, or reduce the bias. We have designed a comprehensive evaluation and considered two experimental scenarios, i.e., user-based and item-based scenario. An example of the former is the recommendation of users to users and an example of the latter can be the recommendation of items to users. We have used two well-known datasets, Twitter and Movielens, and compared a wide range of classical (e.g., KNN) and modern recommender algorithms (deep learning). The comparison has been made with respect to diverse forms of metrics, i.e., precision-recall area under curve (AUC), relative cross entropy (RCE), Gini index, and entropy Score.

The results have shown a substantial difference between the above mentioned scenarios and hence the recommendation of users to users may have a completely different impact on popularity bias compared to recommendation of items of users. We have also analyzed this phenomenon considering the Twitter data. The results have shown that, despite the difference of languages in terms of popularity bias, the above-noted phenomenon can be consistently found in all the languages considered.

For further work we plan to extend our evaluation by considering a larger dataset including more languages. We would like to repeat the analyzes considering less popular languages. In addition to that, we plan to perform a real user study where the recommendation generated by different algorithms are compared by users in terms of popularity bias.

6. Acknowledgements

This work was supported by industry partners and the Research Council of Norway with funding to MediaFutures: Research Centre for Responsible Media Technology and Innovation, through The Centres for Research-based Innovation scheme, project number 309339.

References

[1] M. Karimi, D. Jannach, M. Jugovac, News recommender systems–survey and roads ahead, Information Processing & Management 54 (6) (2018) 1203–1227.

[2] M. S. Pera, Y.-K. Ng, A group recommender for movies based on content similarity and popularity, Information Processing & Management 49 (3) (2013) 673–687.

[3] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 639–648.

[4] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: a factorization-machine based neural network for ctr prediction, arXiv preprint arXiv:1703.04247.

[5] M. Elahi, M. Braunhofer, T. Gurbanov, F. Ricci, User preference elicitation, rating sparsity and cold start. (2018).

[6] F. Ricci, L. Rokach, B. Shapira, Recommender Systems: Introduction and Challenges, in: Recommender Systems Handbook, Vol. 54, Springer US, Boston, MA, 2015, pp. 1–34. arXiv:arXiv:1011.1669v3, doi:10.1007/978-1-4899-7637-6_1.
URL http://www.springerlink.com/index/10.1007/978-0-387-85820-3http://link.springer.com/10.1007/978-1-4899-7637-6{_}1

[7] D. Margaris, C. Vassilakis, D. Spiliotopoulos, What makes a review a reliable rating in recommender systems?, Information Processing & Management 57 (6) (2020) 102304.

[8] M. Elahi, H. Abdollahpouri, M. Mansoury, H. Torkamaan, Beyond algorithmic fairness in recommender systems, in: Adjunct Publication of the ACM Conference on User Modeling, Adaptation and Personalization, 2021.

[9] J. Ugander, B. Karrer, L. Backstrom, C. Marlow, The Anatomy of the Facebook Social Graph (2011) 1–17arXiv:1111.4503.
URL http://arxiv.org/abs/1111.4503

[10] A. J. Chaney, B. M. Stewart, B. E. Engelhardt, How algorithmic confounding in recommendation systems increases homogeneity and decreases utility, arXivarXiv:1710.11214, doi:10.1145/3240323.3240370.

[11] R. Baeza-yates, Bias in Search and Recommender Systems, in: RecSys '20: Fourteenth ACM Conference on Recommender Systems, 2020. doi:10.1145/3383313.
URL https://dl.acm.org/doi/fullHtml/10.1145/3383313.3418435

[12] R. Baeza-Yates, Bias on the web, Communications of the ACM 61 (6) (2018) 54–61.

[13] M. Mansoury, H. Abdollahpouri, M. Pechenizkiy, B. Mobasher, R. Burke, Feedback loop and bias amplification in recommender systems (2020). arXiv:2007.13019.

[14] G. Adomavicius, J. Bockstedt, S. Curley, J. Zhang, De-Biasing User Preference Ratings in Recommender Systems, in: Proceedings of the Joint Workshop on Interfaces and Human Decision Making for Recommender Systems co-located with ACM Conference on Recommender Systems (RecSys 2014), 2014.
URL http://ceur-ws.org/Vol-1253/

[15] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning, arXiv preprint arXiv:1908.09635.

[16] R. Baeza-Yates, D. Saez-Trumper, Wisdom of the crowd or wisdom of a few?, Proceedings of the 26th ACM Conference on Hypertext & Social Media - HT '15doi:10.1145/2700171.2791056.
URL http://dx.doi.org/10.1145/2700171.2791056

[17] S. Wu, J. M. Hofman, W. Mason, D. J. Watts, Who says what to whom on twitter, in: Proceedings of the 20th International Conference on World Wide Web, 2011.

[18] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, X. He, Bias and debias in recommender system: A survey and future directions, arXiv preprint arXiv:2010.03240.

[19] J. Chen, Y. Feng, M. Ester, S. Zhou, C. Chen, C. Wang, Modeling users' exposure with social knowledge influence and consumption influence for recommendation, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 953–962.

[20] X. Wang, Y. Wang, D. Hsu, Y. Wang, Exploration in interactive personalized music recommendation: A reinforcement learning approach (2013). arXiv:1311.6355.

[21] G. Adomavicius, J. C. Bockstedt, S. P. Curley, J. Zhang, Reducing recommender system biases: An investigation of rating display designs, MIS Quarterly 43 (5) (2019) 1321–1341. doi:10.25300/MISQ/2019/13949.

[22] Connecting user and item perspectives in popularity debiasing for collaborative recommendation, Information Processing & Management 58 (1) (2021) 102387.

[23] H. Abdollahpouri, R. Burke, B. Mobasher, Managing popularity bias in recommender systems with personalized re-ranking (2019). arXiv:1901.07555.

[24] H. Abdollahpouri, R. Burke, B. Mobasher, Controlling popularity bias in learning-to-rank recommendation, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, pp. 42–46.

[25] P. Nesi, G. Pantaleo, I. Paoli, I. Zaza, Assessing the retweet proneness of tweets: predictive models for retweeting, Multimedia Tools and Applications (77) (2018) 26371–26396. doi:https://doi.org/10.1007/s11042-018-5865-0.

[26] L. Belli, S. I. Ktena, A. Tejani, A. Lung-Yut-Fon, F. Portman, X. Zhu, Y. Xie, A. Gupta, M. Bronstein, A. Delić, G. Sottocornola, W. Anelli, N. Andrade, J. Smith, W. Shi, Privacy-aware recommender systems challenge on twitter's home timeline (2020). arXiv:2004.13715.

[27] R. Dorfman, A formula for the gini coefficient, The review of economics and statistics (1979) 146–149.

[28] D. Saha, A. R. Kemanian, F. Montes, H. Gall, P. R. Adler, B. M. Rau, Lorenz curve and gini coefficient reveal hot spots and hot moments for nitrous oxide emissions, Journal of Geophysical Research: Biogeosciences 123 (1) (2018) 193–206.

[29] Z. Szlávik, W. Kowalczyk, M. Schut, Diversity measurement of recommender systems under different user choice models, in: Fifth International AAAI Conference on Weblogs and Social Media, 2011.

[30] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (4). doi:10.1145/2827872.
URL https://doi.org/10.1145/2827872

[31] E. Bernhardsson, Annoy: Approximate Nearest Neighbors in C++/Python, python package version 1.13.0 (2018).
URL https://pypi.org/project/annoy/

[32] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowl-

edge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 785–794. doi:10.1145/2939672.2939785.
URL http://doi.acm.org/10.1145/2939672.2939785

[33] S. Rendle, Factorization machines, in: 2010 IEEE International Conference on Data Mining, 2010, pp. 995–1000. doi:10.1109/ICDM.2010.127.

[34] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st workshop on deep learning for recommender systems, 2016, pp. 7–10.

Author Statement

Please find the following author's contributions.

- Mehdi Elahi: Supervision, Formal analysis, Conceptualization, Methodology, Writing - Original Draft, Writing - Review & Editing
- Danial Khosh Kholgh: Software, Investigation, Validation, Visualization
- Mohammad Sina Kiarostami: Conceptualization, Resources, Investigation, Writing - Reviewing and Editing
- Sorush Saghari: Software, Investigation, Validation
- Shiva Parsa Rad: Conceptualization, Writing - Reviewing and Editing
- Marko Tkalcic: Supervision, Formal analysis, Conceptualization, Writing - Reviewing and Editing