Towards inferring ratings from user behavior in BitTorrent communities

Róbert Ormándi, István Hegedűs, Kornél Csernai
Research Group on Artificial Intelligence
University of Szeged
Szeged, Hungary
{*ormandi,ihegedus,csko*}*@inf.u-szeged.hu*

Márk Jelasity
Research Group on Artificial Intelligence
University of Szeged and Hungarian Academy of Sciences
Szeged, Hungary
jelasity@inf.u-szeged.hu

***Abstract*—Peer-to-peer file-sharing has been increasingly popular in the last decade. In most cases file-sharing communities provide only minimal functionality, such as search and download. Extra features such as recommendation are difficult to implement because users are typically unwilling to provide sufficient rating information for the items they download. For this reason, it would be desirable to utilize user behavior to infer implicit ratings. For example, if a user deletes a file after downloading it, we could infer that the rating is low, or if the user is seeding the file for a long time, the rating is high. In this paper we demonstrate that it is indeed possible to infer implicit ratings from user behavior. We work with a large trace of Filelist.org, a BitTorrent-based private community, and demonstrate that we can identify a binary like/dislike distinction over the set of files users are downloading, using dynamic features of swarm membership. The resulting database containing the inferred ratings will be published online publicly and it can be used as a benchmark for P2P recommender systems.**

***Keywords*-peer-to-peer, database, recommendation**

I. Introduction

Without doubt the popularity of P2P file-sharing has been steadily growing in the past decade. Apart from the basic functionalities of filename search and downloading, there is demand for extra functionality such as high quality personalized recommendations.

Most approaches for recommender systems take a rating matrix as input. The ratings are often taken from a small numeric range, but in any case, one needs at least two different values, which stand for "dislike" and "like". This matrix usually contains explicit ratings of items by the users. Since users have to make an effort to rate items, ratings are often scarce and of dubious quality. For this reason, it has long been observed that inferring ratings from user behavior is an important way of enhancing the rating matrix and thus the quality of recommendations. For example, [1] provides an overview of many such methods.

Inferring ratings is a must, when no explicit information is available. This is often case in P2P systems. For example, [2] discusses the case of P2P TV, where ratings are inferred from channel zapping behavior.

We will focus on BitTorrent file-sharing communities, where users do not provide any ratings explicitly. We will work with a large Filelist.org trace collected by the Technical University of Delft. From this trace we shall use swarm membership information as a function of time, and based on dynamic changes in swarm membership, we will infer a binary classification of file-rating ("like"/"dislike").

Our contribution is threefold. First, we will define a method for inferring ratings from dynamic swarm membership data. Second, we will argue that the inferred ratings "make sense", through demonstrating that both the "like" and "dislike" classes of ratings can be captured by several statistical learning methods. This is remarkable because the original swarm membership data does not contain any information that can be interpreted as "dislike" (if a user is a member of a swarm, then our best guess is that the user likes the corresponding file, but lack of membership does not imply that the user does not like the file). Furthermore even the "like" ratings can be predicted better from our inferred data, since swarm membership at a certain time does not always imply the user likes the file. Third, we will make our inferred database publicly available so that it could serve as a baseline for P2P recommender systems.

The outline of the paper is as follows. Section II describes the Filelist.org trace we used for our analysis. Section III discusses the methodology we used to create the inferred dataset from the raw trace. In Section IV we present the results of several statistical learning methods on the original raw dataset as well as the inferred dataset. Section V concludes the paper.

II. Filelist.org Trace

In this section we will provide a brief overview of the dataset that is the basis of our benchmark. The data source we present originates from a BitTorrent-based [3] P2P file-sharing community called FileList.org [2] [4].

A. BitTorrent basics

In a BitTorrent P2P network, each peer (user) downloads and uploads data simultaneously. The *torrent* file format describes one or more files that are to be shared. The files are then split up into *chunks* or *pieces*, which are

[2]This site is now defunct. Older, archived pages are available at http://archive.org.

transferred in *blocks*. The peers can acquire these pieces in any order. A *swarm* is the set of peers participating in downloading a common torrent. Each swarm is identified by its *Info Hash*, that is the SHA-1 hash of the data used to uniquely describe a torrent (file list, file sizes, etc). With this, each node can assure that the right content is being requested while interacting with other peers, as well as verify the integrity of the downloaded data. The peers that have completed downloading all the pieces of a torrent are called *seed*s, whereas the ones still trying to get some of them are called *leech*es. The *sharing ratio* is defined as the *uploaded/downloaded* value for each session, as well as for the lifetime of the membership, if a (private) community is involved.

A *tracker* is typically a central database-driven website that coordinates the peers and keeps track of their uploads, downloads, sharing ratios, client versions and so on. The trace we work with was collected via downloading this tracker database regularly.

Peers start a session by registering at the tracker, and requesting a random subset of online peers' ⟨IP Address, Port⟩ pairs from the tracker. After registering, they can initiate connections to other peers, and also receive connections. Note that they may be firewalled (or NAT-ed) peers which cannot receive incoming connections, thus limiting their ability to communicate.

B. The FILELIST.ORG community

FILELIST.ORG is a private community [5], [6], meaning that members have to be invited by a senior member in order to be able to join the community website. Also, they have to comply with specific rules regarding their overall sharing ratio. Users that have a sharing ratio below given thresholds may have to wait hours before being able to start downloading a newly added torrent, or (in worse cases), be excluded from the community. This type of community is general and quite frequent when the main interaction activity is file-sharing. The tracker website announces the private torrent files which are categorized, so users can check back regularly for new content.

C. The original FILELIST.ORG trace

We will base our benchmark dataset on the trace files gathered by Jelle Roozenburg at the Technical University of Delft [7] between 9th December, 2005 and 12th March, 2006. Over the course of 93 days, $91,745$ peers (users) were observed in $3,068$ swarms (sets of users participating in downloading the same file). For the files that were followed, $5,477$ TB of data was reported to have been exchanged ($2,979$ TB up, $2,498$ TB down) among all the users combined. Normally these two values should be equal, but there are some factors that can change this property in our system. For example, some clients might misreport, or might be deleted from the website over time (for not obeying the rules). Also, content can "leak", i.e., can be mistakenly or deliberately distributed to peers that are not part of the private tracker and thus do not report to it (Peer Exchange, Distributed Hash Table). As regards the churn rate, $9,574,290$ joins and leaves were observed.

For the measurements, the tracker website is periodically crawled to obtain a partial state of the P2P network. These measurements can be more accurate than active measurement since we have a central knowledge of each peer.

After crawling the websites and archiving the HTML pages, the parsing phase yields two datasets, namely swarm churn and peer behavior. Here we shall only examine the peer behavior files. The peer behavior dataset is made up of directories representing swarms and files representing each peer's participation. Each line of these files contains an event the following properties:

- the UNIX timestamp of the event
- a bit indicating whether the peer is online
- a bit indicating whether the peer can be connected by other peers
- the uploaded and downloaded amount of data in KB
- the average upload and download speed in KB/s
- the sharing ratio, i.e. uploaded/downloaded
- the completion of the file in %
- the connection time in minutes
- the client version string

The original trace is rich in data as it consists of $691,319,475$ events.

When analyzing this trace, we came across patterns. We split the online events into two groups (leech and seed) based on the completion field ($< 100\%$ and $=100\%$, respectively), then we observed that the peers act according to one of the following patterns:

- (72%) Start as leech, end up as a seed. We conclude that the peer has successfully downloaded the torrent.
- (15%) Remain as a leech all the time. We conclude that the peer tried, but could not successfully download the torrent.
- (10%) Remain as a seed all the time. We conclude that the peer is either an injector (the one that introduces the torrent to the community and therefore has the content initially) or has acquired the torrent from another source and would like to contribute.
- (3%) Various patterns of repeated transitions from being a leech and a seed. We speculate that this (very small) portion of users have (intentionally or not) deleted the torrent and/or joined the swarm from a different computer. Also, *super-seeding* [8], [9] (or *initial seeding*) might have played a role in creating these patterns.

Table I
ONLINE SESSION INTERPOLATION RULES

Online sessions		
front	end	fill
0	0	0
0	1	1
1	0	0
1	1	1

Table II
OFFLINE SESSION INTERPOLATION RULES

Offline sessions		
front	end	fill
0	0	0
0	1	0
1	0	0
1	1	1

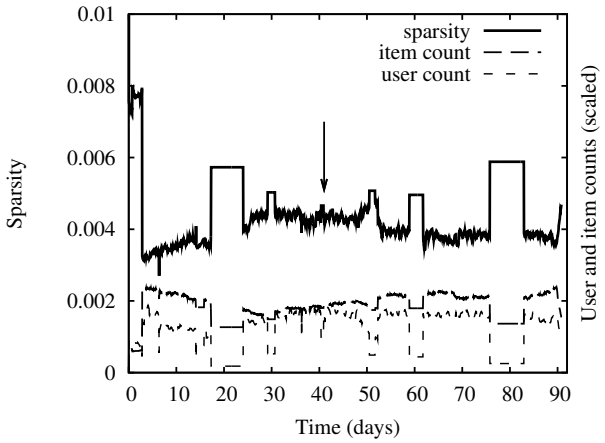

Figure 1. Sparsity changes during trace interval.

III. INFERRING RATINGS FROM THE TRACE

A. Preprocessing Raw Data

The original raw dataset detailed above was first converted into a more convenient format, removing any unnecessary information, then the remaining table consisted of the following fields: user ID, item ID, timestamp, online/offline, completion.

These discrete points define a sequence of online and offline sessions for each user in each swarm. Using these sessions, we extrapolated file-ownership at any point in time as follows. We first filled the inner parts of each online session using the interpolation rules given in Table I. Afterwards, we filled the offline sessions of each user-file pair by applying the rules given in Table II. From these intervals we were able to generate a user-item database for any given point in time.

A user-item database that corresponds to a point in time contains two kinds of entries: "has file" and "does not have file". That is, if a user has already completed downloading the given file, and has not removed the file, then he has the file. Note that if a user is downloading a file, but has not yet completed the downloads, then he does not have the file and is just like a user who has never attempted to download it.

Note that we could have retained information on swarm membership as well; that is, introduce the third label "downloading file". Though possible, we eventually decided not to do it so as to simplify the problem and avoid the semantic problems associated with this label. Also note that it is possible that a user has the file, but is not seeding it, so we still say "does not have file". However, it is impossible to decide whether a user removed the file because he does not like it or because he does not want to seed it. Overall, one has to be careful with interpreting these labels, but, as our evaluation will show, this labeling does result in useful results despite these problematic issues.

B. Sparsity of the Dataset

Sparsity is the ratio of the known and unknown ratings. In our case, we took the label "has file" to be the known rating; otherwise we considered the rating as unknown.

Sparsity is one of the most important characteristics of recommender benchmarks [10]. If the data is very sparse (as in the case of the BookCrossing benchmark [11]) then numerous problems arise such as difficulties with measuring the similarity between users or building statistical models. The sparsity of a recommender database is an important factor in the difficulty of making recommendations based on the given dataset.

For this reason, we examined the dynamics of sparsity as a function of time, as shown in Figure 1. We can observe that sparsity is fairly stable, and has a small value; that is, the data is sparse. Figure 1 shows the number of online users and the number of active files as well, as a function of time. The plateaus of the curves correspond to missing data.

C. Inferring Ratings

In order to infer the preference of users, we first fixed a point in time, and took the corresponding user-item matrix. Originally we took three different points in time, but our preliminary results indicated that there is no visible difference in performance over time, so we kept the time point indicated by an arrow in Figure 1.

Our baseline (or naive) dataset is given directly by the user-item matrix we selected earlier. From the point of view of ratings, we took "has file" to be a positive rating (indicated by the numeric value 1), otherwise we indicated *a lack of rating*. In other words, in the baseline user-item matrix we did not have any entries indicating a negative rating, since we have no basis to infer a negative rating from this data.

To infer negative ratings, and to make the positive ratings more precise, we used information that varied in time: we

Table III
RATING CONVERSION RULES

Dataset labeling			
before	actual	after	inference
0	0	0	unspecified
0	0	1	unspecified
0	1	0	**0 (dislike)**
0	1	1	**1 (like)**
1	0	0	**0 (dislike)**
1	0	1	unspecified
1	1	0	unspecified
1	1	1	**1 (like)**

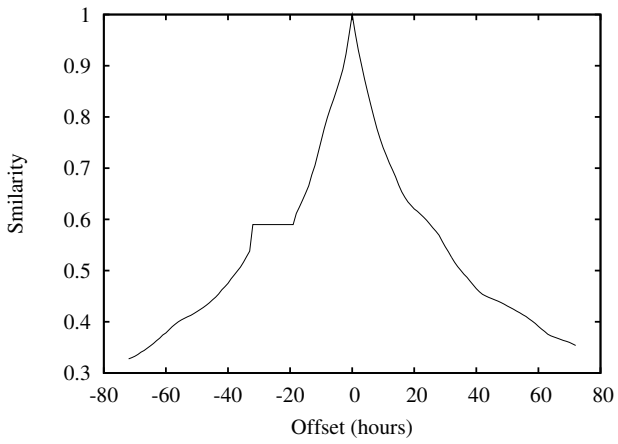

Figure 2. Similarity as a function of time.

looked at file ownership before and after the timestamp of the baseline dataset. The amount of shift in time was the same in both directions. This way, for each user-item pair we got a triplet, which we converted to ratings, as indicated in Table III. These rules are entirely heuristic and are based on common sense. This way we can create negative ratings (with a numeric value of 0).

It is interesting to observe the similarity of the user-item matrices (ratio of entries with an identical label) as a function of time shift. Figure 2 shows the difference between our baseline dataset and the user-item matrix with the given time shift. The figure indicated that there is a significant change in the user-item matrix, and thus the the dynamics might indeed convey useful extra information.

IV. EVALUATION

Since we do not have any ground truth available for the actual like/dislike ratings of users, we need to apply an indirect approach to test the labels we assigned to user-item pairs.

The method we propose is based on the learnability of the labels. That is, if statistical learning algorithms are able to predict the labels based on a training set (which is the subset of the data we generated), then we can at least say that the labels do correspond to some regular property of the data. This regularity might come from an unintended source as well. To deal with this possibility, during the evaluation we explicitly attempted to single out some trivial sources of regularity such as one label being more frequent than the other, etc. In the experiment we could not find any trivial or unintended reasons for the learnability of the labels.

A. Performance of Learning Algorithms

For learning we decided to use the following algorithms available from the WEKA Java library [12]:

- SMO: a support vector machine implementation in WEKA, with default parameters
- J48: a decision tree learning implementation in WEKA, with default parameters
- Naive Bayes: WEKA implementation, default parameters
- Logistic: WEKA implementation for logistic regression, default parameters

In order to generate features from the user-item ratings, we adopted the proposals described in [13]. These features are specifically designed for sparse datasets like ours.

For the baseline user-item matrix, the training set was generated by first selecting 90% of the user-item entries with the "has file" (1) label, and then we selected the same number of entries with the "does not have file" label. The test set was composed of the remaining 10% of the "has file" entries, and the same number of "does not have file" entries, disjunct from the training set.

A similar method was used for the time-shift based datasets, for all the given time-shift values. The only difference is that there we had three labels: 1 (like), 0 (dislike) and null (don't know). Accordingly, to define a two-class learning problem, we created two pairs of training and test sets: one for the "like" class, and one for the "dislike" class.

The results are shown in Figure 3. The F-measure is the usual indicator for evaluating learning algorithms. It is defined as the harmonic mean of *precision* and *recall*. Precision is defined as the number of true positive predictions divided by all the positive predictions (how many of the positive predictions are indeed positive), while recall is given by the true positive predictions divided by the sum of the true positive and false negative predictions (how many of the positives do we catch).

It is evident that all the classes are learnable (randomly assigning labels to data results in an F-measure of 0.5). Besides, it is also evident that both classes achieve a higher F-measure score than the baseline does. That is, clearly, applying the time-shift preprocessing, we were able to achieve a better quality dataset.

B. The published database

The published database is available for research purposes and can be downloaded from [14]. Currently, an inferred dataset from the $[-60; +60]$ hour interval is available in

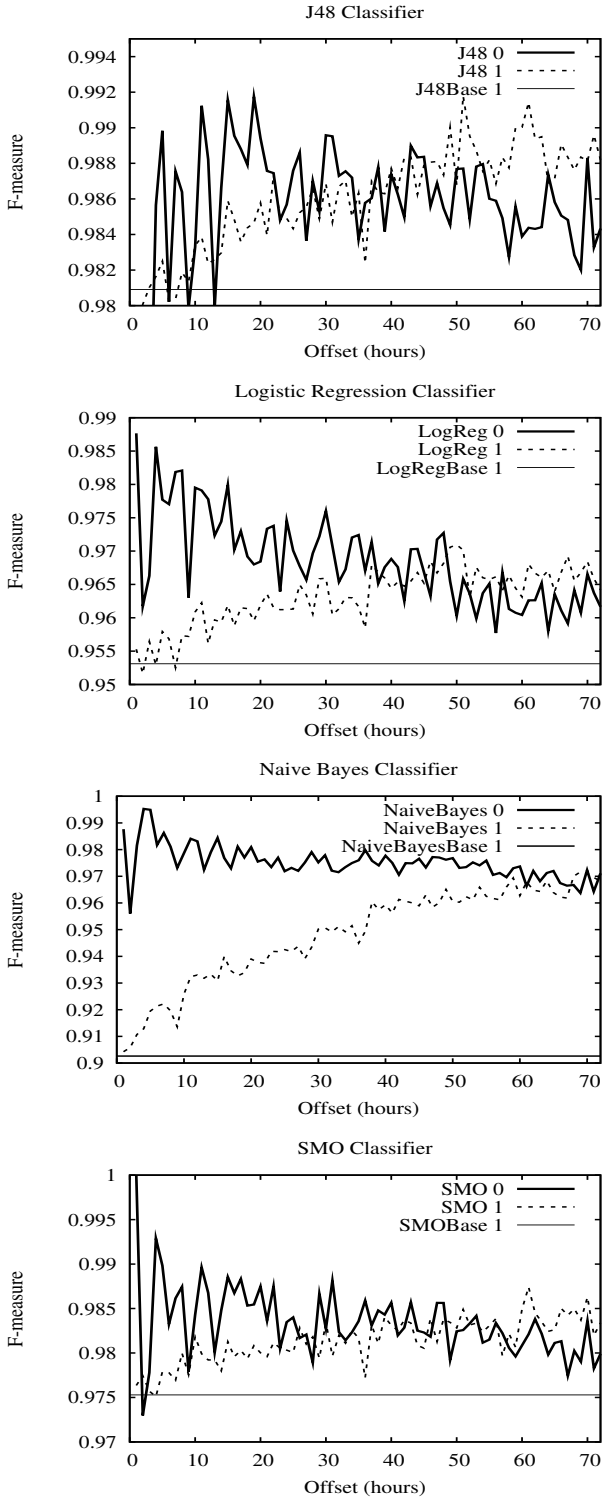

Figure 3. F-measure scores of the different time-shift datasets for positive (1) and negative (0) ratings (a larger value means a better performance score).

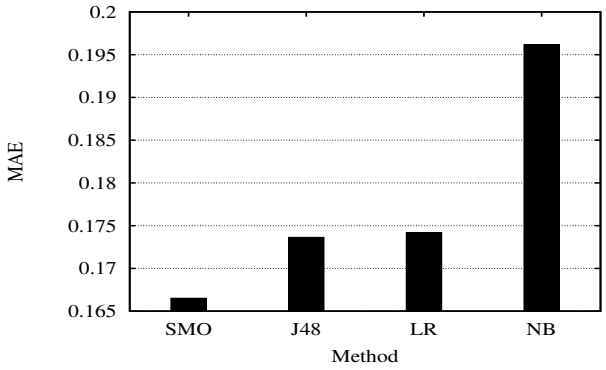

Figure 4. The mean absolute error (MAE) of the SMO, J48, LogReg and Naive Bayes methods (smaller values are better).

the following format: three numerical columns separated by a tabulator which are the user ID, the item ID, and the rate (which is either 1 or 0, meaning the item was liked or disliked, respectively). We conducted experiments with the above-mentioned methods on this database, whose results are shown in Figure 4. In this evaluation case we split the database into train and test sets in the ratio 9:1. We chose the samples of sets at random from a uniform distribution. We used the mean absolute error (MAE) measure to demonstrate the performance of the baseline methods on the database.

V. Conclusions

In this paper we tackled the problem of inferring ratings from user behavior in a BitTorrent community. We proposed a method based on the dynamic features of file-ownership. To simplify it slightly: we supposed that a user likes a file that he keeps seeding, and we supposed that a user does not like a file that he downloads and then deletes, or that he does not seed.

While heuristics of this nature are far from precise, we were able to demonstrate that the negative rating of a file we predict this way is far from random; that is, a wide range of machine learning algorithms are able to learn it to achieve high precision (note that in the raw data set there are no direct negative ratings at all). Using our method, the positive ratings are more learnable as well. While this is only implicit evidence (since we have no ground truth available), we believe that our approach significantly improves the rating data reflecting real user preferences.

We have made our inferred database publicly available as well [14].

Acknowledgments

We would like to thank Tamás Vinkó and Johan Pouwelse for their advice and for making the Filelist.org trace available to us. M. Jelasity was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences. This work was partially supported by the Future and Emerging Technologies

programme FP7-COSI-ICT of the European Commission through project QLectives (grant no.: 231200).

REFERENCES

[1] D. Kelly and J. Teevan, "Implicit feedback for inferring user preference: a bibliography," *SIGIR Forum*, vol. 37, no. 2, pp. 18–28, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?id=959260

[2] J. Wang, J. Pouwelse, J. Fokker, A. de Vries, and M. Reinders, "Personalization on a peer-to-peer television system," *Multimedia Tools and Applications*, vol. 36, no. 1, pp. 89–113, 2008. [Online]. Available: http://dx.doi.org/10.1007/s11042-006-0075-6

[3] B. Cohen, "Incentives build robustness in bittorrent," in *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, 2003.

[4] "Filelist," http://www.filelist.org, 2005.

[5] C. Zhang, P. Dhungel, D. Wu, and K. W. Ross, "Unraveling the bit-torrent ecosystem," in *Technical Report, Polytechnic Institute of NYU*, May 2009.

[6] C. Zhang, P. Dhungel, D. Wu, Z. Liu, and K. W. Ross, "Bittorrent darknets," in *Proceedings of IEEE Conference on Computer Communications (IEEE INFOCOM '10)*, March 2010.

[7] J. Roozenburg, "Secure decentralized swarm discovery in tribler," Master's thesis, Parallel and Distributed Systems Group, Delft University of Technology, 2006. [Online]. Available: http://www.pds.ewi.tudelft.nl/~epema/MSc-theses/MSc-thesis-Roozenburg.pdf

[8] Z. Chen, C. Lin, Y. Chen, V. Nivargi, and P. Cao, "An analytical and experimental study of super-seeding in bittorrent-like p2p networks," *IEICE Transactions*, vol. 91-B, no. 12, pp. 3842–3850, 2008.

[9] Z. Chen, Y. Chen, C. Lin, V. Nivargi, and P. Cao, "Experimental analysis of super-seeding in bittorrent," in *ICC*. IEEE, 2008, pp. 65–69.

[10] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

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

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[13] Y.-J. Park and A. Tuzhilin, "The long tail of recommender systems and how to leverage it," in *Proceedings of the 2008 ACM conference on Recommender systems (RecSys '08)*. New York, NY, USA: ACM, 2008, pp. 11–18.

[14] "The filelist.org-based inferred recommendation dataset," http://www.inf.u-szeged.hu/rgai/recommendation/.