Access Prediction for Knowledge Workers in Enterprise Data Repositories

Chetan Verma[1], Michael Hart[2], Sandeep Bhatkar[2], Aleatha Parker-Wood[2], and Sujit Dey[1]
[1]*Electrical and Computer Engineering, University of California San Diego, San Diego, CA, USA*
[2]*Symantec Research Labs, Mountain View, CA, USA*
{*cverma, sdey*}*@ucsd.edu,* {*Michael_Hart, Sandeep_Bhatkar, Aleatha_ParkerWood*}*@symantec.com*

Keywords: Information Retrieval, Machine Learning, Enterprise, File systems

Abstract: The data which knowledge workers need to conduct their work is stored across an increasing number of repositories and grows annually at a significant rate. It is therefore unreasonable to expect that knowledge workers can efficiently search and identify what they need across a myriad of locations where upwards of hundreds of thousands of items can be created daily. This paper describes a system which can observe user activity and train models to predict which items a user will access in order to help knowledge workers discover content. We specifically investigate network file systems and determine how well we can predict future access to newly created or modified content. Utilizing file metadata to construct access prediction models, we show how the performance of these models can be improved for shares demonstrating high collaboration among its users. Experiments on eight enterprise shares reveal that models based on file metadata can achieve F scores upwards of 99%. Furthermore, on an average, collaboration aware models can correctly predict nearly half of new file accesses by users while ensuring a precision of 75%, thus validating that the proposed system can be utilized to help knowledge workers discover new or modified content.

1 INTRODUCTION

Enterprise knowledge workers are inundated with new options for conducting their work with the rise of Enterprise Social Networks [Leonardi et al., 2013] and cloud based applications [Salesforce, 2015, Office365, 2015] alongside traditional technologies such as email, source control repositories, network file servers, and office software suites. Enterprises are also embracing new computing devices such as mobile devices and tablets in addition to existing personal computers, laptops, workstations and servers. The amount of enterprise data grows significantly each year: studies estimate that unstructured data grows annually by 40-50% [Gantz and Reinsel, 2012]. The fragmentation in the tools and devices used to work and the sheer growth of data places increasingly unrealistic demands on knowledge workers to keep up with the influx of data. In fact, it has been reported that 65% of users have felt at times overwhelmed by the amount of incoming data [IDG Enterprise, 2014].

This paper presents a system that utilizes machine learning and natural language processing to automate the discovery of important new or modified content and identify which subset of users will likely use or benefit from it. The system is designed for file servers and is evaluated with activity collected over eight network file servers from an enterprise customer. Enterprises use file servers for a myriad of purposes including storing application data, back up, enabling collaboration, and hosting personal home directories. The proposed system will support a wide range of applications, such as recommender systems or server cache management systems, by providing predictions about what data will likely be accessed in the near future.

The system bases its predictions on user activity and content metadata. We track content accessed by users over a specified training interval. Data (i.e. access to a particular piece of content) are represented by a set of features that include path components (e.g., parent and ancestral directories), keywords in the path, and extension. Each datum represents an instance in training our model. Combining this training data with the files specifically accessed by the user, this system builds personalized models to predict future file accesses. While traditional approaches for file access prediction such as [Yeh et al., 2001a, Yeh et al., 2001b] cannot be applied to recommend new files, the proposed user

model based approach is generalizable enough to be applied even to content that has not been accessed before, or is newly created. Additionally, analysis of file activity yields the insight that very regular patterns of collaboration occur. The paper demonstrates a method in which an individual's prediction precision and recall can be greatly improved by incorporating the predictions of all other user models.

This work makes the following contributions:

- Observations about the nature of enterprise file activity
- A system that analyzes file activity and metadata and applies machine learning and natural language processing to provide predictions
- A strategy to combine the personalized models of multiple file users to improve the predictions of individual users

The paper is organized as follows. Section 2 provides observations about file activity and preprocessing. Section 3 details the feature space and Section 4 describes the construction of metadata and collaborative filtering-based user models. Section 5 presents an evaluation of the user models and Section 6 discusses the contributions and characteristics of the features used in the models, and scalability and deployment related aspects. Section 7 identifies related work followed by a discussion on directions for future work in Section 8. Section 9 concludes.

2 DATA

For this work, the system focuses on network file servers from corporate enterprises with significant user collaboration. We select network file servers based on social network analysis and use an affinity function where an edge connects two users if the users have accessed at least one file in common. Collaboration is measured by the triangle count, the number of triangles formed by sets of three users mutually connected to each other. The file servers are selected from the 90^{th} percentile based on the triangle count. The normalized triangle count is calculated by averaging triangles over all files in a share. For the purposes of this paper, eight file servers are selected to evaluate the system. Table 1 compares different statistics from each of the eight servers. Note that the statistics are calculated before removal of scripted activity as described below. Removing scripted activities is an important step since our intent is to model the file access patterns of users for applications such as file recommendation, and not necessarily to model the file access patterns of automated processes.

2.1 Detecting scripted activities

Our data suggests that users access files in at least two modalities. Normal file access activity for users typically consists of a small number of file accesses in a short period of time, such as an hour. Another mode is when a large number of files are accessed which manifest as a sudden burst of activities. In order to remove such scripted activities, we record the number of activities performed by every user in each hour session, and label the sessions having exceptionally large number of activities as scripted. In order to obtain an appropriate threshold for this purpose, we utilize Tukey's outlier factor [Wang et al., 2011] as shown in Algorithm 1. A $(user, hour)$ tuple is flagged as scripted if the number of activities corresponding to the tuple exceeds the threshold calculated as

$$Q3 + k \times IQR. \quad (1)$$

Here Q3 is the third quartile and IQR is the interquartile range of the number of activities of (user, hour) tuples. We empirically set k to 5. Since our focus is on removing scripted activities, if the threshold determined using Tukey's outlier factor is less than 1000 activities per user per hour, we use 1000 as the threshold. That is, if a user performs more than 1000 file activities per hour, we label his/her file activities in that hour as scripted. Based on such an approach, if an abnormally high number of file activities are performed from a user's account in an hour, it can be reasonably expected that at least a large fraction of them were not performed by the user directly and the result of an automated or scripted activities. The activities that are determined to be scripted are removed from the file activity log over which models are trained and evaluated. Table 1 provides the proportion of file activities remaining in each share after scripted activities are removed. From this point, we only provide results on the shares after scripted activity is removed.

Algorithm 1 Detecting scripted activities

Input: Num-activities(u, h) tuples $\forall$ user u in share, $\forall$ hour h in dataset. k = 5

Q1= first quartile of Num-activities(u, h) $\forall u, h$

Q3= third quartile of Num-activities(u, h) $\forall u, h$

IQR=Q3-Q1

Output: Threshold = Q3 + $k \times$ IQR

2.2 Metadata tokenization

File metadata in enterprise environments does not share consistent capitalization or delimitation. For example, in the Directory Services, a group such as

Share	Sample period (days)	Users	Files operated on	Total file operations	% after burst removal	Triangle Count	Normalized Triangle Count	Create%	Read%	Write%	Delete%
A	123	992	36,009	11M	99.9	280M	8K	2.3	92.7	2.8	2.2
B	122	464	1,309	136K	99.9	4M	3K	0.7	38.9	60.0	0.4
C	122	160	1,044,779	3M	9.3	50K	<0.1	0.9	96.8	1.6	0.6
D	121	183	746	11K	99.8	710K	951	5.4	88.4	5.9	2.6
E	66	1,288	99,733	292K	16.3	263M	3K	15.6	50.2	17.3	16.9
F	66	937	6,911	4M	100.0	3K	0.4	0.2	99.5	0.2	0.2
G	66	198	334	14K	100.0	1M	3K	0.2	98.7	0.7	0.4
H	57	398	133,006	4M	93.6	6M	45	15.3	57.8	10.0	16.8

Table 1: Statistics of shares used for evaluation. Degree of collaboration in shares is measured using normalized triangle counts as outlined in Section 2. The types of events (create, read, write, delete) offer useful insight into the use of each share. For example, share B observes high "write" workload and may be used as a repository for logs.

Share	Min	Q1	Q2	Q3	Max
A	1	13	17	41	585
B	1	2	3	5	65
C	1	4	17	54	504,692
D	1	2	3	5	65
E	1	1	1	1	15,478
F	1	1	1	2	1,182
G	1	3	6	10	307
H	1	6	17	101	29,604

Table 2: Minimum, median, maximum, first (Q1) and third (Q3) quartile of number of unique files operated upon weekly

Share	Most common extensions (% file activities)				
A	tmp (78)	dat (19)	gnt (2)	pm (< 1)	docx (< 1)
B	pdf (16)	log (12)	pma (11)	bak (10)	txt (8)
C	pdf (85)	doc (4)	xls (3)	tif (1)	msg (1)
D	xls (87)	xlsx (6)	htm (3)	pdf (1)	dat (1)
E	pdf (92)	doc (2)	deploy (1)	resources (1)	vb (1)
F	txt (70)	stk (19)	xls (10)	exe (< 1)	log (< 1)
G	cat (38)	bat (18)	lnk (8)	dll (8)	mpr (7)
H	xls (35)	ret (25)	rpt (4)	unv (3)	pdf (3)

Table 3: Popular file extensions in different shares that users access. Share B sees large workload on log files and Windows Performance Monitor (.pma) data files, corresponding with observation in Table 1.

ABC administrators could be recorded as "ABC admins", but referred to in a directory name as "ABCADMINS". Therefore, extracting meaningful entities from metadata requires tokenization that is not only sensitive to natural language delimiters (e.g. whitespace), but also the likely concatenation of entities in alphanumeric substrings.

This system employs a heuristic based approach to more traditional Natural Language Processing morphological extraction [Bybee, 1985]. The system first tries to develop a list of organizational specific entities (which may be unique to only this organization) by analyzing a definitive entity source. A directory service such as Active Directory [Active Directory, 2015] is an example of such a source, which we use for training our system. Entities could be mined from group names, lines of businesses, and other user and group metadata. A ground truth set of entities is blindly constructed by splitting on a set of hard delimiters. In our case, since the organization resides primarily in the United States, we used whitespace and non-alphanumeric characters to split entries. We compute a frequency dictionary for all entities and denote D as the total number of entities extracted. This frequency dictionary is denoted as the internal dictionary.

In addition to an organizational frequency dictionary, the system will also leverage a general word frequency dictionary. Not all entities in metadata could come from the authoritative source mentioned in the previous paragraph. A frequency dictionary computed from the frequency of terms in general usage, such as Corpus of Contemporary American English [coca, 2008], will provide information about the likely tokenization a user would also arrive at. This frequency dictionary is denoted as the external dictionary.

Tokenization of metadata leverages dynamic programming to address the possible concatenation of multiple entities. The algorithm starts by first splitting the metadata on hard delimiters, such as whitespace. For each substring, the algorithm applies another split if the substring matches a known regular expression for concatenating entities into one continuous alphanumeric sequence. In this system, we consider variations of CamelCase [CamelCase, 2015]. After splitting on hard delimiters and regular expressions, we apply dynamic programming to determine if the substring should be split into two or more tokens. The algorithm iterates by finding the optimal tokenization of each prefix, starting with the prefix of length 1. A split is scored by multiplying the optimal solution for the left side (e.g. prefix) and the score for the right side. The score of the right side is a linear interpolation of the internal and external frequency dictionaries:

$$\beta * f_{internal}(term) * (1-\beta) * f_{external}(term). \quad (2)$$

Empirical results show a β of 0.9 works well in prac-

tice. A penalty factor for either dictionary is applied when a term is not found. The penalty factor used in this paper is $\frac{1}{D*2^{len(term)}}$. Note that this approach favors tokenization with internal entities and fewer tokens.

3 FEATURES

As mentioned in Section 1, the proposed system is trained over primarily three types of features. For each file f, we show below how these features are calculated.

1. **Folder features.** Typically, the files in a file system are organized into folders and sub-folders, with the intent of placing together files that are expected to be used together or are related to the same task. Our aim is to capture the folder that a file belongs to, and to capture the proximity between folders with respect to the file system hierarchy without having to explicitly define a folder to folder proximity metric. Let's say that the folders in the file system are represented by $\mathcal{F}$ where $\mathcal{F}(i)$ represents the i^{th} folder. The folder features of a file f are represented as the vector $\mathbf{X_{f,\mathcal{F}}}$ where the cardinality of $\mathbf{X_{f,\mathcal{F}}}$ is the number of folders in the file system, i.e., $\mid \mathcal{F} \mid$. The i^{th} element of $\mathbf{X_{f,\mathcal{F}}}$, i.e., $X_{f,\mathcal{F}}(i)$ is 1 if $\mathcal{F}(i)$ lies in the path of file f. For example, if f is '$\backslash folder1\backslash folder2\backslash filename$' then $\mathbf{X_{f,\mathcal{F}}}$ would be [1, 1, 0, 0, 0, ...] where the first index of $\mathbf{X_{f,\mathcal{F}}}$ corresponds to the folder '$\backslash folder1\backslash$' and the second index corresponds to the folder '$\backslash folder1\backslash folder2\backslash$'.
2. **Token features.** In addition to folder organization, the nomenclature of files and folders also provides useful insights into file content and categorization. In order to capture this, we tokenize the file path including the file name, and construct a vocabulary based on the popular tokens (keywords). Each file is then represented as a bag of words based on the constructed vocabulary, and based on the tokens present in its path name. Specifically, if $\mathcal{T}$ is the set of tokens in the constructed vocabulary, then the token features of file f are represented as the vector $\mathbf{X_{f,\mathcal{T}}}$ where the cardinality of $\mathbf{X_{f,\mathcal{T}}}$ is equal to $\mid \mathcal{T} \mid$. The i^{th} element of $\mathbf{X_{f,\mathcal{T}}}$, i.e., $X_{f,\mathcal{T}}(i)$ is equal to the number of times the i^{th} token is present in the path of f. Section 2.2 details the tokenization strategy that addresses concatenation of entities, something quite common in enterprise metadata.
3. **Extension features.** In order to understand users' affinities towards certain types of files, we record the file extension as a categorical feature. To utilize this in our models, we construct a vocabulary based on popular file extensions in the share, $\mathcal{E}$. We then represent the extension of a file f by a binary vector $\mathbf{X_{f,\mathcal{E}}}$ where the i^{th} value of $\mathbf{X_{f,\mathcal{E}}}$, i.e., $X_{f,\mathcal{E}}(i)$ is 1 if f has the i^{th} extension, and is 0 otherwise. Cardinality of $\mathbf{X_{f,\mathcal{E}}}$ is equal to $\mid \mathcal{E} \mid$.

For a file f, the metadata feature vector $\mathbf{X_f}$ is obtained by concatenating the above three types of features. Specifically, for file f, the metadata feature vector is obtained as

$$\mathbf{X_f} = [\mathbf{X_{f,\mathcal{F}}} \quad , \quad \mathbf{X_{f,\mathcal{T}}} \quad , \quad \mathbf{X_{f,\mathcal{E}}}]. \tag{3}$$

4 MODELING

In order to model users' file access patterns, we define a training period to train the models, and a testing period to evaluate. We follow a personalized modeling approach where we train one model for each user. For evaluation, we select 30 users based on the number of file activities of all users in a share. Details on the selection of evaluation users are provided in Section 5.1. All files that were operated on during the training period by at least one user in the share are the training instances. For user u, the training label of a file is 1 if u accessed the file during training period, 0 otherwise. Testing instances are the files that are operated upon in testing period, after removing the files that were observed in the training period. This ensures that the testing files are new relative to the training files. The testing labels are determined in the same fashion as for training. Note that we only focus on training and testing over file read events. The total set of training instances is the same across all users, but the labels can differ. The same holds for testing. The overall approach is described in Figure 1(a). We approach the modeling of file access patterns and its evaluation as a classification problem and utilize the features detailed in Section 3.

4.1 Collaborative filtering aware modeling

As discussed in Section 2, file accesses typically demonstrate a high degree of collaboration among users as evidenced by the triangle count. The features defined in Section 3 only capture metadata attributes of files. The models trained on these features can be improved by utilizing the predictions from models of other users in the same share. With this motivation, we describe how the system augments the per-

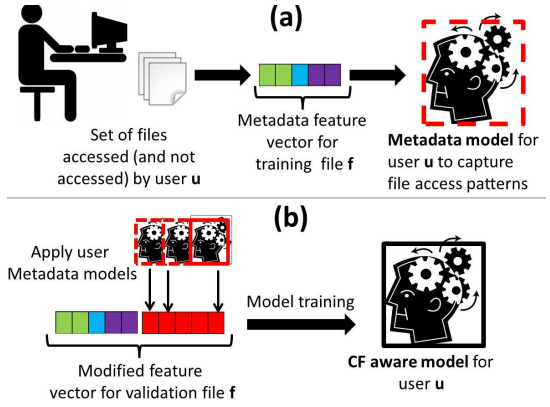

Figure 1: Overall approach of the proposed system. a) shows the training of user models based on only metadata features (metadata models). The files accessed by users are represented in terms of their metadata features, followed by training of the metadata model. b) shows how individual metadata user models are applied on validation files to train collaborative filtering (CF) aware models (Section 4.1).

sonalized user models with additional information to achieve collaborative filtering.

Figure 1(b) shows the modified approach to make the trained models aware of the collaboration among users. We obtain validation instances from the training instances. There are several ways to do so. We experimented with sampling validation instances from training instances for different sampling rates. The best performance, however, was observed when the validation set was kept the same as the training set. On the other hand, the testing set, as required, is completely independent of the training or validation set.

Let U represent the set of users in a share. The models labeled with "Metadata models" in Figure 1 represent the $\mid U \mid$ personalized classification models based on the metadata features. Each of the metadata models is trained over the training instances, and applied on the validation instances. For a file f among the validation instances, the predicted labels from metadata models are concatenated to form $\mathbf{P_{f,U}}$ where the j^{th} value i.e., $P_{f,U}(j)$ is equal to the predicted label of the validation instance f by the metadata model for the j^{th} user in the share. These predicted labels are concatenated with the metadata feature vector of the validation instances to form the feature vector for a second layer of models. Specifically,

$$\mathbf{X_f^c} = [\mathbf{X_f} \quad , \quad \mathbf{P_{f,U}}], \tag{4}$$

and the second layer of models are binary classification models trained with $\mathbf{X_f^c}$ as the feature vector for each validation file f. These collaborative filtering aware models are represented as "CF aware" models in Figure 1(b).

During the testing phase, the predicted label for a given user u on a test file f is obtained as follows. First the metadata models of all the users in U are applied on f to obtain their predicted labels. These labels are then concatenated with the metadata features of f as shown in Eq. 4. Finally, the predicted label for a user is obtained by applying her CF aware model on the concatenated feature vector of f.

Constructing CF aware user models based on the above approach has two key advantages. First, CF aware models can leverage collaboration by factoring the predicted labels of other users' metadata models. It should be noted that this approach does not require explicitly defining a similarity metric between users or their access patterns, and yet enables the model to improve its predictive performance. For example, if the users u_1 and u_2 have similar behavior, it can be expected that the validation files for which the metadata model of u_1 predicts a positive label, are also likely to be accessed by u_2. Now consider a third user u_3 whose behavior is very different from that of u_1 and u_2. Thus, if the metadata model of u_3 predicts a positive label for a validation file f, the likelihood of the user u_1 or u_2 accessing f would automatically decrease. The CF model can leverage such learned knowledge of similar and dissimilar access patterns to improve its correctness.

The second advantage of our CF aware modeling is that it does not suffer from the cold start problem that the traditional collaborative filtering systems suffer from. To make recommendations to the user u, these systems first identify other users who share similar preferences with u, and then propose items which were favored by the other users but not seen by u. Such systems fail to make recommendation for a completely new item. Our approach gets around the cold start problem by utilizing the predicted labels of users in a share along with the metadata features. In Section 5.6, we show how the CF aware models improve the classification performance over the metadata models. With a higher degree of collaboration, we are expected to observe higher gains of the CF aware model. Our results strongly corroborate this observation.

5 EVALUATION

In this section, we describe the evaluation procedure for our modeling approach. In particular, we provide performance results over eight shares (network file servers) with varying time duration and separation between the training and the testing periods. We first describe the procedure for selecting a subset of users for our experiments.

5.1 Selecting users for evaluation

A file recommendation system is useful for active users only. Therefore, we select a subset of users in a share based on the number of their file activities. We rank the users in a share in the increasing order of the number their file activities. For the purpose of evaluation, we then randomly sample 30 users from those users whose activity numbers are above the third quartile. In an actual deployment, all the active users must be considered.

5.2 Evaluation metrics

For a user u and a testing file f, the true label is 1 if u accessed f in the testing period, and 0 otherwise. The model for u is used to predict the label of the testing file. Let $F_{true+,u}$ be the set of test files that are actually accessed by u in the testing period, and $F_{pred+,u}$ be the set of test files that are predicted to be accessed by u. We use precision and recall, which are commonly used metrics to evaluate classification tasks. For our problem, the precision and recall are $\frac{|F_{true+,u} \cap F_{pred+,u}|}{F_{pred+,u}}$ and $\frac{|F_{true+,u} \cap F_{pred+,u}|}{F_{true+,u}}$ respectively. For a file recommendation type of an application, while having high recall is definitely useful, having high precision is essential for usability. If most of the recommendations (i.e., $F_{pred+,u}$) are wrong, the end user will simply ignore the recommendations. Considering this, we use the following metrics for the evaluation.

- **F-score.** F-score is calculated as the harmonic mean of precision and recall and thus provides a balanced picture of the overall predictions. We use the F-score averaged across the evaluation users (AF) as one of the metrics to discuss the results.
- **Recall@75P.** We also evaluate the file access modeling by keeping the precision fixed to a high value. Using the confidence score for each prediction, we reduce the number of positive predictions until the precision is 75%, and use the recall at this precision as a performance metric. Thus recall at 75% precision provides the fraction of a user's actual file accesses that a model was able to correctly predict while ensuring that only 25% of the model's positive predictions are not shown in the user's activities. AR@75P is its averaged value across all the evaluation users.

5.3 Varying training and testing periods

We perform evaluation for several combinations of training and testing periods, with varying time duration and separation between them. For convenience, we divide the entire duration of each share into 7 equal size slices as shown in Figure 2. For our evaluation, we experiment with varying lengths of training periods. For this, we fix the last 2 slices for testing, and use the first 5 slices for training, while ensuring that the testing period starts right after the training period (See Figure 2(a)). Similarly, we also experiment with varying lengths of testing periods as shown in Figure 2(b).

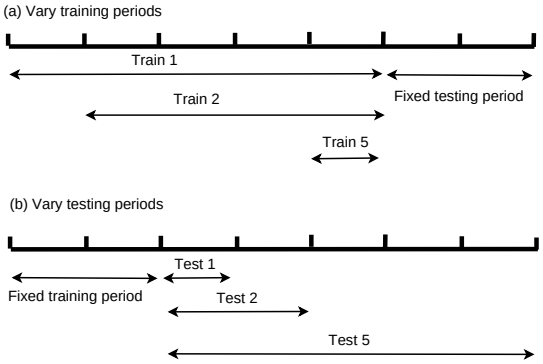

Figure 2: Splitting dataset into various training and testing periods

5.4 Selecting classification model

We experimented with different classification models. Table 4 compares the performance of a few models using the metadata features for share A. It provides model effectiveness in terms of Avg AF score, which is the average of F-score across 30 evaluation users, and across different training and testing periods, as outlined in Section 5.3. The regularization coefficient C for SVM models is obtained by logarithmic grid search over $\{10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$. The gamma parameter for polynomial kernel SVMs is obtained in a similar manner. The best parameters as obtained by three fold cross-validation are finally used for training. Also, L2 regularization is used.

The table also provides the total model training time across all training periods and evaluation users. The time is measured as real time on a 32-core, 64GB, and 2.6GHz machine. The system is implemented using scikit-learn [scikit-learn, 2015], a Python library for machine learning. Our implementation uses multiprocessing to speed-up the overall training.

Based on the results, we pick Linear SVM for our modeling because it provides the best trade-off between effectiveness and training time. Moreover, Linear SVM also provides learned feature weights, which are very useful for understanding the significance of features (see Section 6).

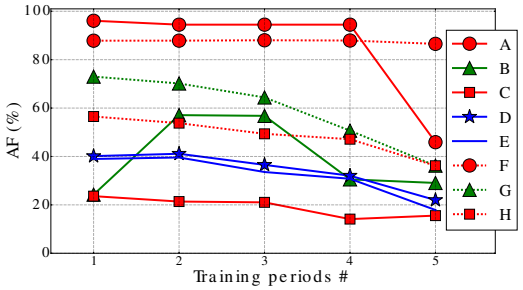

Figure 3: AF for the metadata models with the fixed testing and varying training periods

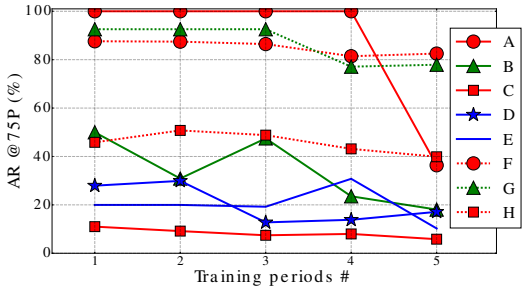

Figure 4: AR@75P for the metadata models with the fixed testing and varying training periods

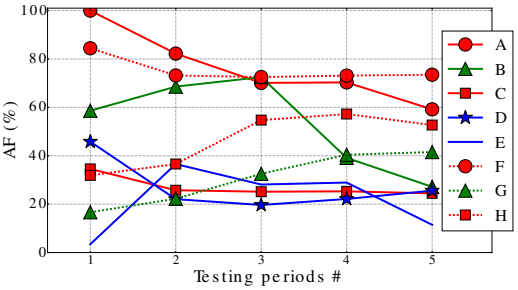

Figure 5: AF for the metadata models with the fixed training and varying test periods

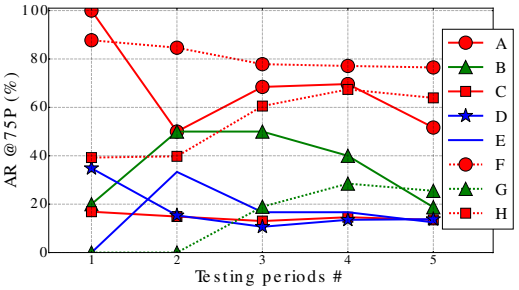

Figure 6: AR@75P for the metadata models with the fixed training and varying test periods

Metric	Linear SVM	Polynomial SVM degree 2	Multinomial Naive Bayes	Decision Tree
Avg AF	80.7	83.0	25.0	77.6
Train time (mins)	31	236	6.2	78

Table 4: Performance comparison between different machine learning algorithms

5.5 Metadata modeling

Figures 3 and 4 respectively show how AF and AR@75P for metadata models are affected by varying length of training periods. The longest training period, the one corresponding to index 1, leads to the best performance for most shares. We observe moderate degradation in performance as the training window shrinks. This suggests that we need a sufficiently long training window for better performance. It should be noted that the variations are observed over different durations of the datasets, ranging from from 57 to 123 days (Table 1). It can be expected that as the window of the training period gets longer, after a point the model performance would decrease. This is because the model may give more importance to access patterns that are outdated with respect to the content that the user is recently accessing. We do not provide a recommended or optimal training period because that would depend on several factors including the number of users, their workload, and the rate of change of access patterns. Nonetheless, in Section 6.6, we discuss the potential to learn such type of configuration parameters based on online model evaluation.

Figures 5 and 6 show similar results but for different testing periods. These results show that beyond testing periods with indices 1 and 2, which are small (Table 2) and thus potentially noisy, the model performance drops mildly as the length of separation between training and testing periods increases. The mild drop shows sustaining ability of our models.

Table 5 summarizes the metadata model results across all the above combinations using the average and the maximum values of AF and AR@75P. AF averaged across all training and test periods is provided as Avg AF. Max AF shows the best AF achieved which is an indicative of the realistic performance of a properly tuned file recommender system. The high AF and AR@75P values seen for most shares demonstrate practicality of our metadata models. The substantial variation in the performance across different shares reflects differences in their characteristics such as rate of activity, rate of change in user preferences, and collaboration.

The last column of Table 5 shows that most of the

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
B	**46.4**±0.0	**51.2**±0.0	**34.9**±0.0	**44.2**±0.0	73.1
C	**23.1**±0.0	**24.2**±0.0	**11.4**±0.0	**12.4**±0.0	87.7
D	**30.7**±0.0	**36.3**±0.0	**19.0**±0.0	**24.7**±0.0	82.8
E	**26.9**±0.0	**35.0**±0.0	**17.9**±0.0	**24.1**±0.0	66.0
F	**81.5**±0.0	**84.3**±0.0	**82.9**±0.0	**84.6**±0.0	9.8
G	**44.8**±0.0	**51.3**±0.0	**50.6**±0.0	**58.3**±0.0	99.9
H	**47.6**±0.0	**50.2**±0.0	**49.9**±0.0	**53.0**±0.0	74.4

Table 5: Performance summary of the 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	**100.0**±0.0	**78.5**±0.0	**100.0**±0.0	21.2
B	**48.6**±0.0	**77.1**±0.0	**32.2**±0.6	**62.1**±0.0	73.1
C	**23.5**±0.0	**36.0**±0.0	**10.1**±0.0	**16.0**±0.0	87.5
D	**32.3**±0.0	**47.5**±0.0	**21.5**±0.0	**38.3**±0.0	83.3
E	**27.6**±0.0	**41.1**±0.0	**25.3**±0.0	**100.0**±0.0	65.7
F	**81.5**±0.0	**87.9**±0.0	**83.3**±0.0	**89.2**±0.0	9.8
G	**55.5**±0.1	**76.2**±0.0	**58.0**±0.2	**89.9**±0.4	96.6
H	**47.6**±0.0	**57.2**±0.0	**49.6**±0.0	**66.5**±0.0	75.4

Table 6: Performance summary of the CF aware 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.

correctly recommended files to a user were *not* created by that user in the testing period. This is good because recommending a newly created file to the user, who created it, is obviously futile.

5.6 Collaborative filtering aware modeling

Table 6 provides a summary of the performance of our CF aware models. On comparing Table 6 with Table 5, we observe that CF aware models provide substantial performance improvement over metadata models for most shares. As discussed in Section 4.1, the CF aware models are expected to benefit shares with high amount of collaboration between users. This is confirmed from the fact that shares B, D, E, G which show the most improvement are amongst the top five shares in terms of normalized triangle counts (Table 1). Although share A is also among the top five shares, the performance of its metadata models is already too high to show significant improvement.

In the next section, we discuss the contribution of different types of features to the performance results.

Feature type	Percentage of feature type in top 10 model features	Total number of features of the type
Folder	10.4%	82.0%
Token	47.9%	14.8%
Extension	6.3%	0.1%
User	35.4%	3.1%

Table 7: Analysis of features with respect to feature types. These numbers are obtained by aggregating the weights per feature in collaborative filtering aware user models for different training periods as described in Figure 2 and for the eight shares used for evaluation.

We also discuss scalability issues, and considerations for real world deployment.

6 DISCUSSION

Given the features in our model, which were most significant in the trained models? In order to perform this analysis, for each user, we train a Linear SVM for CF model (Section 4.1) based on each of the six training periods as outlined in Section 5.3. The significance of a feature with respect to a trained user model can be obtained based on the absolute weight given to the feature in the model. For each user model, we select the top ten most significant features. Table 7 shows the proportions of different features among top features per user model, aggregated across different evaluation users, shares, and training periods. We provide insights about file user activity based on how the models leveraged each feature type below.

6.1 Folder feature analysis

Despite the fact that folder features accounted for more than 80% of the feature space, only 10.4% of top features were drawn from this category by the personalized models. Folders within three levels of the root account for more than 80% of top ten folder features. This makes intuitive sense because folders that are farther from the root are intrinsically sparser, and our models apply regularization, which discourages applying significant weights to sparser features when more frequent and predictive features are present. Despite their proximity to the root, these "shallow" folders still wield significant predictive power. Interestingly, we found that no file in our test set was immediately descendant of a folder feature (i.e. a folder distance of zero) in the top ten folder features. Files in the test set were at least one folder away from the folders constituting the folder features. In fact, the data shows that more than a third of the files active in testing period were 3 or more folders below a folder

feature in the top ten features. Despite the distance, the ancestral folder still provides quite a bit of predictive value.

6.2 Token features

To preserve the privacy of individuals, groups and organizations, we can only discuss trends observed in the tokens that were highly influential in classification. We noticed that tokens were drawn from the applications used to generate the data. The tokens would either refer to the application name or the application generated unique prefixes or suffixes in the folder or file paths. Additionally, paths contained the names of groups for this organization. This would likely help members of that group identify which subtrees of the file system hierarchy contained data integral to their role. Lastly, tokens referring to the month, year and content type were important in many models. As expected, the timestamps of file activity aligned with the month and year referred to in the path. This circumstantially corroborates the importance of temporal information in our model.

6.3 Extension features

Analyzing the sign of the weights for extension features yields an interesting observation: more than 85% of the weights of the extension feature in the top ten features were negative. Extensions can yield insight into the type of content and/or application that generated it (which may be used to infer the role or function of users). Since weights for this feature were predominantly negative, this indicates it would be quite unlikely that the user would use the application that generated this file, which could imply something about the nature of their role, e.g. what it is not. It is possible that for the model, which applies regularization, finding extensions that are strongly anticorrelated with the user activity served as a strong indicator and would substantially contribute to minimizing the penalty attributed to the model. This would explain why even though this feature category accounts for 0.1% of all features, it still accounted for 6.3% of the top ten features.

6.4 User feature analysis

Providing to the model the likelihood that another user will access this files allows the classification model to achieve collaborative filtering by accounting for other users' preferences. What is noteworthy is that 35.4% of the top ten features for the personalized models are the probabilities of a user accessing this file, which account for 3.1% of total features. The weights for user features where the user is not the same as the user for whom the personalized model is being built were negative 32% of the time and positive 68% of the time. For users where the weight of another user's file access likelihood is negative serves as an interesting signal that these two users have different file access patterns and we should not expect them to have a significant intersection of files accessed in common. On the other hand, 68% of the time the weight of the user feature was positive, indicating that the user for whom the model is trained and this other user have a significant interest in the same type of files. Interestingly, particular users appeared as top ten features for many different user models in the same share. In fact, we observed that the same user appeared as a top ten feature in five of the thirty models 20% of the time and the same user appeared as a top ten feature for ten of the thirty models 9% of the time. This suggests that perhaps there are particular users whose file access patterns serve as exemplars for how users access resources.

6.5 Scalability

As files are generated or modified on a share, our system needs to apply personalized model of each user to make the predictions. Therefore, a high rate of file operations, or a high number of users will both adversely affect the scalability our system. We can address these factors to optimize the testing time as discussed in Section 8. Moreover, a recommendation system may not be essential for all the file servers. For instance, it would not make much sense to deploy our system on a networked home directory or on a backup server. Additionally, it may not make much sense to provide file recommendation to low-volume file users, but rather focus on enterprise search when they do need to find information. It may be prudent to train a model for only the users that are determined to be sufficiently active in a share. Furthermore, for shares that do not demonstrate high degree of user collaboration, training and using CF aware models may not be recommended since they are computationally much more expensive than metadata based models.

6.6 Considerations for real world deployment

In this section, we discuss the considerations for deploying the proposed system in an actual enterprise environment.

In this paper, we validate our system against file activities that occurred in the past. Whereas for an

actual deployment, the system can be evaluated in an online manner. This will help in monitoring workload characteristics, and measuring the effectiveness more accurately, and in a continuous manner. This can be used as a feedback to tune the models and make them adaptive. For instance, parameters such as the length of the training window and frequency of retraining can be tuned based on the feedback.

The precision, as reported by our evaluation, provides a lower bound to the precision that may be observed in an actual deployment. To understand this, consider that a user is recommended a file that he/she was not aware of, and the user ends up accessing the file. This would be a case of true positive, whereas our current evaluation would show this as a false positive.

Lastly, since a recommendation system can interact with users, it may be possible to obtain subjective evaluations of the recommendations such as recommendation quality.

7 RELATED WORK

Prior works on modeling file access patterns have been mostly focused on performance enhancement of storage systems, e.g., reducing I/O latency by prefetching [Amer et al., 2002, Xia et al., 2008, Kroeger and Long, 2001, Yeh et al., 2002, Yeh et al., 2001a, Yeh et al., 2001b, Whittle et al., 2003, Paris et al., 2003]. These systems make predictions for only existing files, whereas our approach can make predictions for newly created files too. However, our focus is on recommendation rather than caching.

The approach by Song et al. [Song et al., 2014] is closer to our work since it aims at assisting knowledge workers by recommending files and actions. It uses a data mining technique to first group similar files into abstract tasks, and then mines frequent sequences of abstract tasks into workflows. It then makes recommendations by identifying the workflow that best matches the current file usage pattern of the user. While the technique attempts to generalize beyond exact file matching, it cannot provide recommendations for new files. In contrast, we train personalized machine learning models that provide recommendations even for new files. For evaluation of our approach, we use only those test files that are new with respect to the training files. The ability to recommend new content is important in order to connect knowledge workers with new data which is being generated at a tremendous rate [Gantz and Reinsel, 2012].

Unlike all the previous works, we use much richer file metadata including filename, path, file system hierarchy, extensions and collaborative filtering in our models. As a result of this, our work can also supplement existing Data Governance systems with predictive capabilities. While our approach is not ideal for file caching in performance sensitive applications, it could be effective in cloud services to reduce network latency by caching files on client-facing web servers or directly on clients. It could also be useful for scenarios with intermittent connectivity, such as choosing files to cache on mobile devices.

The personalized model based file recommendation as proposed in our paper is a content-based recommendation system. As compared to traditional collaborative filtering based recommender systems [Linden et al., 2003, Breese et al., 1998], our approach does not suffer from cold start problem, i.e., inability to recommend a new item (file). It should however be noted that unlike traditional collaborative filtering techniques, we do not use actual access information. Rather, we predict the access likelihood of a user for a particular test item and combine it with metadata features. This enables us to circumvent the cold start problem, and thus benefit from collaborative filtering.

Finally, advanced machine learning models such as factorization machines [Rendle, 2010], deep neural networks [Salakhutdinov et al., 2007, Hinton et al., 2006] and topic models [Nagori and Aghila, 2011, Ovsjanikov and Chen, 2010] can also be employed for modeling file access predictions. To a large extent, these techniques are complimentary and can contribute in making our models more effective. Notwithstanding, we approach the problem as a classification problem and show reasonable effectiveness even with a simple Linear SVM-based model. Our focus is more on the domain specific application, with the goal of extracting meaningful features from file metadata and user activities.

8 FUTURE WORK

There are several directions in which the proposed system can be extended to improve both its efficiency and efficacy, and to make it applicable to new and emerging scenarios.

Optimization techniques can be developed that can make the model testing much faster, by judiciously selecting the user models that need to be applied on a new file. Such techniques may be able to trade off model correctness for testing speed in some scenarios.

The metadata features show a high degree of spar-

sity as a result of how they are constructed. A file typically has very few keywords in its path, and thus most of its token features would be zero. Similarly, very few of its folder features, and at the maximum of one extension feature of a file are non-zero. While sparsity can be helpful for training user models [Ngiam et al., 2011], the large dimensionality of data may negatively affect the performance of the models. It is possible that the correctness and speed of the proposed system can be further improved by capturing the interdependence between different features through dimensionality reduction techniques such as Principal Component Analysis [Jolliffe, 2005][Van der Maaten et al., 2009]. For example the folder features demonstrate substantial interdependence and redundancy and techniques to transform them to a suitable space may be explored.

Modeling the file metadata and user features in context of temporal nature of file accesses could also be a potential direction for further work. For example, giving more importance to recent events while training user models may accommodate shifts in user interests, leading to improved performance. Identifying and modeling repetitive activity may also be informative since users may be interested in similar tasks after fixed time intervals, such as on the same day each week. As mentioned in Section 6.6, deployment of the proposed system in an enterprise environment offers an online framework to evaluate the trained models. Online model training or update techniques can be developed that utilize the model evaluation information to improve the trained models by adapting them to new access patterns or newly observed features. For example, consider a scenario where a trained model is seen to perform poorly because most of the recent activity for a user is confined to a recently created folder that was not part of the folder features in the trained model. Such information can be derived from the online evaluation and can be used to update features of the trained model and to adapt the model to reflect the updated access patterns.

In addition to training personalized user models, insight into directed preferences of users may be useful for recommending content. For example if it is determined that user u_1 often accesses documents created by user u_2, then a recent modification by u_2 may be useful information for u_1 and can be used as an indicator to recommend relevant content.

Lastly, file access prediction offers interesting possibilities for applications such as information security, by offering new measures of access improbability.

9 CONCLUSION

This paper presents a system that provides file recommendation to assist knowledge workers process increasing volumes of data. The system utilizes natural language processing to derive usable information from file metadata, and machine learning to train personalized user models that have good predictive value, even for files that have not been observed in the past. Through extensive experiments on real world data we demonstrate the feasibility of the system to offer high quality recommendations, which is reflected particularly in the significant recall at high precision across eight shares. We also show that for shares exhibiting a high degree of collaboration between its users, the predictions from different user models can be combined to improve the performance of an individual user's model. It is observed that the trained models have a high temporal longevity, and experience moderate performance degradation for short training periods. Since the system requires training personalized models for each user under consideration, it should be applied only on shares and users that display sufficient activity and are determined to be of interest.

REFERENCES

[Active Directory, 2015] Active Directory (2015). Active directory. http://msdn.microsoft.com/en-us/library/bb742424.aspx.

[Amer et al., 2002] Amer, A., Long, D. D. E., Paris, J.-F., and Burns, R. C. (2002). File access prediction with adjustable accuracy. In *International Performance Conference on Computers and Communication (IPCCC)*.

[Breese et al., 1998] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Conference on Uncertainty in artificial intelligence*.

[Bybee, 1985] Bybee, J. L. (1985). *Morphology: A study of the relation between meaning and form*, volume 9. John Benjamins Publishing.

[CamelCase, 2015] CamelCase (2015). Capitalization styles. http://msdn.microsoft.com/en-us/library/x2dbyw72\%28v=vs.71\%29.aspx.

[coca, 2008] coca (2008). The corpus of contemporary american english: 450 million words, 1990-present. Available online at http://corpus.byu.edu/coca/.

[Gantz and Reinsel, 2012] Gantz, J. and Reinsel, D. (2012). The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. In *IDC iView: IDC Analyze the Future*.

[Hinton et al., 2006] Hinton, G., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[IDG Enterprise, 2014] IDG Enterprise (2014). Big data survey.

[Jolliffe, 2005] Jolliffe, I. (2005). *Principal component analysis*. Wiley Online Library.

[Kroeger and Long, 2001] Kroeger, T. and Long, D. D. E. (2001). Design and implementation of a predictive file prefetching algorithm. In *USENIX Annual Technical Conference*, pages 105–118.

[Leonardi et al., 2013] Leonardi, P. M., Huysman, M., and Steinfield, C. (2013). Enterprise social media: Definition, history, and prospects for the study of social technologies in organizations. In *Journal of Computer-Mediated Communication*.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing*, 7(1):76–80.

[Nagori and Aghila, 2011] Nagori, R. and Aghila, G. (2011). LDA based integrated document recommendation model for e-learning systems. In *International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*.

[Ngiam et al., 2011] Ngiam, J., Chen, Z., Bhaskar, S. A., Koh, P. W., and Ng, A. Y. (2011). Sparse filtering. In *Advances in Neural Information Processing Systems*, pages 1125–1133.

[Office365, 2015] Office365 (2015). Microsoft office 365. `http://en.wikipedia.org/wiki/Office_365`.

[Ovsjanikov and Chen, 2010] Ovsjanikov, M. and Chen, Y. (2010). Topic modeling for personalized recommendation of volatile items. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

[Paris et al., 2003] Paris, J.-F., Amer, A., and Long, D. D. E. (2003). A stochastic approach to file access prediction. In *International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI)*.

[Rendle, 2010] Rendle, S. (2010). Factorization machines. In *IEEE International Conference on Data Mining (ICDM)*.

[Salakhutdinov et al., 2007] Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *ACM International Conference on Machine Learning*.

[Salesforce, 2015] Salesforce (2015). Salesforce.com. `http://www.salesforce.com/`.

[scikit-learn, 2015] scikit-learn (2015). scikit-learn Machine Learning in Python. `http://scikit-learn.org/`.

[Song et al., 2014] Song, Q., Kawabata, T., Ito, F., Watanabe, Y., and Yokota, H. (2014). File and task abstraction in task workflow patterns for file recommendation using file-access log. In *IEICE Transactions on Information and Systems*.

[Van der Maaten et al., 2009] Van der Maaten, L. J., Postma, E. O., and van den Herik, H. J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(1-41):66–71.

[Wang et al., 2011] Wang, C., Viswanathan, K., Choudur, L., Talwar, V., Satterfield, W., and Schwan, K. (2011). Statistical techniques for online anomaly detection in data centers. In *IFIP/IEEE International Symposium on Integrated Network Management*, pages 385–392.

[Whittle et al., 2003] Whittle, G. A. S., Paris, J.-F., Amer, A., Long, D. D. E., and Burns, R. (2003). Using multiple predictors to improve the accuracy of file access predictions. In *International Conference on Massive Storage Systems and Technology (MSST)*, pages 230–240.

[Xia et al., 2008] Xia, P., Feng, D., Jiang, H., Tian, L., Xia, P., Feng, D., Jiang, H., Tian, L., and Wang, F. (2008). Farmer: A novel approach to file access correlation mining and evaluation reference model for optimizing peta-scale file systems performance. In *The International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC)*.

[Yeh et al., 2001a] Yeh, T., Long, D. D. E., and Brandt, S. A. (2001a). Performing file prediction with a program-based successor model. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*.

[Yeh et al., 2001b] Yeh, T., Long, D. D. E., and Brandt, S. A. (2001b). Using program and user information to improve file prediction performance. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*.

[Yeh et al., 2002] Yeh, T., Long, D. D. E., and Brandt, S. A. (2002). Increasing predictive accuracy by prefetching multiple program and user specific files. In *Annual International Symposium on High Performance Computing Systems and Application (HPCS)*.