Automating the Organizing Process for PDF Files

Samual Kahsay
Earlham College
801 National Road West
Richmond, Indiana 47374
sakahsay13@earlham.edu

1. ABSTRACT

The progression of technology is exponentiation in almost all fields. Technology makes many aspects of human life simpler, and less complex. The progress and the intent of innovation within technology/computers has in no way stopped, however some components within these systems have been forgotten. File organization is very much still a manual task. Whether these files are virtual or not the task of organizing them is equally tedious and labor inducing.There is a lot of room and need to create a space withing virtual data that takes the physical labor of organizing.

This paper presents a solution for automating files. The focus is PDF files in particular, however the general idea and algorithms used are capable of expanding to many other file formats. Using and expanding on a Open source program (tagspaces) that shares the same general idea to make organizing easier and more automatic. The expansion to the program includes similarity and metric learning to create an algorithm and source code for organizing PDFs and files types.

Keywords: *Tagspaces, Java Script, Open source, PDF, Kernel, Similarity Learning, Metric Learning, Organization*

2. INTRODUCTION

As humans, our brains take away the hassle of trying to organize all the data we take in manually. Color, texture, temperature, and many other elements are processed by our brain to help us differentiate and categorize. This process is inherent for the functionality of humans. We move through varying spectra of light, color, temperature and texture. This data is then used to process and come to a conclusion, often subconsciously. This process, when translated for hardware via software, is called similarity learning. Metric learning is the process in which we determine how different relative objects or images are mathematically. This is better explained of how we as humans define and process opposites. White is the opposite of black, meaning the distance between white and black is largest distance and the color gray would fall in the middle. To translate to a language a computer can understand, the abstract distance in this example is given values to calculate the similarities and differences between certain elements being tested. The applications of Similarity Learning can enhance ones everyday life, by automating most of oneâĂŹs virtual data.

The motivation behind this project comes from a personal need to organize my massive virtual data without the tedious manual labor. This application will be able to bypass the manual labor and automate the organization of PDF files. Using machine learning algorithms, to create an application that organizes PDF files based how similar they are to each other, creating a folder and placing all the files that are relative to a number of similarity.

3. RELATED PRIOR WORK

The following categories are separated by algorithms and open source program considered and included in the resulting pdf and file sorting software. These sections on the Kernal Algorithms and Tagspaces' Open Source documentation include a synopsis for understanding the base of the program. The last subsection is the preliminary results of test conducted on tagspaces and research applied on the software.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

3.1 Kernel Algorithms

Kernel Method is the name given to a set of algorithms for pattern recognition. These algorithms are the ones used to implement classification, hashing, ranking and regression similarity learning. This principle for algorithms is used to find different avenues in finding the patterns between raw data. The Kernel Method gives a base for a cheap and multidimensional system to implement similarity and metric learning. The algorithms that will be focused on for this application are Classification and Ranking similarity learning.

Classification similarity learning is given pairs of similar objects $(x_i,\ x_i^+)$ and non similar objects $(x_i,\ x_i^-)$. Using the two identities of the vector, a plane is created to place a given vector such as $(x_i^1,\ x_i^2)$ with a binary label y_i. The placement within the plane determines whether two objects are similar or not. This is repeated for every pair of objects introduced.

Ranking Similarity Learning is the easiest to implement. The function for Ranking similarity Learning is to use a initialized triplet input, which is per-ranked as a reference. The following inputs are then compared to the reference and ranked according to their similarities. Ranking is a cheaper and faster product of similarity and metric learning.

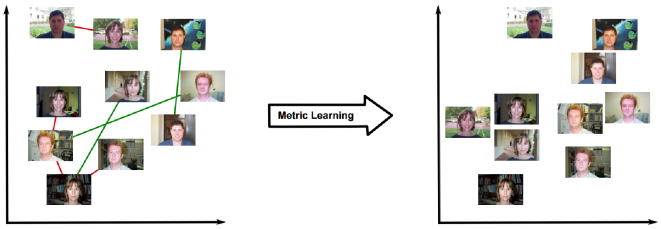

Figure 1: Metric Organization Graph Example [1]

3.2 Tagspaces Open Source

Tagspaces is an open source organizing application. It uses tags to better sort and find files. Tagspaces works with multiple file types. In this section we navigate through the process in which this project incorporated Tagspaces in the PDF and file sorting component. Tagspaces includes a preview for the files selected and a personal search engine. Using the source code in this application as a base to start my personalized PDF organizer. The Source code is in Java script, and works for windows. The goal, to produce an application using the resources and code provided by Tagspaces while incorporating a machine learning component (Kernel algorithms) and making it more automated.

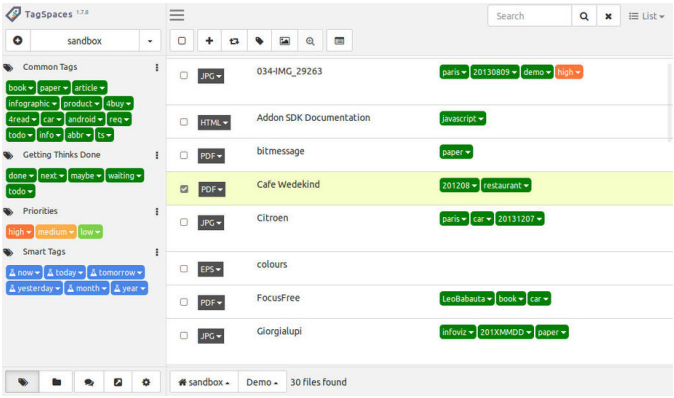

Figure 2: Tagspaces Layout

Having run Tagspaces and exploring the source code in windows. The key benefit of using Tagspaces is that it already has a very defined representational layout.

3.3 Preliminary Results

Test were conducted on the functionality of Tagspaces. These trial runs tested for a visual understanding of the algorithm used to run the software, understanding of the layout, accessibility in terms of complexity and where improvement was needed.

Tagspace once downloaded is a standalone program that mirrors a directory by linking and reflecting all the files within the directory to the programs window.The main operating function of the program is giving the user the ability to tag a file and push that tag in the metadata of the file for better arrangement customized to the users tags/input. Tagspaces has a large number of other features not found in other objectively similar software. One very unique feature is the organizing files by mapping and graphing the placement of the programs within the directory figure (). These types of features within the program add another layer of complexities. Tagspaces also has a preview window for some file extensions such as (pdf, docx, jpg, etc.).

From these test we can conclude, though Tagspaces is a very well organized and a interactive software with a lot of features, it doesn't allow you to be lazy. The complexities introduced with this software create a better system of organizing, however the tedious labor of organizing still fall on the user and not the program. This has shaped the software the outlook for this project. The application will be one that asks very little of the user, while still managing to organize the files according to certain defined criteria. Making it very

distinct from Tagspaces.

4. DESIGN

The personal virtual library requires as much mental and tedious labor as a physical library. This project hopes to create a space for an application that can make the virtual library automatically organized. This will be done by using certain Tagspaces algorithmic ideas from their source code and then incorporating a self-designed algorithm based on the Kernel Algorithms to create a PDF organizer, that is easy to operate.

This project is composed of two different applications that work in parallel. The first application organizes files based on their extension. The second works by reading through metadata of all the PDF files given as input, then placing files in folders based on keywords.

The following subsections navigate through the process for the program. The method, research and algorithm/source code for the program are the main elements that helped create the software.

4.1 Research

The research component consisted of web articles, open source compatibility searching, and looking for implementable algorithms using similarity and metric learning.

The articles [1] [2] [3] [8] were used as a base for this project. These articles were mostly concentrated on similarity and metric learning and or the Kernel algorithms. The application as is uses a very basic form of similarity and metric learning and the Kernel algorithms. The current demo of the program is very one dimensional. That being said, the understanding of the similarity and metric learning components of the software is still quintessential to if one wishes to expand on the current performance.

Image from University of Chicago, Computer Science Department. (http://ttic.uchicago.edu/gregory/pose/psh.html)

Figure 3 illustrates the basic product of metric learning. The points which in this case represent files are given a grid. The different sections within the grid will become folders. The files that exist within that grid then are placed into that folder.

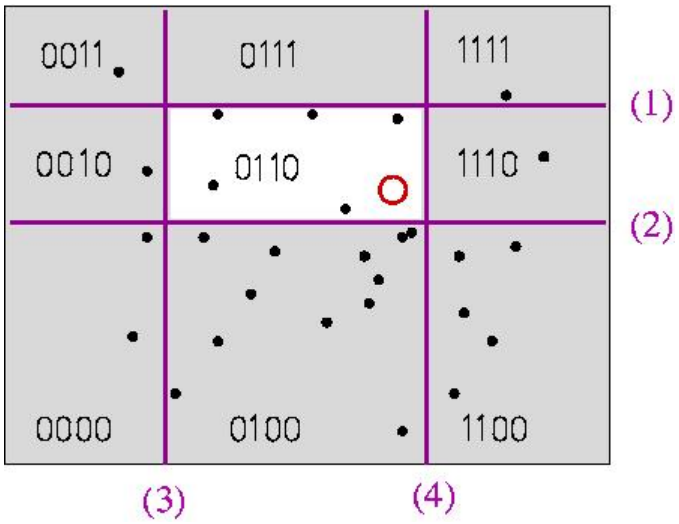

Figure 3: Assigning Files Within a Grid

The seconds aspect of the survey was looking for a open source program that would be compatible with windows. Tagspaces was among a short list of other programs, that is file organization based and open source. Tagspaces though a great program, is it not perfectly in sync with this project. The method algorithm and implementation of the Tagsapces program differs from my program in a lot of key ways. Tagspaces is used within my program as a reference for metadata searching. The second use for Tagspaces, though it has not yet been incorporated to my program, is the preview window implementation for sorted items.

An algorithm with a sudo-code component for similarity and metric learning was very much needed. The math is very complex, so having Tagspaces as a base to reference for my code was a great equalizer. The algorithm created though does not solely run using similarity and metric learning, it includes a general and basic mathematical mapping idea from the Ranking and classification algorithms (illustrated in figures 1 and 5).

4.2 Method

Research, reference open source documentation and source code, preliminary tests all took part in the creation of the outline of this application. From the outline and general algorithm created by these cases, the process of creating the actual program became feasible. The process includes materials such as Java as the main language for the program, online servers and local servers for compiling and a windows computer for testing.

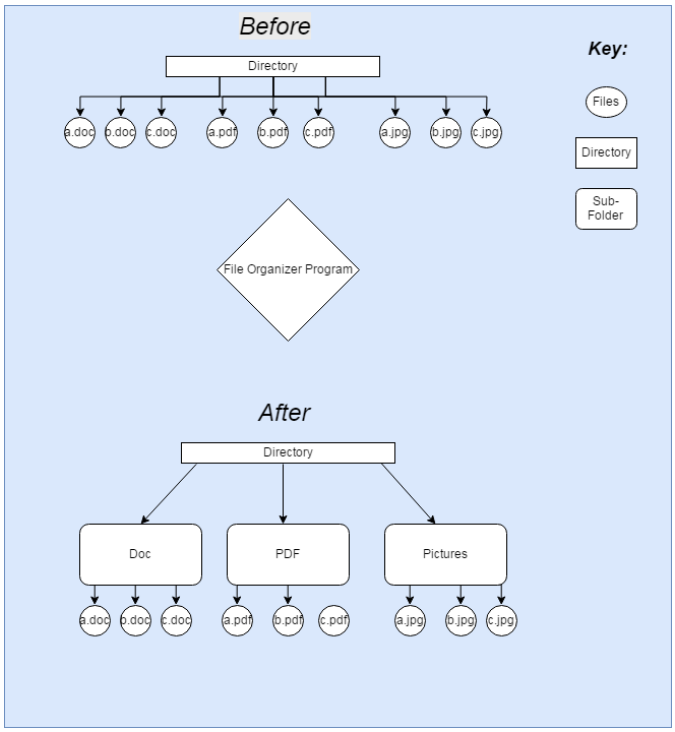

Figure 4: Mapping the Process of Organizing Based on Extension

The cluster machines at Earlham College as well as an online server, which can be found at compilejava.net, were used to run and compile the program. Initially the program was coded using node.js to mirror Tagspaces source code. However, the Tagspaces and FileSorter algorithm differed to vastly to work in parallel. From this, code snippets of Tagspaces source code were taken but then converted to java for better functionality and mobility as a program. The source file or parts of algorithms used from Tagspaces to move forward are sections from gettingstarted.js, core.js, meta.js, fileopener.js, tag.ui.js, and tagutils.js.

The next part was making edits on the source code to incorporate my algorithm. This took place in file called FileSorter.java and PDFSorter.java. The File Sorter code used sections from fileopener.js and code.js from Tagspaces. PDF Sorter code used tag.utils.js and fileopener.js. These sections initiated the first element of the program.

The second phase is the process in which the applications run. FileSorter.java and PDFSorter.java once compiled and executed become FileSorter.jar and PDFSorter.jar. FileSorter when run will prompt the user for a "custom extension" if the user has none then the simply continue. At this point the program looks at all the files that live within the directory, making a list of all the extensions. Once this occurs the extensions are organized and matched a folder. The folders are created with names of the type of files. The last step moves all the files to their matching folder. Hard coded file types include (jpg, html, pdf, docx, etc.).

PDFSorter works with relatively the same processes.This program is intended to mostly be used a with FileSorter in sequence. FileSorter creates a folder name "Work Space" where it will store all the PDF files. From there the PDFSorter will run. Key distinctions form the FileSorter are that it uses keywords instead of extensions. Using the metadata the user will be asked to supply keywords as input to search the metadata. When a keyword is matched with a file, a folder is created to place the file in. The software will also have an inventory of keywords hard coded such as (years, bank statements, assignments, etc.).

Once both programs are done sorting, a message window will inform the user that their files have now been sorted.

4.3 Algorithm

Their is hand full of source codes that are used, which are stated above. These programs are in java. The explicable file shown bellow is in java. The PDFSorter.Jar is the collaborative of all the snips taken from the different source codes. The following codes are sections from FileSorter. The sections represent the illustrate the process that occurs in the program.

```
public static void
        process(String ex, String d, String[] content) {
                File tmp = new File("");
               int i = 0;
    while (i < content.length) {
      if (!tmp.isDirectory() && content[i].endsWith(ex)) {
          tmp = new File(content[i]);
  FileOrgnizer.move(tmp.getAbsolutePath(),
String.valueOf(FileOrgnizer.genrate(tmp.getAbsolutePath())
 + d + "\\" + content[i]);
            }
             ++i;
        }
    }
```

The Above code is the process section. This is responsible for comparing and grouping file types. This is looped for every file in the directory.

```
public static String createFolder(String name) {
        File d = new File(name);
        d.mkdir();
        return name;
    }
```

The above code creates folders. The Name of the folders are hard coded or the user can chose to name a custom folder.

```
public static void
move(String from, String to){
    Path From = Paths.get(from, new String[0]);
    Path To = Paths.get(to, new String[0]);
     try {
       Files.move
(From, To, StandardCopyOption.ATOMIC_MOVE);
 }
         catch (Exception ex) {
             System.err.println(ex.getMessage());
         }
    }
```

The last section moves the grouped files from the process section into the folders created by the createFolder section.

5. CONCLUSION

Similarity and metric learning has expanded and opened up a new and innovate space for what computers are capable of. It produces many applications. The move to the public market will produce even more applications in the future. Snapchat, Iphones and Facebook all have implemented a system that deeply embeds and utilizes similarity and metric learning in different ways. The need to have a automated file organizing system is one of those applications. Using these algorithms, one can create a software that is complex enough to automate the storing and categorizing of ones personal data the way one would do it him or herself. This project is the product of that idea. This project creates the software needed to make once PDF files organize itself. We are living in the future, so as a result we should not have to organize our own files.

In conclusion, this project is a creative process in which one can discover everything can be easier. so why is it not? This projects aims to make life easier one organized PDF file at a time.

6. FUTURE WORK

In the future I wish to expand on the types of files this program will be compatible with. I wish to run similarity and metric algorithms on video and music files. This program now can easily work for a lot of text based files, however I wish to incorporate image recognition and sound filtering components to this project. Also, expand on what operating systems it works on. In the Very near future I will be working on PDFSorter to get rid of bugs such as (running but not functioning).

7. ACKNOWLEDGMENTS

This work was supported by the Earlham College Computer Science Department as part of the Senior Seminar project. Special thanks goes to my adviser Xunfei Jiang who guided me throughout this project. Lastly, a thank you to Bret Marshall, a fellow senior computer science major at Earlham College, who assisted me with figuring out java and java script along with letting me use his account to run my programs.

8. REFERENCES

[1] Aurelien Bellet and Matthieu Cord. Similarity and distance metric learning with applications to computer vision. *N.p.*, 7, September 2015.

[2] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. *IEEE Computer Society Conference*, 1, 2005.

[3] Andre Elisseeff and Jason Westo. A kernel method for multi-labelled classification. *BIOwulf Technologies*, 24, March 2016.

[4] David G. Lowe. Similarity metric learning for a variable-kernel classifier. *Computer Science Department University of British Columbia*, 25, Nov. 1995.

[5] Kilian Q. Weinberger. Distance metric learning for large margin nearest neighbor classification. *Department of Computer and Information Science, University of Pennsylvania*, 2005.

[6] Wikipidia. Similarity learning.

[7] Eric P. Xing. Distance metric learning, with application to clustering with side-information. *University of California, Berkeley, n*, 1, March 2003.

[8] Liu Yang. Distance metric learning: A comprehensive survey. *Department of Computer Science and Engineering Michigan State University*, 19, May 2006.