The Next Generation's Personal File System Management

Iru Wang IRUWANG@STANFORD.EDU
Xuan Yang XUANY@STANFORD.EDU
Yi-Chen Tsai YTSAI@STANFORD.EDU

Abstract

The current file systems are hierarchical, which can cause duplicate storage and cannot represent human's mind map. In this paper, we explore the possibility of a heuristic, relational personal file system. Regarding each file as a node in the graph, we implement K-means, EM, LDA and Tree Bagging algorithms respectively to group the related files. In this way, we convert the current hierarchical file system to relational file system. We compare the results of these algorithms, the error of K-means, EM, LDA and random forest are 62.33%, 61.4%, 42.33% and 16% respectively. Among all the unsupervised learning, LDA - a popular and generative model in topic modeling - gives the best accuracy, but still does not surpass supervised learning. Therefore we propose to combine LDA and Tree Bagging algorithm, using the semi-supervised learning to classify the files in the future. In the end, we also discussed the potential of combining latent fator model with above methods to classify a large scale of file sets, thus extending our method from personal file system to corporate file system.

1. Introduction

Text mining is an important application in the fields of machine learning and natural language processing. Various topic modeling techniques have wide spread use in text mining applications. In this paper, we implement these techniques onto personal files classifications. We attempt to gain an insight on how a personal file system should cluster and organize files. A bag-of-words model is implemented, in which word order is ignored(Wallach, 2006). The iterative optimization clustering algorithms have been demonstrated reasonable performance for clustering(Nathiya & Punitha, 2010). A simple method would be K-means clustering, which is preferred in many large-scale applications(Brian Kulis, 2012). However, since our matrices are sparse, the error we get is 62.33%. The Expectation Maximization (EM) algorithm is also common and effectual when clustering data, we improve our result to an error of 61.4%. Latent Dirichlet Allocation(LDA) has become a standard tool in document analysis(D. M. Blei & Jordan, 2003). We describe the implementation of this algorithm and get its result to be 42.33%, better than EM and K-means. However, the error rate of LDA is still not satisfying, so we implement a popular supervised-learning method in classification problems, which is Tree-structured classification, using bootstrap aggregating to improve the accuracy. We then compare the result of above three algorithms with the result of Tree Bagging, whose performance is the best, with optimal error rate to be 16% . The detailed implmentation and analysis of above algorithms are in section 5. According to the above results, it is better to use random forest classifier for the personal file system. However, since the time complexity of random forest is relatively high, the classifier will be very inefficient if the filesystem is very large. In order to solve the issue with a large scale of files, we can adopt factor analysis, so it will be unneccessary to parse all the files and build the dense matrix, thus the algorithm can still be running at an accepted speed with only a little accurcy sacrificed.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

2. Motivation

The current hierarchy file system cannot accurately classify personal files. Each file may fit into many hierarchical locations, for example, a school project file may belong to school/ as well as project/, or even school/project/, causing misleading search and redundant duplicates. Furthermore, a hierarchical file system does not represent how human think and interact with files. Here we explore the possibility of a heuristic, relational personal file system. By exploring the algorithms in text minig field, a novel file system can be created to help human classfiy the files automatically. And each time user create a new file or download a new file, the file system can group this file to the most suitable group according to personal perference for user to access next time. In addtion to the above features, the file system can also help user to manage and clean up the messy desktop and imporve user's working efficiency on computer.

The personal file system can furthur be utilized by companies to manage all files on the server.

3. Related Work

A similar problem with the file system classfier is spam email problem, which has been studied by many people for decades. There are many effective methods proposed to filter junk emails, one study is based on a bayesian approach. By casting the problem in a decison framework, they make use of probabilistic learning methods in conjunction with a notion of differential misclassification cost to produce filters, considering domain-specific features in addition to using Gibbs EM algorithm on raw text of email messages(M. Sahami & E.Horvitz, 1998). In the topic modeling field, one study is based on a hierarchical generative probabilistic model that incorporates both n-gram statistics and latent topic variables by extending a unigram topic model to include properties of a hierachical Dirichlet bigram language model(Wallach, 2006). We found there are many studies about file system classifier or the way to change from a hierarchical file system to a relational file system. One study about file classification is predicting the properties of new files as they are created, by exploiting the strong associations between a files' properties, names, and attributes assigned to them(M. Mesnier & Seltzer, 2004). However, it can not help user manage the file system automatically based on their relations.

4. The Dataset

4.1. Inital Date Set

Throughout history of topic modeling, a lot of data sets have been used for testing documents, such as The 20 Newsgroups dataset. However, due to our focus on personal file system and the variety of each person's preference, we decided to test on documents that solely belong to one person.

A user may have all kinds of files, including music, picture, text, etc. As a starting point, we decided to focus on classifying text-based files. We extract plain texts from Microsoft Word documents, PDF documents, Microsoft Outlook emails by some tools(too, c)(too, a)(too, b). There are 215 text files in our initial dataset. The dataset was labelled by the owner before we applied the classifier to the dataset. In the end, we compare the tag defined by user with different algorithms' results to find each algorithm's error rate.

4.2. Parsing Data Set

First, we use online Ngram tool to parse files and get all the words and its frequency. Second, we compare words with a list of tokens, which serves as features for files. Here, we have chosen our tokens to include only the medium frequency word stems, assuming that words that occur too often or too rarely do not have much classication value. Examples of very frequent tokens might be 'the', 'and', which may appear so commonly that they are not indicative. Words were stemmed using a standard stemming algorithm, which means that 'house', 'houses', 'housing' may be recognized by the same token: 'hous', being treated as the same word.

Using this method, we construct a matrix with file id as row id, token id as column id, and Matrix[file id][token id] represents the number of occurance of the correspoding token in that file.

4.3. Spliting Dataset

For unsupervised learning, such as K-means, EM and LDA, it is unnecessary to split data into training set and testing set, simply have all text files serve as training set. For supervised learning, such as Tree Bagging, we randomly choose 90% of text files as training set, and the others as testing set.

For the latent factor analysis to be adopted to corporate filesystem, we need a large scale of dataset, in which case we would use 20 Newsgroups, splitting data into training and testing sets, then perform hold-out validation on them.

5. The Details of Learning and Results

5.1. K-means clustering algorithm

k-means clustering is a method of vector quantization originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster(k-m, b).

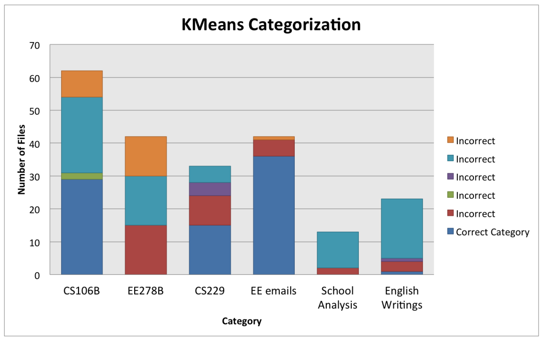

Figure 1. KMeans Categorization Results

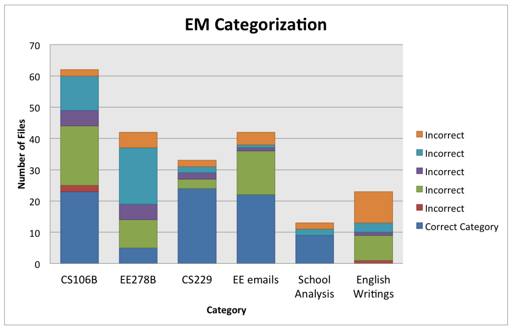

Figure 2. EM Categorization Results

The K-means clustering algorithm is as follows:(k-m, a)

1. Initialize cluster centroids $\mu_1, \mu_2, ...\mu_k \in R^n randomly.$

2. Repeat until convergence: {

For every *i, set*

$$c^{(i)} := arg_j min||x^(i) - \mu_j||^2 \quad (1)$$

For every j, set

$$\mu_j := \frac{\sum_{i=1}^m 1c^(i) = jx^(i)}{\sum_{i=1}^m 1c^(i) = j} \quad (2)$$

}

In the algorithm above, we set k to be 6, which is the number of group we want to classifiy to. We use greedy algorithm to find the most likely label matching, assuming that the bigger the quantity this label matches that group, the higher the likelihood that this is the correct group for that label. Due to our matrices being sparse, most of the files are categorized into the same group. Therefore, the error rate we get is as high as 62.33%.

In the *Figure 1*, we see that the correct categorized files account for only about 40% of the total files. For each category, there are a certain amount of files that is miscategorized to other categories. We can see that the category for EE emails has the highest discrimination rate of 85.7%, which indicates that it is the easiest category to be discriminated.

5.2. Expectation Maximization (EM) algorithm

EM algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step(EM-).

We partition the data into 6 groups and use the same greedy method to compare them with how the user label them. We improved our result to an error rate of 61.4%. It is still very high due to the fact that our matrices are sparse, and each label may have different meanings between the user and the machine, causing miscategorization.

In *Figure 2*, we see that the correct categorized files also account for only about 40% of the total files. We can see that in this case, the category for CS229 has the highest discrimination rate of 72.7%, which indicates that it is easier than other categories to be discriminated.

5.3. Latent Dirichlet Allocation(LDA)

Latent Dirichlet Allocation(LDA) is a generative probabilistic model that reshapes the topic modeling community. It allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an innite mixture over an underlying set of topic probabilities.(D. M. Blei & Jordan, 2003)

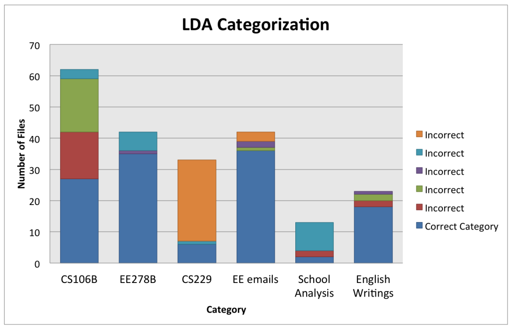

Figure 3. LDA Categorization Results

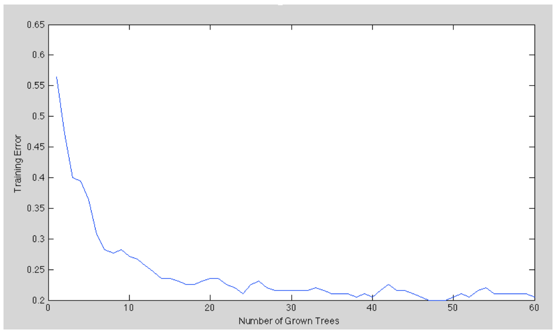

Figure 4. Tree Bagging training error

The probability of a corpus is given by:

$$p(D|\alpha,\beta) := \prod_{d=1}^{M} \int p(\theta_d|\alpha)(\sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn},\beta))d\theta_d \tag{3}$$

After running LDA, we use the highest topic probability for a file to be its category, and get the error rate as low as 42.33%, as shown in *Figure 3*. If we account for the second high topic probability to be also its category, we can reduce the error rate to 40.0%. It only decreased a little, which means that the correlation between topics may be low. We see that the most discriminated category is once again EE emails, having 85.7% discrimination rate, and the topmost indicative word for this topic is "staff", giving us an insight on what this topic is mainly about.

5.4. Tree Bootstrap Aggregating(Tree Bagging)

Bagging and boosting are general techniques for improving prediction rules. Both are examples of what Breiman (1998) refers to as perturb and combine (PC) methods, for which a classification or regression method is applied to various perturbations of the original data set, and the results are combined to obtain a single classifier or regression model. Bagging and boosting can be applied to tree-based methods to increase the accuracy of the resulting predictions(Sutton, 2005).

We randomly choose 90% of files as training set. After the decision trees learned from the features and the labels of these 195 files, we use the rest 10% files as the testing set, and compare the predicted result with the user-defined label. We ran it for different number of decision tree: 10, 20, 30 , up to 70. After running 10 times for each case, we get the lowest average error rate to be 16%. *Figure 4* and *Figure 5* are showing the relation of trainig error and testing error for different number of decison trees respectively. We can see that

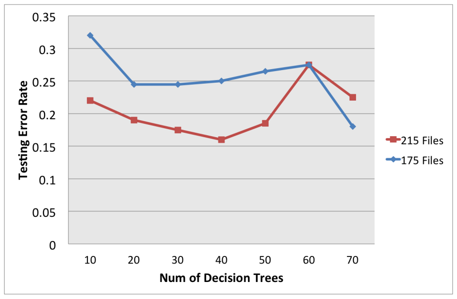

Figure 5. Tree Bagging testing error for different number of files

the training error does not change much after we have more than 25 trees, but at 40 trees, testing error gets to a local minimum. This is probably because that as the number decision trees grows to some extent, each tree loses its correctness due to the lack of nodes in the tree. In *Figure 5* we also see that as we increase the number of files in the dataset, our testing error rate may decrease largely. Therefore, it is important that we have sufficient datasets. *Figure 6* shows that testing error may depend on how many files are in training data.

By using Tree Bagging, the error rate is much tolerable. However, we need the user to label most of the files for us, which is somewhat against our goal of machine automatically categorizing files for users. Moreover, the time cost for Tree Bagging of training data with all features is large: a typical run of 40 trees using 195 training files and 1728 features may take around 400 seconds. This is a disadvantage compare to K-means, EM and LDA, which at most take tens of seconds for a run.

6. Conclusion

For unsupervised learning, the algorithm that gives us the best result is Latent Dirichlet Allocation (LDA), which has an error rate of 42.33%. For supervised

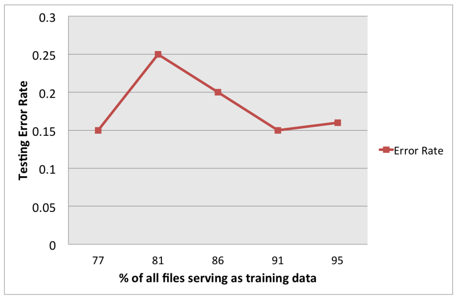

Figure 6. Error rate of different percentage of total files serving as training data

learning, Tree Bagging gives us an error as low as 16.0%, but we need the user to label most of the files for us. Also, Tree Bagging is time consuming. Therefore, it would be best if we can combine the two and use a semi-supervised method. This method can be implemented as below:

First We use the unsupervised LDA to categorize all the files, and use the topic of highest probability as the file's label.

Second As user encounters a miscategorization, we make adjustments to our file's label. We then run Tree Bagging as a background application to hide its run time.

Third When there is a new file to be categorized, we use our built tree to predict which category this file belongs to.

By doing this, we have a good starting point and also a smooth process of modifying labels as well as adding in new files.

7. Future Work

Since Tree Bagging already produces relatively good prediction accuracy, we plan to further reduce error rate, computation complexity and user label workload. There are multiple things we can do to achieve our objectives:

1. We can increase dataset by using larger number of files from one person, or we can train the model based on large number of customer's personal files, thus we can learn more from user's varied preferences and accordingly customize the classification.

2. We can develop a system combining LDA and Tree Bagging, so that user do not need to label the group for each file all the time, they only need to provide the information at the beginning of the training phase, and when the unsupervised error rate exceed certain threshold.

3. In order to reduce computation complexity, we propose to use factor analysis algorithm to learn based on sparse matrix instead of dense one, therefore only a small portion of the files are required to be parsed and extra the token inside for us to classify all the files. We can extend the personal file system to corporate file system.

After achieving more desired performance of our classifier, we can further design UI to make this a public product for user to use on their computers or phones, or for the company to use on their servers.

8. Acknowledgement

Thanks for Professor Andrew Ng.'s great lectures and all the help from TAs.

References

http://cs229.stanford.edu/notes/cs229-notes7a.pdf. a.

http://en.wikipedia.org/wiki/K-means_clustering. b.

http://www.pdfmate.com/pdf-to-text.html. a.

http://www.outlookextractor.com/2013/02/convert-outlook-email-text.html. b.

http://www.mobileread.com/forums/showthread.php?t=138076. c.

Brian Kulis, Michael I. Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

D. M. Blei, A. Y. Ng and Jordan, M. I. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

M. Mesnier, E. Thereska, G. R. Ganger D. Ellard and Seltzer, M. File classification in self-storage systems. In *Proceedings of the First International Conference on Autonomic Computering (ICAC-04)*, New York, N.Y, 2004.

M. Sahami, S. Dumais, D. Herkerman and E.Horvitz. A bayesian approach to fitering junk e-mail. Technical report, AAAI Workshop on Learning for Text Categorization, Madison, Wisconsin, 1998.

Nathiya, G. and Punitha, S. C. An analytical study on behavior of clusters using k means, em and k-means algorithm. *International Journal of Computer Science and Information Security*, 7(3), 2010.

Sutton, Clifton D. Classification and regression trees, bagging, and boosting. *Handbook of Statistics*, 24, 2005.

Wallach, Hanna M. Topic modeling: Beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, Cambridge, UK, 2006.