IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 24, 2015, accepted November 2, 2015, date of publication December 29, 2015,
date of current version March 1, 2016.

Digital Object Identifier 10.1109/ACCESS.2015.2513000

Improving Scalability of Personalized Recommendation Systems for Enterprise Knowledge Workers

CHETAN VERMA[1], MICHAEL HART[2], SANDEEP BHATKAR[2], ALEATHA PARKER-WOOD[2], AND SUJIT DEY[1]

[1]Mobile Systems Design Laboratory, Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA 92092, USA
[2]Symantec Research Labs, Mountain View, CA 94043, USA

Corresponding author: C. Verma (cverma@ucsd.edu)

This work was supported by Symantec Corporation.

ABSTRACT Enterprise knowledge workers have been overwhelmed by the growing rate of incoming data in recent years. In this paper, we present a recommendation system with the goal of helping knowledge workers in discovering useful new content. In particular, our system builds personalized user models based on file activities on enterprise network file servers. Our models use novel features that are derived from file metadata and user collaboration. Through extensive evaluation on real-world enterprise data, we demonstrate the effectiveness of our system with high precision and recall values. Unfortunately, our experiments reveal that per-user models are unable to handle heavy workloads. To address this limitation, we propose a novel optimization technique, active feature-based model selection, that predicts the user models that should be applied on each test file. Such a technique can reduce the classification time per file by as much as 23 times without sacrificing accuracy. We also show how this technique can be extended to improve the scalability exponentially at marginal cost of prediction accuracy, e.g., we can gain 169 times faster performance on an average across all shares by sacrificing 4% of F-score.

INDEX TERMS Information retrieval, machine learning, enterprise, file systems.

I. INTRODUCTION

The amount of data created, replicated and consumed every year has been growing exponentially over the years [1]. Between 2010 and 2020, the digital data is expected to grow by 50 times [2]. Following a similar trend, enterprise data was projected to increase by 76% within the last one and half years [2]. The tremendous growth of data has become one of the biggest challenges for enterprises. It places unreasonable burden on enterprise users or knowledge workers. Studies [1] have shown that nearly 65% of knowledge workers report feeling overwhelmed by the incoming data. The situation is further exacerbated by the fact that the enterprise data resides across heterogeneous devices and environments, including network file servers, source control repositories, emails and file servers, cloud applications [3], [4], and enterprise social networks [5]. This makes finding relevant content a very challenging task for knowledge workers.

In this paper, we present a system that assists knowledge workers to *discover* relevant new content by providing personalized recommendations. In order to serve these recommendations, our system monitors user activity over a training period and trains per-user access models. Although our approach currently focuses on files on networked file servers (shares), it could potentially be applied to other types of data as well. For training user models, we extract interesting metadata features based on file path and hierarchical directory structure. In our experiments, we observe that many enterprise users show a high degree of collaboration, which is inferred based on their common file accesses. To leverage user collaboration, we propose a collaborative filtering aware modeling approach that provides significant improvement over metadata-based models. Our evaluation uses activity logs from eight different shares of an enterprise customer. We measure the effectiveness of our system in providing predictions for files that are new with respect to the training data. Our experiments show that the metadata-based models can achieve an average recall of nearly 50% at 75% precision across all shares and users. Our collaborative filtering-based approach remarkably improves the same recall of nearly 50% at 75% precision to over 70%.

Although effective, per user models are not at all efficient in terms of the runtime performance of testing. This is because every file that is created or modified in a share needs to be tested against the model of each user in the share. This could be computationally very expensive especially for heavy workloads with a large number of users. For instance, for handling a heavy workload (i.e., activity involving high file edit rate) in our experiments, our system will need 48 64-core machines to test all edited files. The high computational and financial cost poses a severe constraint that could adversely affect the adoption of our system. To address this issue, we propose a novel technique, called Active Features-based Model Selection (AFMS), that automatically selects only those user models that are highly likely to generate positive predictions, while ignoring the models that will definitely or most likely generate negative predictions. Thus using this technique, a test file is tested against only the selected models, thereby speeding up the overall testing time. Our experiments show that AFMS drastically reduces testing time without any loss in the prediction accuracy. Furthermore, we show that AFMS can be used adaptively to trade-off marginal prediction accuracy for significant performance gains during the periods of heavy workload.

Our previous version [6] of this work presented the basic technique for modeling file accesses. This paper presents a new technique, AFMS, for improving solubility, along with new experiments and insights. The main contributions of our work are the following:

1) Propose a system to analyze user file accesses in order to train personalized models, and recommend relevant content with a high degree of accuracy
2) Propose a novel model selection technique that can speed up the testing time by more than 6 times on average, and by over 23 times in the best case without any loss in the prediction accuracy
3) Extend the model selection technique to speed up the testing time significantly at the cost of marginal loss in the prediction accuracy
4) Demonstrate practicality of our system using a systematic study and evaluation based on real world enterprise data

II. RELATED WORK

Several previous approaches have addressed the topic of file access prediction, but with the goal of improving the I/O performance of storage systems [7]–[14]. They aim at reducing the widening gap between CPU and disk storage performance by prefetching files to the cache memory. There are a couple of key differences between our approach and the previous ones. Such approaches are focused on modeling accesses patterns that are generated by automated activities, whereas our focus is on activities that are manually initiated by physical users. This allows us to consider certain features that could be prohibitively expensive for caching applications. None of the previous approaches make predictions for completely new content. On the other hand, by leveraging innovative features derived from metadata and user collaborations, our approach can make fairly good predictions for new content. In fact, our goal is to be able to discover useful new content, which could be present in new or modified files.

The approach from Song et al. [15] is closest to ours in terms of providing recommendations to knowledge workers. This approach infers abstract tasks and frequently used workflow patterns from historical user activity. It then makes recommendations based on the workflows that match current activities of the user. Although this approach extends beyond simple file matching, it cannot make predictions for new content.

In terms of features, our approach is significantly different from any of the previous approaches. Our features are derived from rich file metadata, including file name, path, extension, and file system hierarchy. These features allow our system to compare new file activities to the learned activities in the past in terms of similarity in the metadata attributes. More importantly, our system leverages metadata features to automatically generate features for collaborative filtering in a novel way so as to address the cold start problem (i.e., inability to provide recommendation for new content). In that respect, our approach is better than the traditional collaborative filtering approaches [16], [17] because they suffer from the cold-start problem.

Lastly, advanced machine learning models, e.g., factorization machines [18], topic models [19], [20], and deep neural networks [21], [22], could be used to model access patterns. However, these techniques are complementary and can be applied to make our system more effective. Nonetheless, we demonstrate that simple linear Support Vector Machine (SVM) model can be effectively used to build a practical system. Most importantly, it allows us to develop the optimization technique AFMS for significantly reducing the classification time and improving the scalability of our system. This technique, in addition to the basic modeling, and the scalability optimization approach, constitutes a novel contribution of our research.

III. THE FILE RECOMMENDATION SYSTEM

Our system uses a binary classification approach for providing file recommendations. For this purpose, it builds a separate classification model for each user of a file share. The dataset of a given user includes all the files that have been accessed by all users of the share. The label of a file in the dataset is 1 if the user accesses it, 0 otherwise. It is important to know that if a user accesses a particular file during the training phase, it is highly likely that the user will access the same file during the testing phase. Recommending such a file to the user is not very useful. Hence the main objective of our system is to discover useful but new files that may have been created or modified by other users of the same share. For this reason, the testing phase includes only those files that are new with respect to the training phase for the user. Our system then applies the model of the user to these files to predict classification labels. The rest of this section describes the features and the modeling techniques used in the system.

A. METADATA FEATURES

The metadata features include features extracted from different attributes of a file:

- **Folder**: We create a folder feature corresponding to every folder and its ancestor folders observed in the training period.[1] For a file f, we capture the folder features as a binary vector with the value 1 in the locations corresponding to the folder or the ancestor folders of f, and 0 elsewhere. These features help in learning a user's preference for certain subtrees of the file system hierarchy in a share without having to define a folder distance metric.
- **Token**: We tokenize the path name of each file using a novel tokenization approach, the details of which can be found in [6]. We construct a token vocabulary using all the tokens observed during the training phase. For extracting features, we capture tokens of a file as bag-of-words representation on the entire vocabulary.
- **Extension**: We construct an extension vocabulary based on the popular extensions observed in the training data. We then use this vocabulary to represent a file's extension with one-hot encoding based features

B. MODELING TECHNIQUES

We first build basic metadata models based on the metadata features. Using these models, we then develop an innovative approach to leverage collaboration among users for building collaborative filtering aware models that provide better performance over the basic metadata models.

1) METADATA-BASED MODELING

For each user, we train a metadata model using the metadata features. Figure 1 (a, b) shows the training and the testing phases of this approach. Considering the trade-off between accuracy and training time efficiency [6], we pick Support Vector Machine (SVM) with a linear kernel as our modeling technique because it provides high accuracy without incurring significant training time. More importantly, a linear SVM provides feature weights that capture the relative significance of individual features. We leverage this aspect in our scalability optimization technique AFMS as described in Section VI.

2) COLLABORATIVE FILTERING (CF)-BASED MODELING

Enterprise users often show a high degree of collaboration in terms of accessing common files in a share. We make this observation based on the measurement of the metric, *normalized triangle count* (Section IV-A), that captures the degree of collaboration among users. As an illustration, see Figure 2 that shows user communities in terms of social network graphs for a couple of shares.

We leverage user collaborations to build CF models that are significantly more effective than the metadata models.

[1]Please note that no features are derived for completely new attributes seen only during the testing phase.

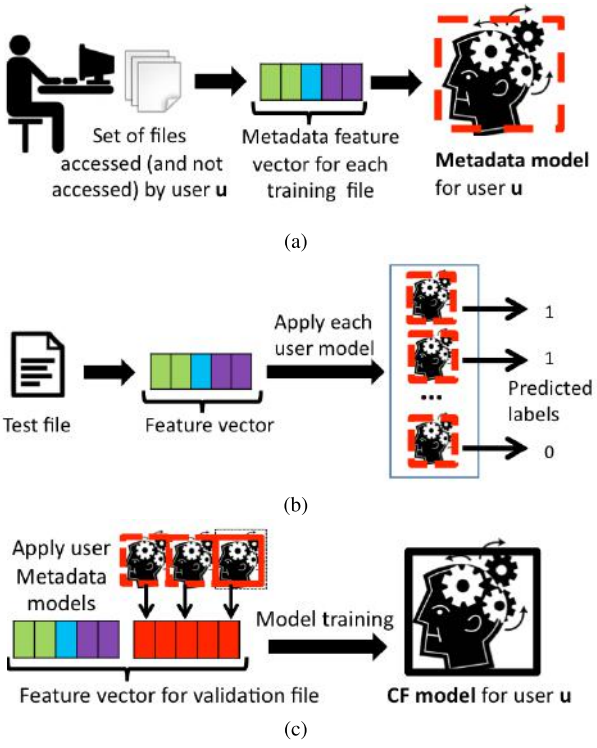
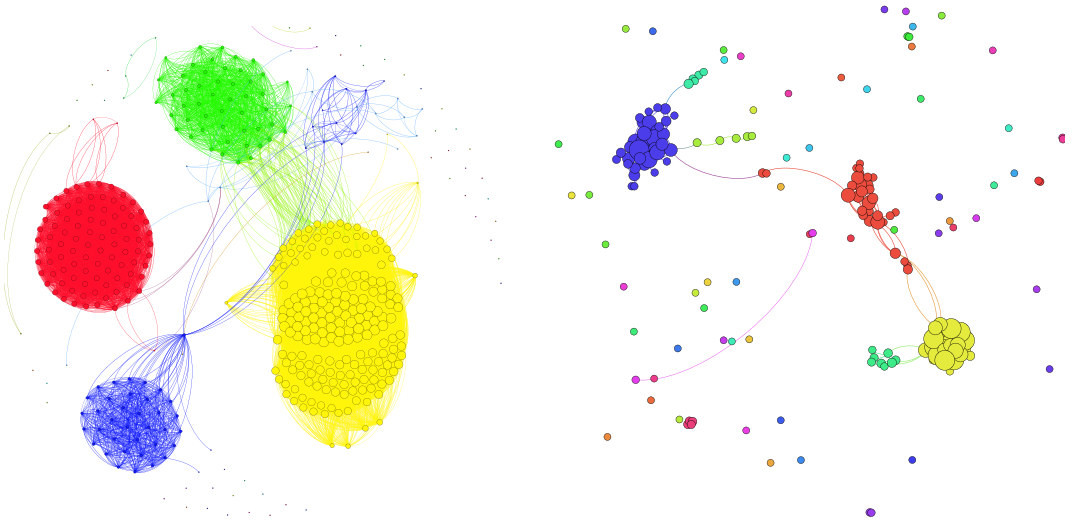


FIGURE 1. Approach Overview. (a) Training: metadata models for each user. (b) Testing: apply the trained models for all users in share. (c) Training: CF models for each user.

Here are the details of the method used in building a CF model for a user u. First, we divide the original training set into a new training set and a validation set. We then use the new training set to train a metadata model for each user. Second, for each file in the validation set, we apply the metadata models of all the users, and generate a vector of the resulting predicted labels. Third, we train the CF model for the user u using the files in the validation set. For this, we construct the feature vector for each validation file by concatenating the metadata features with the vector of the predicted labels (Figure 1(c)). For each test file observed during the testing phase, we first apply the metadata models of all the users to generate a vector of the predicted labels. Finally, we append this vector to the metadata features to create a new feature vector and feed it to the user u's CF model to obtain the final predicted label.

An interesting point to be noted here is that our CF-based modeling technique does not require a pre-defined user-user relationship or even a user similarity metric. Rather, the model can automatically learn positive or negative correlation between the activities of users, and accordingly adapt to make better predictions. For instance, if two users have similar access patterns, the CF model for one user could learn that its labels positively correlate with labels predicted by the other user's metadata model. Thus, greater the degree of collaboration, the better will be the effectiveness of the CF models. Another benefit of our technique is that, unlike pure collaborative filtering-based systems developed in the past, it does not suffer from a cold-start problem. Previous systems



(a) (b)

FIGURE 2. Social network graphs for shares show that users collaborate and tend to form communities by accessing common files. In these graphs, each node corresponds to a user, and an edge connects two users when the Jaccard index of common files accessed by those users is above 0.5. (a) Share B. (b) Share D.

TABLE 1. Statistics of shares used for evaluation.

Share	# days	# users	# files operated on	# file operations	Norm. Triangle Count
A	123	992	36,009	11M	8K
B	122	464	1,309	136K	3K
C	122	160	1,044,779	3M	<0.1
D	121	183	746	11K	951
E	66	1,288	99,733	292K	3K
F	66	937	6,911	4M	0.4
G	66	198	334	14K	3K
H	57	398	133,006	4M	45

recommend an item to a user if other *similar* users have shown a preference for that item. Therefore, these systems are incapable of recommending a completely new item because no other user has seen it before. In contrast, the novelty of our technique is in using CF features that are based on predicted labels rather than the history of past accesses.

IV. EVALUATION FRAMEWORK

This section describes the evaluation data and methodology.

A. DATA

We use file activity logs from eight network file shares of an enterprise customer for evaluation. Table 1 provides key statistics of this data. Because our recommendation system targets user collaborations, we select shares from 90^{th} percentile in terms of triangle count – a metric to capture degree of collaboration among users. Triangle count for a share is the number of triangles formed, where a triangle edge corresponds to a connection between two users based on accessing at least one common file [6]. The table shows the triangle count values normalized with respect the number of files.

Removal of Automated Activity: We observe two types of file activities in our data. The first corresponds to the activities that are manually initiated by users. These activities are characterized by a low number of file operations per hour. The second corresponds to scripted activities that are performed by automated computer programs or scripts. These activities are generally seen as bursts or exceptionally high number of file operations per hour. Since our goal is to assist knowledge workers and not automated programs, we need to remove the scripted activity from the evaluation data. For this purpose, we remove the activities by a user in an hours if the number of activities are determined to be exceptionally high. Towards this, an appropriate threshold is determined using Tukey's outlier factor [23].

Selection of Evaluation Users: Since our system helps users discover new or modified content, it is primarily targeted for active users. Therefore, for each share, we sample 30 users from the highest quartile of users in terms of number of activities, and use them in our evaluation.

B. EVALUATION METHODOLOGY

We experiment with different training and testing periods to evaluate the performance of our system under different settings.

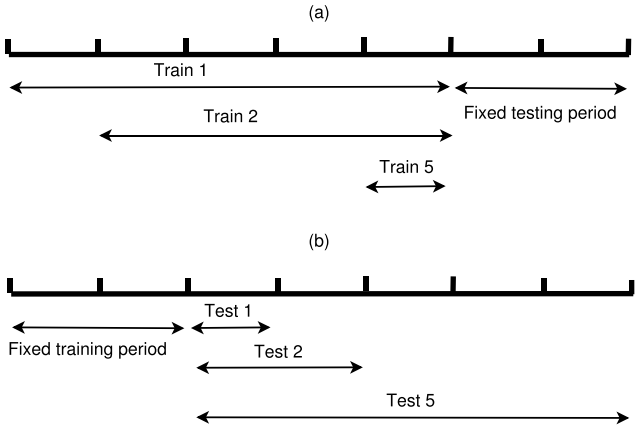


FIGURE 3. Splitting dataset into various training and testing periods. (a) Vary training periods. (b) Vary testing periods.

1) VARYING TRAINING AND TEST PERIODS

We divide the total duration of data from each share into several slices and create training and testing periods such that testing occurs immediately after training. For consistency, each share is divided into seven slices and five training and five testing periods are created as shown in Figure 3. We perform experiments with different combinations:

- **Fix testing, vary training.** A longer training window has more number of activities, thereby allowing better model training. However the activities become older as the distance between them and the testing period increases, thus preventing the model from reflecting the latest user patterns accurately. On the other hand, a small training window has less number of activities, but these activities are relatively fresh. Thus, there exists a trade-off between the size and the *recency* of the training data. We study this trade-off by varying the training duration, while keeping the testing period fixed.
- **Fix training, vary testing.** As a user's behavior evolves over time, it is expected that the effectiveness of the user's model would degrade as the testing period becomes longer. We can measure the robustness of a model by observing the rate of degradation. With this goal, we experiment with varying testing periods, while keeping the training period fixed.

2) EVALUATION METRICS

Consider an experiment with a certain combination of training and testing periods for a user u. Let $F_{true+,u}$ denote the set of files that u accessed in the testing period, and $F_{pred+,u}$ denote the set of files predicted by u's model. Then the precision and the recall of the model respectively are $\frac{|F_{true+,u} \cap F_{pred+,u}|}{|F_{pred+,u}|}$, and $\frac{|F_{true+,u} \cap F_{pred+,u}|}{|F_{true+,u}|}$. We use the following metrics to evaluate the effectiveness of the model.

- **F-score**: The F-score for a model is the harmonic mean of the precision and the recall, and provides a balanced picture of the model's effectiveness.
- **Recall@75P**: In practice, it is important for a recommendation system to have high precision because a large number of false positives may discourage a user from using the system altogether. Therefore, we additionally measure the recall values when the precision is high – 75%. This is achieved with the help of confidence scores associated with the predicted labels.

The measurements for F-score and Recall@75P are averaged across all the evaluation users to obtain AF and AR@75P respectively. For each share, we report the average and the maximum values of AF and AR@75P across the results over all combinations of training and testing periods. Please note that the maximum values provide realistic performance estimate of a properly tuned system.

V. PERFORMANCE EVALUATION

We evaluate our system using the Python-based scikit-learn library [24]. We train models on a 32 core, 64GB, 2.6GHz machine, and conduct testing on a separate 8 core, 32GB, 2.7GHz machine, leveraging multiple cores with multiprocessing. We use 3-fold cross-validation to learn the regularization parameter C for the linear SVM models by varying C over $\{10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$.

A. METADATA MODELS

Table 2 shows performance results of the metadata models. The numbers in column Max AR@75P are reasonable performance estimate of a well tuned system. The average of these numbers across all the shares, 49.4%, shows that our metadata models can capture nearly half of users' access for new files, while having under 25% wrong file recommendations. This demonstrates the effectiveness of the trained models. It should be noted that the performance of our system shows a significant variation across different shares owing to differences in rates of changes of file access patterns across the shares.

We would like to point out that our dataset includes an undesirable case where during the testing phase, a user accesses a new file that was recently created by the same user. Since recommending such a file to the user is not useful, we measure their proportion in our recommendations. The last column of Table 2 shows that most of the correctly recommended files do not fall in the above category, thereby upholding the validity of our models.

B. CF MODELS

We experimented with different strategies to sample validation files for training CF models. We use the strategy that provides the best performance, which is using the training set itself as the validation set. The results in Table 3 clearly highlight the significant gains of CF models over metadata models. The average of Max AR@75P across all shares is 70.2%, which is an improvement of a whopping 20% over the performance of metadata models. Also, it is not very surprising that the top four shares, B, D, E, and G, in terms of gains of the CF model, are found in the top five shares

TABLE 2. Performance summary for metadata models. Performance numbers are averaged over 5 iterations with random initialization of the linear SVM model training. Numbers are listed along with the standard deviations.

Share	Avg AF	Max AF	Avg AR@75P	Max AR@75P	% TP by others
A	**80.7**±0.0	**91.1**±0.0	**77.6**±0.0	**93.9**±0.0	**20.4**±0.0
B	**46.4**±0.0	**51.2**±0.0	**34.9**±0.0	**44.2**±0.0	**73.1**±0.0
C	**23.1**±0.0	**24.2**±0.0	**11.4**±0.0	**12.4**±0.0	**87.7**±0.0
D	**30.7**±0.0	**36.3**±0.0	**19.0**±0.0	**24.7**±0.0	**82.8**±0.0
E	**26.9**±0.0	**35.0**±0.0	**17.9**±0.0	**24.1**±0.0	**66.0**±0.0
F	**81.5**±0.0	**84.3**±0.0	**82.9**±0.0	**84.6**±0.0	**9.8**±0.0
G	**44.8**±0.0	**51.3**±0.0	**50.6**±0.0	**58.3**±0.0	**99.9**±0.0
H	**47.6**±0.0	**50.2**±0.0	**49.9**±0.0	**53.0**±0.0	**74.4**±0.0

TABLE 3. Performance summary for CF models. Performance numbers are averaged over 5 iterations with random initialization of the linear SVM model training. Numbers are listed along with the standard deviations.

Share	Avg AF	Max AF	Avg AR@75P	Max AR@75P	% TP by others
A	**80.7**±0.0	**100.0**±0.0	**78.5**±0.0	**100.0**±0.0	**21.2**±0.0
B	**48.6**±0.0	**77.1**±0.0	**32.2**±0.6	**62.1**±0.0	**73.1**±0.0
C	**23.5**±0.0	**36.0**±0.0	**10.1**±0.0	**16.0**±0.0	**87.5**±0.0
D	**32.3**±0.0	**47.5**±0.0	**21.5**±0.0	**38.3**±0.0	**83.3**±0.0
E	**27.6**±0.0	**41.1**±0.0	**25.3**±0.0	**100.0**±0.0	**65.7**±0.0
F	**81.5**±0.0	**87.9**±0.0	**83.3**±0.0	**89.2**±0.0	**9.8**±0.0
G	**55.5**±0.1	**76.2**±0.0	**58.0**±0.2	**89.9**±0.4	**96.6**±0.1
H	**47.6**±0.0	**57.2**±0.0	**49.6**±0.0	**66.5**±0.0	**75.4**±0.0

in terms of normalized triangle counts. Share A, which is the remaining share in the top five, doesn't show substantial improvement primarily because the performance of its metadata model is already too high.

C. TEMPORAL VARIATION IN PERFORMANCE

To study the performance variations for different lengths of training and testing periods, we focus only on the metadata models because as compared to CF models, they provide larger variation owing to lower baseline performance.

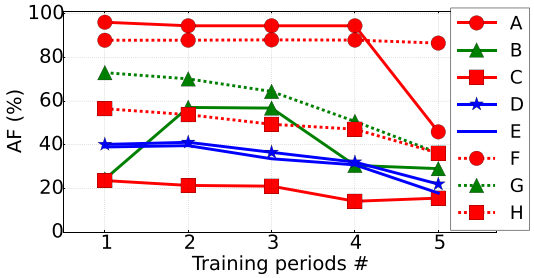


FIGURE 4. AF for metadata models with the fixed testing and varying training periods.

1) VARYING TRAINING PERIOD

Figures 4 and 5 show the performance variation of metadata models with varying training periods. The best performance for most shares is observed for index 1 that corresponds to the longest training period. As expected, the performance degrades as the training window shrinks. Performance can also degrade for a large training window wherein a model gives importance to activities that are outdated with respect to the user's recent access pattern. However, we don't observe this behavior because the longest training periods, ranging from 41 to 88 days for the eight shares, are not long enough to cause the degradation.

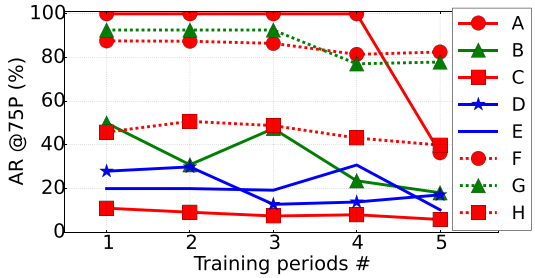


FIGURE 5. AR@75P for metadata models with the fixed testing and varying training periods.

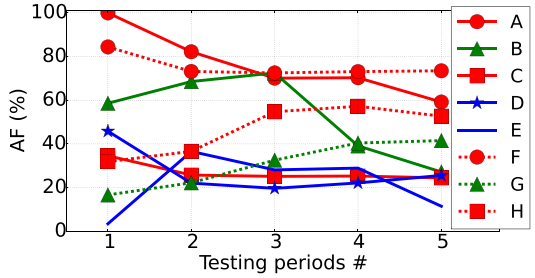


FIGURE 6. AF for metadata models with the fixed training and varying test periods.

2) VARYING TESTING PERIOD

Figures 6 and 7 show the performance variation of metadata models with varying testing periods. Beyond the testing periods with indices 1 and 2 which are the smallest and hence potentially the noisiest, the performance degrades mildly as the testing window widens. This is an excellent indicator of the robustness of our models.

Next, we perform a thorough study of the effect of training window size on the performance of the models of individual users. For this purpose, we select share D, and divide its data into 17 parts, each corresponding to one week. We remove

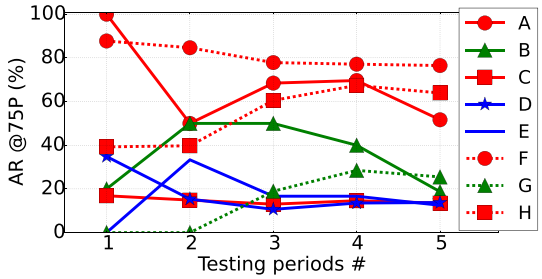


FIGURE 7. AR@75P for metadata models with the fixed training and varying test periods.

4 out of 30 users because they do not have sufficient activity in each week. We then pick the last five weeks for the testing period, and create 12 different training windows, varying from 1 week to 12 weeks. Based on the results, we plot (Figure 8) the distribution of the optimal training window size (in terms the least number of weeks) providing the highest F-score for the evaluation users of share D. The median and the mode of this window is 8 weeks even though there are longer training windows up to 12 weeks, thereby confirming our previous assertion that more training data does not necessarily translate into better performance. Additionally, we observe that the optimal window size varies for each user. The most active user (with highest activity count) requires 4 weeks, whereas the least active user requires 11 weeks. This suggests that the optimal window also depends on workload, with more active users requiring lesser training time.

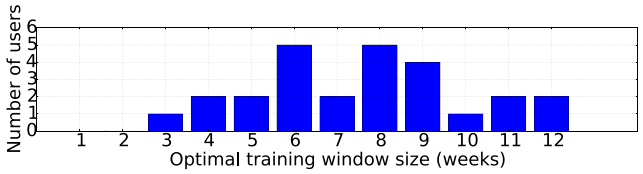


FIGURE 8. Distribution of the least number of weeks for each user of share D that gives highest F-score.

In a practical deployment, we could use an online evaluation approach to continuously tune the system based on performance and workload characteristics. Lastly, it is important to note that some of the recommended files, which are flagged as false positives in our experiments, could in fact be true positives in an actual deployment because users may genuinely find them useful upon discovering them. Hence a precision reported in this evaluation serves as the lower bound for the precision expected from an actual deployment.

D. SCALABILITY

Unlike training, which is an offline process, classification needs to be performed in real time. Therefore, we analyze time and space complexities of only the testing, i.e., the classification phase.

1) TIME COMPLEXITY

In an actual deployment, whenever a file is created or modified in a share, our system applies personalized models of each user in the share. Therefore, the runtime performance depends on the rate of file edit operations R_{edit} (in files per second) and the number of users N.

Let us compute the time taken to classify one file. Let T_{meta_model} and T_{cf_model} represent the average times (in seconds) required to apply a metadata model, and a CF model respectively. We need time $T_{meta\_feature}$ seconds to extract metadata features, and NT_{meta_model} seconds to extract collaborative filtering features by applying the metadata model of each user. We then apply the CF model of each user, which takes NT_{cf_model} seconds. Thus, the total classification time for one file is: $T_{meta\_feature}+NT_{meta\_model}+NT_{cf\_model}$ seconds.

To make approximate calculations, we replace T_{meta_model} with T_{cf_model} and ignore $T_{meta\_feature}$ because it is negligible as compared to NT_{cf_model}. Thus, it would take total time of $2NT_{cf\_model}R_{edit}$ seconds to process all the files edited in each second. Considering the sample values we observed for some of the high workload shares: $N = 16{,}000$, $R_{edit} = 5$ files/second, $T_{cf\_model} = 19 \times 10^{-3}$ second, the total time would be 3040 seconds. Hence, in order to process all the files created or modified per second, we would need at least 48 64-core machines, with each core concurrently performing classification.

Undoubtedly, the computational resources required to handle this kind of workload would amount to a significant financial cost to the enterprise. In the next section, we address this problem by presenting an optimization technique that substantially lowers the computational requirements without sacrificing classification accuracy. With this improvement, we need just one 64-core machine to manage the above mentioned heavy workload.

2) SPACE COMPLEXITY

Unlike computation, memory requirements of our system are quite modest even for heavy workloads. Considering values: $N = 16{,}000$, $R_{edit} = 5$ files/second, $F_{meta} = 35{,}000$ (highest number of metadata features in our datasets), and $S = 4$ bytes (size of value and weight of a feature), we would need maximum $(F_{meta} + N)R_{edit}S$ i.e., nearly 1MB memory for all the features in the worst case. Note that the features of a file remain the same for all the user models. Each metadata model is represented by F_{meta} weights, and each CF model is represented by $F_{meta} + N$ weights. Thus, the total memory to store metadata and CF models of all the users is: $(2F_{meta} + N) \times NS$ i.e., approximately 5 GB. In practice, the required memory would be much smaller because collaborative filtering will be applied to a small subset of users that show high degree of collaboration.

VI. ACTIVE FEATURES-BASED MODEL SELECTION (AFMS)

To speed up the classification time, our optimization technique AFMS leverages properties of a soft-margin linear SVM model. In this model, the predicted label for a file f

TABLE 4. Statistics for metadata models, averaged across 30 evaluation users and different training periods (Section IV-B.1), showing how AFMS is effective.

Share	% models with negative intercept	% active features	% features with positive weight	% models per positive feature
A	100.0	0.10	2.8	2.8
B	98.9	1.90	2.5	3.6
C	100.0	0.10	0.2	0.2
D	100.0	6.60	0.8	0.8
E	96.1	1.40	15.2	18.5
F	98.3	1.60	4.2	5.8
G	74.5	6.10	21.7	40.8
H	100.0	0.01	4.1	4.1
Avg.	**95.9**	**2.2**	**6.4**	**9.6**

is determined using the score:

$$g(f) = w_0 + \sum_{j=1}^{F} w_j * x_j \quad (1)$$

Here, F is the number of features, w_0 is the intercept, w_j is the j^{th} feature weight in the model, and x_j is the value of the j^{th} feature of the file. The model predicts label 1 if $g(f) \geq 0$, and 0 otherwise. Given that our features have non-negative values, without applying the model we can predict label 0 when the following conditions are met:

- **condition 1**: The intercept is negative, i.e., $w_0 < 0$, and
- **condition 2**: None of the weights of active (non-zero) features is positive.

Under these conditions, $g(f)$ is always negative, and hence the predicted label is 0. Otherwise, we need to apply the model to obtain the classification label, which could be 0 or 1. The AFMS technique uses this key intuition to save the cost of applying classification models. Of course, its effectiveness depends on the number of times the conditions are matched. Unsurprisingly, as our experiments show, this happens quite often (refer Table 4):

- **condition 1**: Around 96% of the metadata models have negative intercept (i.e., $w_0 < 0$). This is because a typical user accesses only a small fraction of files. Owing to this natural imbalance, the model favors 0 as the prediction label, resulting in a negative intercept.
- **condition 2**: Our data shows sparsity of features with an average of only 2.2% of metadata features being active (i.e., $x_j > 0$) in the files. Also, the average ratio of the features with positive weights (i.e., $w_j > 0$) per model is only 6.4%, and each of these features on an average has a positive weight in less than 10% of the models. The combination of these three facts significantly lowers the probability of finding active features with positive weights.

The favorable statistics result in matching the conditions quite often, thereby leading to faster testing of files because the cost of checking the conditions is negligible. Note that faster testing of files is referred to as better performance in the discussion below. While we have provided the above statistics based on the metadata models, similar trends are observed for CF models as well. Section VI-B shows the actual gains in the performance for both metadata and CF models. The next section describes the algorithm to implement AFMS.

A. AFMS ALGORITHM

Our technique extends the basic idea to make the optimization tunable so that it can potentially trade-off model correctness for even better classification efficiency. In order to do so, we define that *a feature with index j triggers a model* when $w_j \geq \tau$ or $w_0 \geq 0$ where w_0 and w_j are the intercept and the j^{th} feature weight of the model respectively. Here, we use the threshold parameter $\tau \geq 0$ to make the technique tunable. Given a test file, a model would be applied on it if any of the active features of the test file *trigger* the model. For $\tau = 0$, there is no loss of predictive correctness, and yet there is performance gain due to the optimization. However, as τ increases, fewer models are triggered, and hence there is further improvement in the performance, albeit at the cost of predictive correctness. With fewer models being applied, there will be more number of 0 as the predicted labels, and hence the recall is expected to degrade. However, precision and likewise F-score can degrade or improve depending on the correctness of the recommendations. Section VI-B shows that we can gain significant efficiency at marginal cost of model correctness. Furthermore, the technique can be made adaptive by varying τ based on the rate of file edit operations at a given time.

Please refer to Algorithm 1 to understand variable definitions, and the details of the algorithm. As part of preprocessing, which is done right after the training phase, for each feature we compute the set *FTM* (Feature Triggered Models) consisting of models that are triggered by the feature (Figure 9). For a share, the algorithm computes a separate FTM for metadata and CF models.

The input to the classification procedure is the array $X[1 \ldots F_{meta}]$ of metadata features corresponding to the test file. The procedure produces the outputs: an array $L[1 \ldots N]$ consisting of prediction label for each of the N users by CF models, and an array $Y[1 \ldots N]$ consisting of prediction label by metadata models. The procedure initializes the output arrays with 0 values as the default labels. It also initializes sets M and C that are later used to collect triggered metadata and CF models respectively. The first step obtains the set of metadata models that are triggered by the active features. The second step applies these metadata models to compute metadata predicted labels, which are also the CF features Y. The third step appends the CF features to the metadata features. The fourth step obtains the set of CF models that are triggered by the active features. Finally, the last

Algorithm 1 AFMS Classification

Variables definitions:

N : Number of users
F_{meta} : Number of metadata features
F_{CF} : Number of features in CF models = $F_{meta} + N$
$w_{\sigma(i,j)}$: Weight of j^{th} feature in i^{th} user model, $1 \leq i \leq N$, $1 \leq j \leq F_\sigma$, $\sigma \in \{meta, CF\}$

Offline pre-processing to be done right after model training:

$FTM_\sigma(j) = \{i \mid w_{\sigma(i,j)} \geq \tau$, or $w_{\sigma(i,0)} > 0\}$

Classification:

Input:
$X = [x_1, x_2, \ldots, x_{F_{meta}}]$: Metadata feature vector of the test file f

Output:
$Y = [y_1, y_2, \ldots, y_N]$: Predicted classification label (0 or 1) by metadata models for each user

$L = [l_1, l_2, \ldots, l_N]$: Predicted classification label (0 or 1) by CF models for each user

Procedure:

$M = C = \{\}$ \\initialize triggered metadata and CF models
for $i = 1$ **to** N: \\initialize CF features and output labels
    $Y[i] = L[i] = 0$

for $j = 1$ **to** F_{meta}: \\step1: obtain triggered metadata models
    if $(X[j])$:
        $M = M \cup FTM_{meta}(j)$

for i **in** M: \\step2: compute CF features
    $Y[i] = \text{ApplyMetadataModel}(i, X)$

$X = [X, Y]$ \\step3: append CF features to metadata features

for $j = 1$ **to** F_{CF}: \\step4: obtain triggered CF models
    if $(X[j])$:
        $C = C \cup FTM_{CF}(j)$

for i **in** C: \\step5: compute CF model labels
    $L[i] = \text{ApplyCFModel}(i, X)$

step applies these CF models to compute the CF predicted labels (L).

If a model is not applied, the default label 0 becomes the predicted label, and the algorithm assigns a random negative value as the confidence score for the label. We need the confidence score because it is used to determine the relative ranking of test files, which is then used in the computation of recall at 75%. As a side effect, it is possible that AR@75P for AFMS with $\tau = 0$ may differ from AR@75P computed without AFMS.

1) COMPLEXITY ANALYSIS

In AFMS classification procedure, steps 2 and 5 are the most expensive ones, requiring $O(N)$ classifications. Thus, with the rate of file edit operation R_{edit}, the algorithm needs to perform $O(NR_{edit})$ classifications. The optimization does not change the worst case time complexity because it is possible that all the models are triggered for a particular dataset. However, given the presence of natural class imbalance in user-activity datasets, our optimization significantly improves the average case time complexity. For instance, with $\tau = 0.5$, testing of a single file on an average requires approximately 0.5 metadata- and 0.2 CF-based classifications out of 30 evaluation users. This reduces the classification time of metadata and CF models by 62 and 169 times respectively. The next section provides detailed performance results.

B. AFMS EVALUATION

We measure the performance improvement using a simple metric:

$$Speed\text{-}up = \frac{\#\text{ models applied without AFMS}}{\#\text{ models applied with AFMS}}. \quad (2)$$

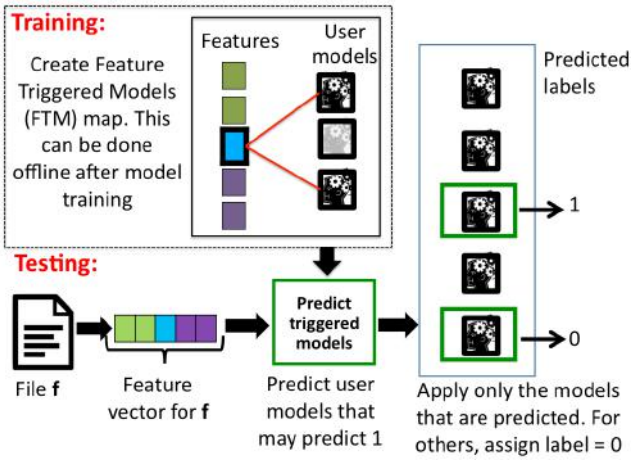


FIGURE 9. AFMS technique first creates FTM in an offline manner. The classification step applies only those models that are triggered by active features.

Speed-up provides a direct estimate of reduction in classification time.

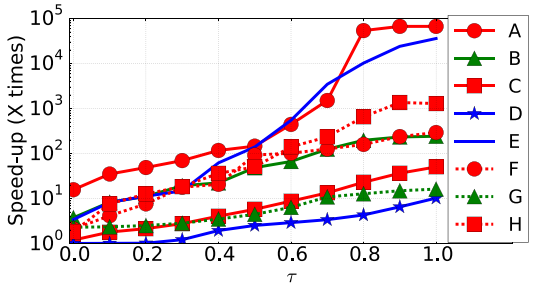


FIGURE 10. Speed-up vs. τ for metadata models.

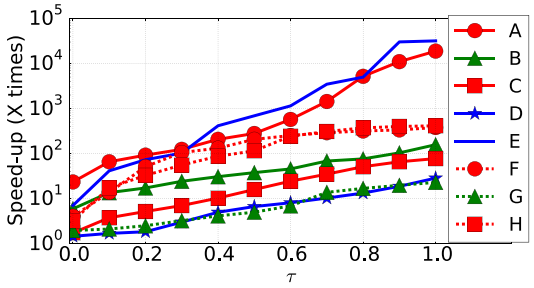


FIGURE 11. Speed-up vs. τ for CF models.

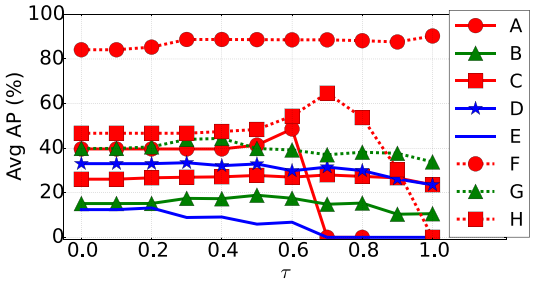


FIGURE 12. Precision vs. τ for metadata models.

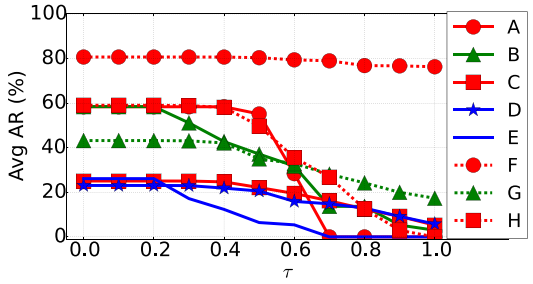


FIGURE 13. Recall vs. τ for metadata models.

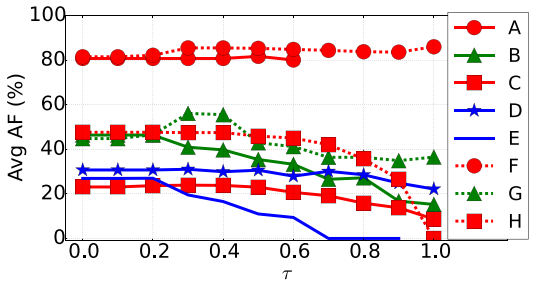


FIGURE 14. F-score vs. τ for metadata models.

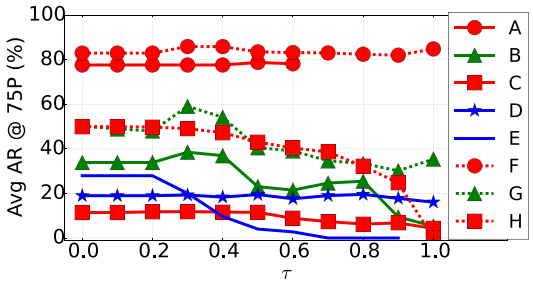


FIGURE 15. Recall@75%Precision vs. τ for metadata models.

Figures 10-19 show the effect of τ on speed-up, and various correctness metrics including average precision, recall, and F-score. The numbers are averaged across all the evaluation users, and all training and testing periods for individual shares.

In general, we observe that speed-up increases exponentially with respect to τ, whereas the correctness metrics degrade almost linearly, with a mild slope, as τ increases. This clearly shows that AFMS can obtain significant efficiency at marginal cost of accuracy.

As expected, for $\tau = 0$, AFMS does not lose accuracy, e.g., CF model Avg AF values from Table 3 and Figure 18 remain the same for all shares. And yet, AFMS provides 4 times average speed-up across all shares for metadata models, and 6 times average speed-up for CF models. The speed-up for $\tau = 0$ is as high as 15 times and 23 times for share A for metadata and CF models respectively. With $\tau = 0.5$, the average speed-up for metadata models is 62 times, and for CF models is 169 times. The most gain is observed for share A as 147 times for metadata models, and 280 times for

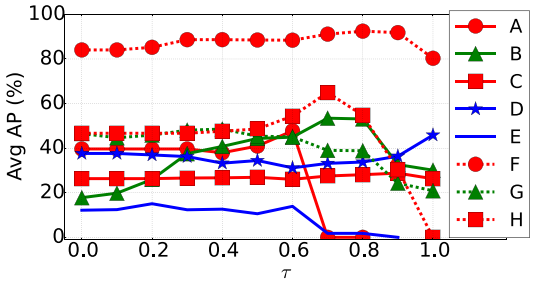


FIGURE 16. Precision vs. τ for CF models.

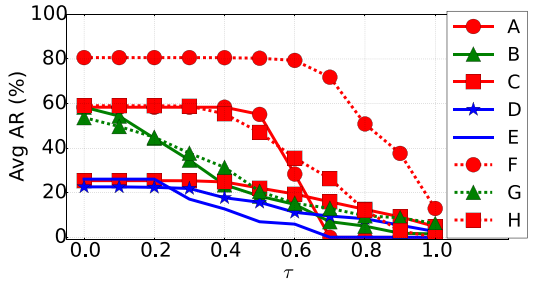


FIGURE 17. Recall vs. τ for CF models.

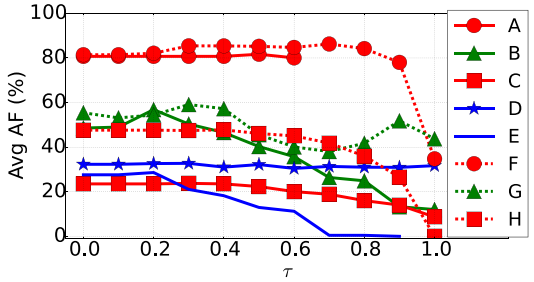


FIGURE 18. F-score vs. τ for CF models.

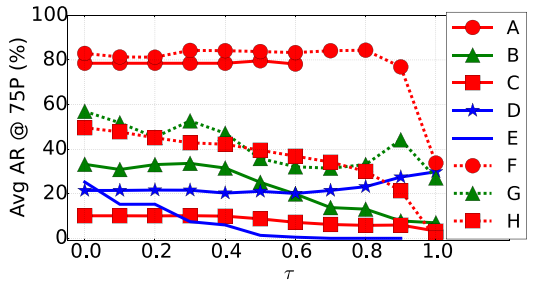


FIGURE 19. Recall@75%Precision vs. τ for CF models.

CF models. However, this threshold affects the correctness, dropping averages across shares. For example, for CF models, the Avg AF changed from 49.6% to 45.8%, Avg AR from 48% to 33.3%, Avg AR@75P from 44.8% to 36.9%, but slightly improving Avg AP from 38.9% to 42.5%. These results confirm the expected effect as discussed in Section VI-A.

VII. CONCLUSION

This paper presents a system to assist knowledge workers in discovering useful new or modified content. The system builds personalized user models using features derived from file metadata and user collaboration. Our experiments showed that the basic modeling approach does not scale well under heavy workloads. To address this problem, we developed a novel optimization technique that improves runtime performance without sacrificing prediction accuracy. We also showed how the technique can be adapted to improve the performance significantly at marginal cost of the predictive correctness.

For future work, we could improve our system along different directions. The current system uses only the file metadata information to provide recommendations. In future, the content of files could also be utilized to improve the effectiveness of the system. In addition, leveraging interactions observed between the features, we could reduce dimensionality of the data for better efficiency and effectiveness. We could implement a weighing scheme that gives more importance to the recent activity as it is more reflective of the current preferences of users. Finally, we could incorporate features that capture producer-consumer relationship between users.

REFERENCES

[1] *Big Data Survey*, IDG Enterprise, Framingham, MA, USA, 2014.
[2] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," in *Proc. IDC iView, IDC Analyze Future*, 2012, pp. 1–16.
[3] (2015). *Salesforce*. [Online]. Available: http://www.salesforce.com/
[4] (2015). *Microsoft Office 365*. [Online]. Available: http://en.wikipedia.org/wiki/Office_365
[5] P. M. Leonardi, M. Huysman, and C. Steinfield, "Enterprise social media: Definition, history, and prospects for the study of social technologies in organizations," *J. Comput.-Mediated Commun.*, vol. 19, no. 1, pp. 1–19, 2013.
[6] C. Verma, M. Hart, S. Bhatkar, A. Parker-Wood, and S. Dey, "Access prediction for knowledge workers in enterprise data repositories," in *Proc. Int. Conf. Enterprise Inf. Syst.*, 2015, pp. 1–12.
[7] A. Amer, D. D. E. Long, J.-F. Paris, and R. C. Burns, "File access prediction with adjustable accuracy," in *Proc. Int. Perform. Conf. Comput. Commun. (IPCCC)*, Apr. 2002, pp. 131–140.
[8] P. Xia, D. Feng, H. Jiang, L. Tian, and F. Wang, "FARMER: A novel approach to file access correlation mining and evaluation reference model for optimizing peta-scale file system performance," in *Proc. Int. ACM Symp. High-Perform. Parallel Distrib. Comput. (HPDC)*, 2008, pp. 185–196.
[9] T. Kroeger and D. D. E. Long, "Design and implementation of a predictive file prefetching algorithm," in *Proc. USENIX Annu. Tech. Conf.*, Jan. 2001, pp. 105–118.
[10] T. Yeh, D. D. E. Long, and S. A. Brandt, "Increasing predictive accuracy by prefetching multiple program and user specific files," in *Proc. Annu. Int. Symp. High Perform. Comput. Syst. Appl. (HPCS)*, 2002, pp. 12–19.
[11] T. Yeh, D. D. E. Long, and S. A. Brandt, "Performing file prediction with a program-based successor model," in *Proc. Modeling, Anal. Simulation Comput. Telecommun. Syst. (MASCOTS)*, Aug. 2001, pp. 193–202.
[12] T. Yeh, D. D. E. Long, and S. A. Brandt, "Using program and user information to improve file prediction performance," in *Proc. Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Nov. 2001, pp. 1–9.
[13] G. A. S. Whittle, J.-F. Paris, A. Amer, D. D. E. Long, and R. Burns, "Using multiple predictors to improve the accuracy of file access predictions," in *Proc. Int. Conf. Massive Storage Syst. Technol. (MSST)*, Apr. 2003, pp. 230–240.
[14] J.-F. Pâris, A. Amer, and D. D. E. Long, "A stochastic approach to file access prediction," in *Proc. Int. Workshop Storage Netw. Archit. Parallel I/Os (SNAPI)*, 2003, pp. 36–40.
[15] Q. Song, T. Kawabata, F. Itoh, Y. Watanabe, and H. Yokota, "File and task abstraction in task workflow patterns for file recommendation using file-access log," *IEICE Trans. Inf. Syst.*, vol. E97-D, no. 4, pp. 634–643, 2014.

[16] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.
[17] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. Conf. Uncertainty Artif. Intell.*, 1998, pp. 43–52.
[18] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2010, pp. 995–1000.
[19] R. Nagori and G. Aghila, "LDA based integrated document recommendation model for e-learning systems," in *Proc. Int. Conf. Emerg. Trends Netw. Comput. Commun. (ETNCC)*, Apr. 2011, pp. 230–233.
[20] M. Ovsjanikov and Y. Chen, "Topic modeling for personalized recommendation of volatile items," in *Proc. Eur. Conf. Mach. Learn. Principles Pract. Knowl. Discovery Databases*, 2010, pp. 483–498.
[21] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. ACM Int. Conf. Mach. Learn.*, 2007, pp. 791–798.
[22] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[23] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan, "Statistical techniques for online anomaly detection in data centers," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 2011, pp. 385–392.
[24] (2015). *Scikit-Learn Machine Learning in Python*. [Online]. Available: http://scikit-learn.org/

CHETAN VERMA is a final year Ph.D. Student at the University of California San Diego. His research interests include machine learning, data mining, predictive analytics, multimedia retrieval, and enterprise information systems. He has been a part of research internships at Symantec Research Labs, Yahoo Labs, and Samsung Research. He has nine referred publications and four submitted patents.

MICHAEL HART is a Technical Director in the Symantec Research Labs, who has made significant contributions to Symantec across many different product lines since joining in 2011. He developed a linguistically motivated, machine learning driven, zero history URL reputation technique that is currently leveraged by Managed Security Services and MessageLabs, which blocks tens of millions of spam emails each month. He contributed to the fundamental technology that drives Symantec DLP's VML technology and provided a crucial update to the detection rules used for HIPAA policy violations with a novel technique that automatically extracts medical terminology. He developed many critical features for the Data Insight 4.0 release, including social network analysis for file user activity to identify anomalous behavior that may be indicative of insider activity. He has eight referred academic publications in security and machine learning, 11 patents submitted, three granted, and a member of the NSA Enduring Security Framework Insider Threat working group. His research interests lie in the intersection of machine learning, natural language processing and security.

SANDEEP BHATKAR received the B.Tech. degree in computer science and engineering from IIT Bombay, and the Ph.D. degree in computer science from Stony Brook University, New York. His Ph.D. research developed software diversity-based defense techniques (such as address space randomization and data space randomization) to combat exploits of memory errors. He is a Researcher with Symantec Research Labs, Mountain View, CA. At Symantec Research Labs, he built tools and techniques for automated malware analysis and labeling. His recent focus is on data analytics for detection and visualization of advanced persistent threats. His research interests are in the areas of software security, program analysis, and machine learning.

ALEATHA PARKER-WOOD received the Ph.D. degree in computer science from the Storage Systems Research Center at UC Santa Cruz, in 2014. Prior to graduate school, she was a Software Engineer at a number of Silicon Valley companies, including Adobe, Borland, and several startups. She is a Researcher with Symantec Research Labs, focusing on datacentric security and anomaly detection. Prior to Symantec, she held a postdoctoral position in the databases group at Le Conservatoire National des Arts et Metiers in France. She has interests in security, data management, distributed systems, and applied machine learning.

SUJIT DEY received the Ph.D. degree in computer science from Duke University, Durham, NC, USA, in 1991. He served as the Chief Scientist of Mobile Networks with Allot Communications from 2012 to 2013. He founded Ortiva Wireless in 2004, where he served as its founding CEO and later as the CTO till its acquisition by Allot Communications in 2012. Prior to Ortiva, he served as the Chair of the Advisory Board of Zyray Wireless till its acquisition by Broadcom in 2004. Prior to joining the University of California at San Diego (UCSD) in 1997, he was a Senior Research Staff Member with NEC Research Laboratories, Princeton, NJ, USA. He is a Professor with the Department of Electrical and Computer Engineering, UCSD, where he heads the Mobile Systems Design Laboratory, which is engaged in developing innovative mobile cloud computing architectures and algorithms, adaptive multimedia and networking techniques, low-energy computing and communication, and reliable system-on-chips to enable the next-generation of mobile multimedia applications. He is the Director of the UCSD Center for Wireless Communications. He also serves as the Faculty Director of the von Liebig Entrepreneurism Center, and is affiliated with the Qualcomm Institute. He has co-authored over 200 publications, including journal and conference papers, and a book on low-power design. He holds 18 U.S. patents, resulting in multiple technology licensing and commercialization. He has been a recipient of six IEEE/ACM best paper awards, and has chaired multiple IEEE conferences and workshops.