

The 2021 RecSys Challenge Dataset: Fairness is not optional

Luca Belli, Alykhan Tejani*, Frank Portman*, Alexandre Lung-Yut-Fong*,
Ben Chamberlain, Yuanpu Xie, Kristian Lum, Jonathan Hunt,
Michael Bronstein, Vito Walter Anelli, Saikishore Kalloori,
Bruce Ferwerda, Wenzhe Shi

Abstract

After the success the RecSys 2020 Challenge, we are describing a novel and bigger dataset that was released in conjunction with the ACM RecSys Challenge 2021. This year’s dataset is not only bigger (~1B data points, a 5 fold increase), but for the first time it take into consideration fairness aspects of the challenge. Unlike many static datasets, a lot of effort went into making sure that the dataset was synced with the Twitter platform: if a user deleted their content, the same content would be promptly removed from the dataset too. In this paper, we introduce the dataset and challenge, highlighting some of the issues that arise when creating recommender systems at Twitter scale.

1 Intro

Recommender systems are a central component of modern social media platforms. On Twitter, when a user logs in, they are presented with a series of Tweets (the Home Timeline). Users can decide if the content they are shown is sorted in reverse chronological order or is algorithmically ranked by a recommender system. If they opt for the latter, content is presented in decreasing (predicted) order of value for the user: the Tweets that the recommender system predict are more relevant for the reader are put in a higher position.

For users who select algorithmically ranked Home Timelines, the quality of the recommender system’s rankings directly impacts the quality of their experience on the platform. From the point of view of the person who is receiving the Tweets, a well-performing recommender system may result in a more interesting or enjoyable experience on the platform. It may also offer opportunities to participate conversations they would like to contribute to or access to important news, perspectives, or connections to new communities they would like to be a part of. From the point of view of the Tweet authors, the recommender system impacts the extent to which their Tweets reach a receptive audience. A better job reaching that audience can lead to more social influence.

Given the impacts of recommender systems in social media, it is important that they work well for all users, not just some subset. That is, it is important that recommender systems behave “fairly” by some definition of the word. Recent years have seen a flurry of methodological work in “fair ML.”, e.g. [4]. Though there has been high quality work developing notions of fairness for recommender systems (see, e.g., [1, 3, 9, 11]), fairness in classification tasks has received the bulk of the attention. The existence of a publicly available dataset with a pre-defined “fairness task”—the COMPAS dataset—undoubtedly played no small part in spurring research into fair classification. In hopes of providing a similar resource to researchers interested in developing state-of-the-art methods for fairness in recommender systems at real scale, in this paper we introduce a fairness challenge and accompanying dataset comprised of Twitter data.

In Section 2 we formally introduce the task of the challenge, explaining the differences between the challenge and Twitter’s product usage. In Section 3 describes the challenge’s dataset. Finally in Section

*Authors contributed to the research equally

4 we discuss the 2021 RecSys Challenge in detail, providing details on how submissions are evaluated and the baseline that we provided.

2 Problem Definition

Twitter is what’s happening. When people log in on Twitter, they are presented an updating display of Tweets— content produced by other users. Twitter users indicate the content they would like to see in multiple ways: users can follow other users, they can follow list of multiple users, or they can follow specific Topics (e.g. “Sports”, “Art”). Tweets from the users or topics they follow can appear on their timeline.

Users can decide if this timeline is sorted by reverse chronological order or if it is algorithmically ranked. If they opt for the former, their timeline consists of all the latest Tweets, arranged so that the newest appears first. If they opt for the latter, content is presented in decreasing (predicted) order of value for the user: the Tweets that our models predict are more relevant for the reader are put in a higher position. Given the fast pace of Twitter and the speed with which the conversation changes, it is very important that the most relevant and timely content is presented to the reader.

For the purposes of the RecSys 2021 Challenge, we define the core problem for this challenge as follows:

Given a (reader, Tweet) pair, “fairly” predict the probability that the reader will engage with the Tweet.

We define engagement as one of four different types: *Like* (reader clicks the Like button), *Reply* (reader replies to the Tweet, thus creating a new Tweet themselves), *Retweet* (reader shares the original Tweet with their followers) and *Quote* (reader adds their comment or media before sharing the original Tweet). We define what we mean by “fairly” in section 4.

3 The Data

The data for the challenge represents a snapshot of a 4 week period on public users on Twitter. The first 3 weeks were used for the training set and the last week was used for testing and validation. We were aiming for approximately 1 billion records, equally balanced between positives and (pseudo)negatives. Figure 1 describes the data generation process for the challenge.

3.1 Features

Features available for prediction are divided into three separate feature groups: *user-*, *Tweet-* and *engagement features*. There are two instantiations of *user features*, one for the author (producer) and one for the reader (consumer) of the Tweet. *Tweet features* groups all the attributes describing the original Tweet, that is possibly engaged with by the consumer. Tweet tokens represent the text of the Tweet. We wanted to find a reasonable trade-off between obfuscation (i.e. including plain text would have made re-identification trivial) and making sure that the content of the Tweet would be available for NLP tasks. As in the previous year, we released the text in form of the BERT tokens: after tokenizing the text, we release the list of the BERT IDs for each Tweet.

Finally, the *engagement features* contain all the details of the engagement itself. These are the outcomes to be predicted. The dataset features are described in detail in Table 1.

3.2 Sampling

To create a useful dataset for a supervised learning task, we need a mixture of both positive and negative samples. On Twitter, engagements for public profiles are public so the positive case is straightforward. On the other hand, the converse is not true. In this case a negative sample is a *(reader, Tweet)* that was seen, but not engaged with. This information is not publicly available: authors and other browsers of Twitter do not know which users *saw* which Tweet. Therefore releasing “true” negatives would have constituted releasing nonpublic information about our users.

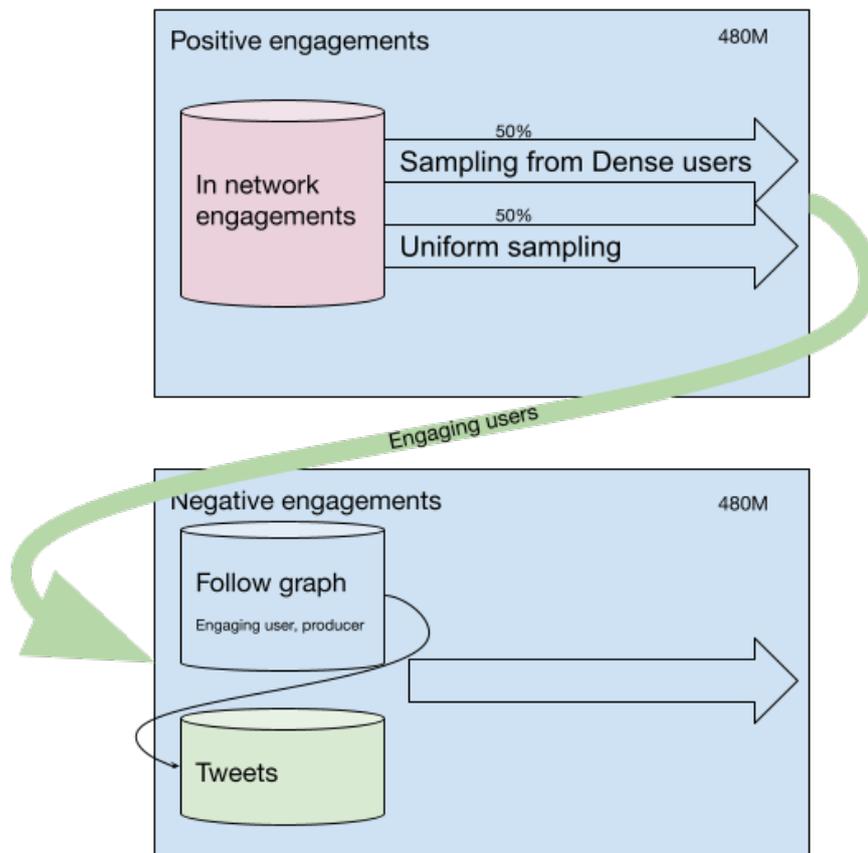


Figure 1: Data generation sampling pipeline

For the negative features we did the following: for each reader, we collected the public Tweets that were produced by their followees during the sampling period. Some of those might have been (publicly) engaged with, so we removed them from this pool (they may separately have been in the positive sample). The rest of Tweets were Tweets that were not engaged with, but here is the catch: *this group contains both Tweets that were seen and not seen by the reader*. Negative examples were sampled from this group. We call the collection of negatives “pseudonegatives”, since they represent tweets that were not engaged with, though the user may or may not have had the opportunity to engage with them.

It is very likely that a model trained on this dataset would under-perform with an equivalent model trained on a set of pure negative samples. Since the goal of this challenge is not to implement the solutions in the production case, the loss of predictive performance is easily worth not revealing private user information. Note that special care was taken to make sure the readers in the negative set were the same users that were in the positive set, to avoid the cold start problem.

3.3 Data set release conditions

Before having access to the dataset, participants are required to be approved for the use of Twitter’s API, and thus adhere to the Developer Agreement and Policy which includes (but is not limited to) the restrictions to “off-Twitter matching” to data that has been directly provided by the person and/or public data. This policy serves to prevent misuse of the dataset, such as inference attacks on private

| Feature | Name | Signature | Description |
|---------------------------|------------------------------|---|---|
| User | userId | <i>string</i> | User identifier (hashed) |
| | follower count | <i>int</i> | Number of followers of the user |
| | following count | <i>int</i> | Number of accounts this user is following |
| | is verified | <i>bool</i> | Is the account verified? |
| Tweet | account creation time | <i>int</i> | timestamp of the creation time of the account |
| | tweetId | <i>string</i> | Tweet identifier (hashed) |
| | presentMedia | <i>list[string]</i> | Tab-separated list of media types; media type can be in (Photo, Video, Gif) |
| | presentLinks | <i>list[string]</i> | Tab-separated list of links included in the tweet (hashed) |
| | presentDomains | <i>list[string]</i> | Tab-separated list of domains (e.g. twitter.com) included in the tweet (hashed) |
| | tweetType | <i>string</i> | Tweet type, can be either Retweet, Quote, Reply, or Toplevel |
| | language | <i>string</i> | Identifier corresponding to inferred language of the tweet |
| | tweet timestamp | <i>int</i> | Unix timestamp, in seconds of the creation time of the Tweet |
| Engagement | tweet tokens | <i>list[int]</i> | Ordered list of Bert ids corresponding to Bert tokenization of Tweet text |
| | tweet hashtags | <i>list[string]</i> | Tab-separated list of hashtags present in the tweet |
| | reply engagement timestamp | <i>int</i> | timestamp of the Reply engagement if one exists |
| | retweet engagement timestamp | <i>int</i> | timestamp of the Retweet engagement if one exists |
| | quote engagement timestamp | <i>int</i> | timestamp of the Quote engagement if one exists |
| like engagement timestamp | <i>int</i> | timestamp of the Like engagement if one exists | |
| engageeFollowsEngager | <i>bool</i> | Does the account of the engaged tweet author follow the account that has made the engagement? | |

Table 1: List of features provided for the challenge dataset

datasets.¹

Over time, a user might decide to delete one or more of their Tweets, or decide to delete their profile (or make it private). Again, since we want the dataset to only contain public data at all times, the dataset itself is constantly updated to track what is on the Twitter platform. Specifically this means that the dataset is shrinking as we are not adding new content over time, only deleting data that is not available anymore. Furthermore participants are required to keep their dataset up-to-date as required in the Developer Agreement and Policy and we provide a file that includes only details on the content that needs to be scrubbed.

This of course means that people accessing the dataset at different time will have access to different datasets. Specifically people who trained on the first day might have a dataset advantage with respect to people who trained on the last available day. Given the magnitude of the dataset, and the tendency of such models to be constantly re-trained over the course of the challenge, we do not expect this to impact things too much as participants are always required to use the latest version available.

3.4 Comparison to data from Twitter’s Recsys2020 challenge

This challenge has much in common with last year’s: Recsys Challenge 2021. For those familiar with Recsys Challenge 2020 [2], briefly summarize the major similarities and differences from this year’s version in Table 3.4.

| | 2020 Challenge | 2021 Challenge |
|------------|--|---|
| size | ~ 200 million records | ~ 1 billion records |
| Process | Scrubbing, pseudonegatives, pre-featured text tokens | Scrubbing, pseudonegatives, pre-featured text tokens |
| Sampling | Fully uniform sampling | Half uniform sampling and half of records sampled from a denser subgraph |
| Submission | Submit TSV file with predictions (no latency constraint) | Submit model that must run within 24h on fixed hardware |
| Metrics | Accuracy metrics determine winner (PR-AUC and RCE) | Accuracy metrics coupled with a popularity based fairness metric determine winner |

Table 2: 2020 Challenge versus 2021 Challenge

¹<https://developer.twitter.com/en/developer-terms/agreement>

3.5 Differences with Respect to the Twitter Case

This dataset captures the main characteristics of those used at Twitter to build recommender systems. However, because this data was compiled for external users, some simplifications were necessary.

To respect users’ privacy, only (a strict subset of) features that are already public (e.g. number of likes for a given Tweet) are made available in the dataset. Machine learning practitioners within Twitter have access to a larger number of features, both public and private.

In the interest of privacy, we obfuscated all the field that would make the re-identification trivial (e.g. the Tweet id) before releasing the dataset. We are aware that this approach is ineffective against linkage attacks (e.g. [7] and [12]) and the data can be easily reconstructed. However there are reduced privacy concerns since all the data that we started with is already public. Contrast this dataset with a dataset where a mixture of both private and public data was released (e.g. Netflix challenge) and public features were used to infer private ones.

This dataset is also much smaller scale than data availability to machine learning practitioners within Twitter. Even with its almost 1B rows and more than 200 GB, it only represent a small fraction of the all the content that is produced on Twitter at any given moment. That said, to the best of our knowledge — at the time of writing — this dataset is the biggest publicly available social media-based dataset for recommender systems.

When a production model ranks the Tweets for a user u , it has access to the full set of candidates (e.g. all Tweets that were produced by u ’s followees since last time they requested a fresh timeline). In the dataset we cannot guarantee that all the candidates are present — both at training and validation/test phases.

Finally, for the purpose of this challenge, we are only including content from users that are explicitly followed by the reader. In reality the situation is more complicated, and other kind of content might be presented to the user. For example, as mentioned above users may be presented with Tweets that belong to Topics they follow. A user’s timeline may also include Tweets that someone they follow liked or responded to, even if they do not follow that person themselves. Additionally, paid advertisements may also appear among the set of Tweets displayed to a user.

4 The RecSys 2021 Challenge

In the past, ML challenges focused on optimizing for a single metric (e.g. accuracy) without considering other factors. For the first time in the history of the RecSys challenge, we have included fairness considerations into the evaluation metrics. To the best of our knowledge, this is the largest challenge to date that has incorporated a fairness component.

Fairness is a societal concept, not an optimization one [10] and it can come in many different shape or forms. When defining what fairness means for the purpose of this challenge, we considered three criteria:

- We wanted to ease participants into the fairness space, without requiring too much prior knowledge or special infrastructure.
- At the same time, we wanted the problem to be meaningful and have a real significance for Twitter. In short, we did not want to focus on a “toy problem.”
- Finally we wanted to rely on the available features (see Section 3.1). This prevented us from considering any metrics that depend on demographic data, as those features are not included in the dataset.

A popularity-based (measured as the number of followers for an author) metric, has all the above characteristics. In this scenario, the quality of the recommendations should be independent from the popularity of the authors. Said in another way, users should not be getting worse recommendations for being less popular on the platform.

Concretely, we divide users into 5 quantiles (based on the number of followers of the authors in the test set). In the parlance of fair ML, the “sensitive attribute” for the purposes of evaluation is the binned

user popularity variable. The number of rows in the dataset is not be equal for each cohort since more popular authors have a larger audience which typically translates into more opportunity for incoming engagement.

We also want to acknowledge that having more followers might be itself an effect of a feedback loop from the recommender system: more popular authors might be shared more thus gaining more even popularity. We do not want to suggest that the producer popularity constraint is the only, or even the most important, aspect to consider for serving fair recommendations. At the same time we do feel that it allows us to make an important step forward with the recommender system community about the recommendations not being only about accuracy.

4.1 Metrics

For last year’s challenge, we used Relative Cross Entropy (RCE) and Area under the Precision Recall Curve (PR-AUC) for each engagement, ranked these and then summed the positions to get the final score. For more details on how those metrics were calculated, refer to [2].

However to calculate PR-AUC we used the Python `scipy` library [13]. The method implemented there has a drawback that constant predictions you will get a PR-AUC of 0.5. This was exploited by a lot of teams who just submitted constant predictions - meaning they would have very bad RCE but a PR-AUC of 0.5.

To prevent the same problem this year, we used average precision and RCE.

4.1.1 Relative Cross Entropy

Relative Cross Entropy (RCE) corresponds to the improvement of a prediction relative to the straw man, or the naive prediction, measured in cross entropy (CE). The naive prediction corresponds to the case that does not take into account the user and Tweet features, e.g., it always predicts the average (observed) CTR of the training set. Suppose the average CE of the naive prediction is CE_{naive} and average CE of the prediction to be evaluated is CE_{pred} , then RCE is defined as $(CE_{naive} - CE_{pred}) \times 100/CE_{naive}$. Note that the lower the CE the better the quality of the predictions, so the higher the RCE. 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.

4.1.2 Average Precision

We used Average Precision from the `scikit-learn` package [8], that “summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight.” While this metric is basically the same as PR-AUC, it helps with the aforementioned problem with constant predictions.

4.1.3 Final score

As discussed, we wanted the performance of the recommendations to be independent of the popularity of the user. After dividing the authors in five quantiles (based on the number of followers of the authors on the test set), average precision and RCE are calculated for each group. The two scores are then averaged across groups (so that entries that have great performance on popular users but perform poorly on smaller accounts are penalized) and ranked. The final score is the sum of the position according to each score. So for example, if someone ranked first in the RCE ladder and third in the AP one, their final score would be 4.

4.2 Baseline

In this section, we will describe a simple baseline model architecture that we use for this competition. We used a similar baseline as last year [2]. Specifically, it mainly consists of the feature processing and prediction components.

4.2.1 Feature processing

We apply different feature processing methods based on different feature types. For Numeric Features (e.g. follower count), we apply z -score normalization first to shift the feature into a reasonable range, and then transform each normalized feature with a tiny neural network, which consists of one hidden layer of size 2 and \tanh activation. The missing features are populated with 0.

Categorical features (e.g., Tweet type, language) and binary features are represented using one-hot encoding. Please note that we reserve a bucket for any missing categorical and binary feature values. ID features are hashed into a given number of buckets due to its extremely high cardinality. Tweet text feature are tokenized and embedded using a pre-trained BERT model [5].

4.2.2 Model

The one-hot encoded representations of the aforementioned categorical, binary and ID features are concatenated and converted to dense representation of size 16, which are then concatenated with the normalized numeric features. We omit the text encoding module in the submission for simplicity and computational efficiency. The obtained processed feature vector is then fed into a 3 layer multi-layer perceptron, which consist of 3 dense layers of size 128, 64, and 32, respectively. The activation functions are leaky ReLU. The final prediction of the model is of size 4, corresponding to 4 types of engagement (Retweet, Reply, Like and Retweet with comment). In the final prediction layer, the activation function is chosen as sigmoid rather than softmax since different types of engagements are not mutually exclusive to each other. The model is trained with Adam optimizer [6] using Huber loss for 1 million steps, and the learning rate is 0.001.

4.3 Latency constrains

In the RecSys Challenge 2020 [2], after the training and validation phase, participants were asked to upload their prediction on the test set. This gave total freedom to participants to train and test their model on their own hardware and did not impose any restrictions. In practice, Twitter’s models need to be able to score content on the order of milliseconds. While we did not want to impose the same production constraints, we also wanted to nudge participants away from slower and more complex models (e.g. ensemble ones) even if this comes at the cost of accuracy.

While during the training stage, participants were free to train on hardware of their choosing, for the testing phase, we asked people to upload their code, which would be run on a dedicated Docker instance. Furthermore, we imposed a 24h timeout limit for the entirety of the dataset. The dedicated instance had 1 CPU with 64Gb of RAM.

5 Conclusions

In this paper, we introduce Twitter’s Recsys 2021 Challenge. This challenge introduces some innovations: the inclusion of a fairness consideration in the evaluation criteria, a novel approach for indirectly incentivizing low-latency solutions, and a requirement that participants respect the changing desires of users– that tweets that a user has deleted are no longer included in the dataset. In building this challenge, we balanced several different considerations: fidelity to the real problems at Twitter, user privacy and agency, and providing data that is of a format and scale that is useful for Recsys practitioners who hope to participate in a Challenge that resembles the type of model-building that takes place at Twitter. We hope this challenge will help shape the direction of similar challenges in the future.

6 RecSys 2021 Challenge Summary

For RecSys 2021 Challenge, we have two winning groups from Academia and Industry and each group consists of three winning positions. Based on the resources available in Academia and Industry, we have identified three winning solutions from the general leaderboard and the three best solutions whose teams are composed only of academic people. In the following, we provide a brief overview of all the

winning solutions. The industry based solutions for twitter user engagement predictions are mainly either ensemble of stacked models or hybrid of various deep neural networks and XGBoost models. The solutions in the order of winning position are:

- 1st place: *GPU Accelerated Boosted Trees and Deep Neural Networks for Better Recommender Systems*, where authors used an end-to-end GPU-accelerated ensemble of stacked models, using in total 5 XGBoost models and 3 neural networks for engagement predictions.
- 2nd place: *Synerise at RecSys 2021: Twitter user engagement prediction with a fast neural model*, with a simple feed forward neural network that predicts the probability of different engagements types for user and target Tweets by obtaining meaningful Tweet text representation using Distil-BERT model and EMD.
- 3rd place: *User Engagement Modeling with Deep Learning and Language Models* where a hybrid Java and Python pipeline was used to extract features from Tweets content which are used to train 4 XGBoost models, one for each engagement type. The authors trained a neural classifier with multi-layer perceptions to predict on users engagement probabilities.

We also have three winning position for teams composed only of academic people. The academia solutions are mainly consists of using LightGBM and XGBoost models for Twitter user engagement predictions. The acadmic winners are:

- 1st place: *Lightweight and Scalable Model for Tweet Engagements Predictions in a Resource-constrained Environment* where the approach used was an optimized LightGBM model which leverages a wide variety of meaningful features from the text of each Tweet and adopted two types of models for engagement prediction: Neural Network (NN) and Gradient Boosting for Decision Tree (GBDT) models.
- 2nd place: *Addressing the cold-start problem with a two-branch architecture for fair tweet recommendation* proposed a two-branch architecture that separates Twitter authors according to their total number of interactions in the dataset. Their solution consists of prediction the user engagement using NN method by identifying similar users and LightGBM models
- 3rd place: *Team JKU-AIWarriors in the ACM Recommender Systems Challenge 2021: Lightweight XGBoost Recommendation Approach Leveraging User Features* used a model that relies on features that can be computed from user engagement counts and exploit a simple XGB classifier, trained on a subset of the training data for twitter user engagement predictions.

Acknowledgements

The authors would like to thank (in alphabetical order) Robin Burke, Rumman Chowdhury, Michael Ekstrand, Vladislav Grozin and Martha Larson for the fruitful conversations, especially around the topic of fairness. Furthermore the work of Chandler Bair was instrumental in the success of this Challenge.

References

- [1] Abolfazl Asudeh, HV Jagadish, Julia Stoyanovich, and Gautam Das. Designing fair ranking schemes. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1259–1276, 2019.
- [2] Luca Belli, Sofia Ira Ktena, Alykhan Tejani, Alexandre Lung-Yut-Fon, Frank Portman, Xiao Zhu, Yuanpu Xie, Akshay Gupta, Michael Bronstein, Amra Delić, Gabriele Sottocornola, Walter Anelli, Nazareno Andrade, Jessie Smith, and Wenzhe Shi. Privacy-aware recommender systems challenge on twitter’s home timeline, 2020.
- [3] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international acm sigir conference on research & development in information retrieval*, pages 405–414, 2018.
- [4] S. Corbett-Davies and Sharad Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *ArXiv*, abs/1808.00023, 2018.

- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
- [7] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset, 2006.
- [8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [9] Piotr Sapiezynski, Wesley Zeng, Ronald E Robertson, Alan Mislove, and Christo Wilson. Quantifying the impact of user attention on fair group representation in ranked lists. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 553–562, 2019.
- [10] Andrew D. Selbst, Danah Boyd, Sorelle A. Friedler, Suresh Venkatasubramanian, and Janet Vertesi. Fairness and abstraction in sociotechnical systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* '19*, page 59–68, New York, NY, USA, 2019. Association for Computing Machinery.
- [11] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228, 2018.
- [12] Latanya Sweeney. Guaranteeing anonymity when sharing medical data, the datafly system. In *Proceedings: a conference of the American Medical Informatics Association. AMIA Fall Symposium*, page 51–55, Nashville, TN, USA, 1997. Hanley & Belfus, Inc.
- [13] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.