A DATA-DRIVEN REAL-TIME ANALYTICAL FRAMEWORK WITH IMPROVED GRANULARITY USING MACHINE LEARNING AND BIG DATA ANALYSIS

Yubo Zhang[1] and Yu Sun[2]

[1]Shenzhen College of International Education,
3 Antuoshan 6th Rd, Futian District, Shenzhen, China, 518043
[2]California State Polytechnic University, Pomona,
CA, 91768, Irvine, CA 92620, USA

ABSTRACT

During daily studying and working, people have to research massive amounts of information on the internet and download numerous files. Some of these files can be easily categorized to relevant files. However, there are always some files left unorganized due to their difficulty in categorization [1]. Such files pile up in the download folder as time passes, making the folder extremely messy. Many people do not have the motive to clean up the folder as it requires a lot of energy and time. Based on this common problem, my group developed an app that can clean the messy folder up. After applying our program which is based on machine learning, the files will firstly be divided into five general parts – document, video, music, photos and package. Then the files will be further categorized based on the contents they present. For example, photos are divided into animals, families and so on. In order to achieve the content-categorizing function, several powerful apis were introduced in our program, and they helped us to achieve the optimal results.

KEYWORDS

Data Processing, Deep Learning, Machine learning.

1. INTRODUCTION

I am a high school student who was previously annoyed by the messy download files on my computer. Everyday I download many files from websites, but I do not have enough time and energy to clean up these files one by one -- I just leave them there. Files gradually piled up and messed up my computer, so I wanted to find a way to clean up the files [8]. My first action was to find a suitable app in the app store. However, many of them were way too expensive and they were not personalized enough to achieve my wish – categorizing the files perfectly as if they were done by myself. Thus, I decided to create a personalized program that can help me clean up the files automatically. The first version of my program could only fit the environment on my computer as it was based on my personal preferences. It was indeed powerful when I tested it on my laptop – every file went to the most desirable place as I wished. However, it was not general enough to cater for others' needs to organize their files [3]. Thus, I made several amendments on the original code to reach the second version, which had a larger range of applications and could be used for the public. As a result, others encountering the same problem can also apply my program and then clean their folders up effectively.

David C. Wyld et al. (Eds): DMML, SEAS, ADCO, NLPI, SP, BDBS, CMCA, CSITEC - 2022
pp. 59-66, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.120705

There are numerous related methods on this topic. One example is the system developed decades ago to classify emails by scanning over their bodies, senders, receivers, and the titles. The results come from the comparisons between the preset keywords and the features extracted from the emails. Therefore, such classification can be unreliable in many cases. For example, if none of the preset keywords matches perfectly with the features, then the system will not be able to classify the documents, inhibiting its capability. Another example is using machine learning to detect the contents of images [9]. Such techniques have become very mature. By feeding the computer with numerous sample images, the computer can learn the features of nearly every commonly-used item such as cars and desserts, and its accuracy is quite high – higher than 85% in most cases. Thus, I will apply such techniques to achieve parts of my project. Another example is the concept of a new file organization system. Different with the traditional system that follows a tree structure, such systems import 'concepts' to replace the folders. One single file can be placed under numerous concepts so that the paths to the file vary. However, such a new idea has not been widely adopted yet, and the majority of the operation systems are still using the traditional organization methods. Thus, the new concept, though providing some novel approach to organization, is yet to be widely used.

In this paper, we follow the same line of research by adopting existing mature techniques into our system and develop the rest parts by using our own logic and algorithms. Our goal is to develop a system that can automatically clean up the messy download folder and then transfer each file to the proper place of the computer. Some of our methods are inspired by the well-researched methods such as the use of machine learning on classifying the pictures. Based on the results given by the system developed by such a classifying technique, our system is able to place each image into a proper place under the image folder, depending on the content of the image. Another strength of the program is its capability to categorize the files missing key information. For example, if there is an untitled audio under the download folder, the program is able to detect the content of the audio and then transfer it to the right place. It may find out that the audio is a song by a famous singer. As a result, the audio will be placed under the singer's folder, which is rational and effective.

In order to prove my results, I first operated the program on my laptop. The results were satisfying: all the documents and files went to the ideal places. My first experiment proved the effectiveness of the program on my laptop. However, when I ran the program on some of my friends' laptops, the results were not very good initially. One main reason was that they had different organizing habits with me: my categorizing methods did not match up with theirs. As a result, the program simply created several new folders in their laptops and directly transferred the files from the Download folder to the new ones. This was really bad as it actually further messed up their laptop. My friends still had to do the categorization work on their own, and they had to drag each file from the new folders rather than the download folder to the optimal ones, which was more tiring [2]. Despite detecting this problem, I was not able to solve it due to the fact that every person has unique habits. Fully personalizing the program requires techniques that are far beyond my abilities. Thus, I could just make some amendments on the program to make the logic of my categorization more reasonable at the current stage.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges that we met during the experiment and designing the sample; Section 3 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 2; Section 4 presents the relevant details about the experiment we did, following by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Choosing an appropriate categorization method

The first challenge that I needed to confront is choosing an appropriate categorization method. This problem was crucial and was required to be tackled before entering the code developing stage: it directly determined the structure of the program [10]. Numerous aspects had to be considered carefully in order to deal with this challenge perfectly. Firstly, the first layer of the categorization system was to be set up, dividing the files into several general parts, for instance, videos, pictures and documents. After that, a second layer was to be built, further categorizing each part in reasonable ways. For instance, the picture part could contain family, pet, travel and other sections. The categorization work was somehow difficult as the system was supposed to be reasonable for every user. Therefore, many considerations had to be taken into account to develop a general system that could cater for everyone's basic needs.

2.2. How to realize the categorization system designed in the last stage

The second challenge was how to realize the categorization system designed in the last stage. Firstly, I tried to develop the system based on the names of the file. However, I soon found that I could only complete the first layer of the system. This was because even though I could have a general categorization of the files by detecting the differences between their extensions, I could not further divide them up in most cases as the majority of their names did not contain useful information. Therefore, in order to make further categorization possible, I needed to enable the program to scan over the contents of files and make categorizing decisions based on the results. This was difficult to me as it required the application of deep learning, which I hardly knew anything about [11][12]. Thus, in order to achieve this, I searched numerous apis on the internet and fitted the most proper and powerful ones into my project, maximizing the rationality of my categorization.

2.3. Compatibility

Compatibility was the third challenge after resolving the last problem. For example, one particular api used for detecting the contents of pictures could work pretty well on my computer, whereas it could not run properly on my laptop as my laptop did not have a discrete graphics card. This problem could happen on many other laptops, weakening the power of my program. Thus, I needed to find apis that were not only powerful but also could be applied by the majority of the computers. This was somehow difficult. Even though some apis were powerful and were supposed to be compatible with every computer, running them successfully could use a lot of memory, leading to crashes on weak computers [13].

3. SOLUTION

In order to categorize the files, X Cleaner will firstly examine all files in the download folder. Based on the extensions of their names, the files will then be divided up to five general parts – Documents, Musics, Images, Videos, and others. If the file name contains enough information for the program to complete the categorization (it may include the author's name, subject name, etc.), then the program will directly use the existing information and transfer the files to proper places. Otherwise, the program will use additional tools to make up key information. For the music, documents, and images, the program will upload the files to the internet and use existing

algorithms to recognize related information. For example, information can cover the name (if missed in the file names), the album and the artists of different musical files. The existing online algorithm can also recognize the general contents of images and put them into various groups based on the results. Additionally, we have set numerous key words for each commonly used subject. Thus, by comparing the highest frequently used words in the documents and the existing key-word base, the program can make a precise judgment on the related subjects of documents, finally transferring the files to the right places. However, a different categorizing method is applied on the videos. Since examining the contents of videos is too difficult at the current stage, videos lacking crucial information will be directly transferred to a folder named 'other videos' under the video folder. Finally, the rest of the files like zip files will be moved to the other folder.

```
all_file_path = get_all_file_path('%s/Downloads' % (get_my_user_folder_path()))
i = 0
while i < len(all_file_path):
    try:
        source_file_path = all_file_path[i]
        split_filename = os.path.splitext(source_file_path)
        file_ext = str.lower(split_filename[1])
        if file_ext in music_ext:
            if music(source_file_path):
                num_organized += 1
            else:
                num_not_organized += 1
        elif file_ext in video_ext:
            if video(source_file_path):
                num_organized += 1
            else:
                num_not_organized += 1
```

Figure 1. Screenshot of code 1

In order to achieve the first step, firstly five general parts are given with frequently used extensions respectively. For example, extensions like 'jpg', 'jpes', and 'png' are under the image part. After assigning the extensions, the program will then loop through the whole download folder, as shown in the picture abode, and finish the first step up with all files flowing into five general sections. Further categorizing work will then take place.

```
def images(source_file_path: str) -> bool:
    try:
        api_key = 'acc_b580a33df0d99bb'
        api_secret = '508be5bf54a6c634ee73057797de606f'
        upload_info = requests.post(
            'https://api.imagga.com/v2/uploads',
            auth=(api_key, api_secret),
            files={'image': open(source_file_path, 'rb')},
        )
```

Figure 2. Screenshot of code 2

A crucial part of the second step is the disposal of images. The code abode is used to upload the files assigned to the image section to the existing online api. Then, the well-trained online program will go through each file and compare its features with the past information, determining

the type of its content. After that, the result will be sent back to the program. Based on the result, the program will transfer each image file to the appropriate place under the image folder.

```
try:
    audio_file = eyed3.load(source_file_path)
    if audio_file.tag == None:
        pass
    else:
        if audio_file.tag.album_artist == None:
            audio_file.tag.album_artist = 'unknown_artist'
        else:
            audio_file.tag.album_artist = change_forbidden_filename(audio_file.tag.album_artist)
        if audio_file.tag.album == None:
            audio_file.tag.album = 'unknown_album'
```

Figure 3. Screenshot of code 3

Another crucial part is processing the music files. The logic behind this part is very similar to the one of image section. Online api is applied again to get the necessary information. However, the result will be more precise and detailed this time. By examining the features of the audios and making comparisons to the audios around the world, the api can not only send back information about the general type of the music but also the name, the album, and even the author of the music. This is very beneficial for further categorization. If an artist has never appeared on the laptop, the program will create a folder using his name and then move the file to a minor folder named by the album under the artist name. Otherwise, if the artist folder exists but the album folder is absent, the program will create the minor folder again. Such a categorizing method is very detailed and effective.

The program will categorize each file one by one, according to their sequence in the download folder. After the last file entering the right place, the program will print out a line, notifying that the categorizing work has been finished. The notice will also include the number of files categorized. The whole process follows a simple but rigorous logic: recognizing the type of the file and then further categorizing it by examining its features and content [4]. In order for the program to run successfully, having a reliable network connection is indispensable. This is because the online apis take an important role in examining the contents of the files. Otherwise, every laptop using the program has to be trained properly long before starting the program to acquire the ability of recognizing contents offline.

4. EXPERIMENT

4.1. Experiment 1

In order to test the usefulness of our program, I borrowed five laptops, which showed some variety between the samples, from my friends and then ran the program on them. The process of the first experiment is very simple. Firstly, the program will be run on a laptop with multiple documents in its download folder. After that, we will check if the program crashes during the previous process. If the program does not crash, we will check if all files are cleaned up from the download folder and moved to the proper places. The time taken to clean up the files will also be measured.

The experiment went through quite well. All files from five laptops were cleaned up and were moved to ideal places. The average time taken to clean up each file was estimated to be around 2.5 seconds – – this was not very fast but still acceptable. Thus, a simple conclusion could be drawn: the program was powerful enough to cope with daily problems for a high-school student.

However, one thing worth mentioning was that 4 laptops, taking up 80% of the samples, were macbooks. Therefore, even though the program ran smoothly on the only laptop with a windows system, the program had yet been proven to be strong and stable enough in such a system.

4.2. Experiment 2

In order to test the effectiveness of the results, I designed a survey for 20 of my friends and asked them about the feeling of running my program. The survey had two parts: the first part was about the attitude towards the general categorization idea behind the program and the second part was their feeling about the final results.

The results showed that 17 people, taking up 85% of the samples, deemed the logic behind the program was reasonable. However, only 10% of them, which were 2 people, were satisfied with the final categorization results. The results of the survey were reasonable. Firstly, the general logic of categorizing matched up with the general needs and habits of the majorities – putting files of the same type together. However, as the categorizing work moved into more detailed parts such as the separations between document files, the effectiveness of the program was significantly limited.Since everyone has unique detailed habits and requirements, the unsatisfactory results were reasonable and predictable.

The results matched up with previous challenges to a large extent. The first experiment shows that the program could work smoothly in the majority of the environments. This means that the third challenge has been tackled successfully with the environment becoming compatible with numerous devices. In addition, the second experiment shows that the general logic behind the categorizing method is rational while it will encounter problems when dealing with more detailed parts. This matched up with the first and the second challenge. The first challenge has been successfully solved – we have created a categorizing system that could properly function. The difficulty of solving the second challenge is further addressed. Even though we could use the program to clean up the download folder, the result is still not personalized enough to cater to the customer's specific needs and habits.

5. RELATED WORK

In the first reference work, Kendrick the author has created a system to categorize the emails by using a nearest-neighbor classifier [5]. Such a classifier will compare the features extracted from the emails and the given target message, and then move the emails to the most matching group based on the result. Such a structure is similar to my approach to categorizing the documents: both having preset key words for classification. Kendrick's system is more powerful than mine as it will consider more features of the emails. While my work only extracts the features from the body of a document, Kendrick's system also takes the author and other parts into account. This can further increase the reliability of the classifier with more information provided.

In the second reference work, a new file management method, called CMF, is introduced [6]. Unlike traditional hierarchical file systems that follow a 'tree structure', CMF contains two essential parts, concepts and containment, which are converted from a folder. Like being assigned to a folder in a traditional system, a file is assigned with concepts in CMF. However, rather than being limited in a single place, a file can be under multiple concepts in such a system. The concepts all contain paths to the file, and users can access to the file through various paths. CMF is very different from my approach as my system follows the traditional way. The biggest advantage of CMF is that it can create quick access to files and maximize personalization. Once I

introduce such a system effectively to my existing system, the problem of lack of personalization can be tackled.

In the third reference work, the method of analyzing the content of images is introduced [7]. The model is trained numerous times to acquire the features of various groups of images. The api used in our program for image categorization shares the same concept with the reference work – training the model with a large number of preset images. Unlike the reference work in which the model is trained directly in the computer, my program does not download the model. When there is a need for image categorization, my program will send the image to the online api and retrieve the result. The advantage is that the users will not have to download the picture-categorizing system that can take up much space in their laptops.

6. CONCLUSIONS

In conclusion, we developed a system that can categorize different types of files and make further classification within the same type [14]. Machine learning is applied in the program, enabling it to identify the contents of pictures and audios. In addition, in order to categorize documents lacking key information, the program can extract features from their bodies and make comparisons with preset key words, reaching final results. Based on the classification results, the program is able to create branches under each type and then transfer the files to proper places. Two experiments have been conducted to test the effectiveness of the program. The first experiment showed that the program can run quite steadily on different devices. And the results matched perfectly with the assumptions. Thus, the first and second challenges were solved. A rational categorization system was built, and the program perfectly realized such a system. The second experiment further proved the rationality of the system — the majority of the users agreed with the logic behind the categorization system. However, the second experiment also revealed the fact that the third challenge is yet to be solved. Almost all the users deemed that the system is not personalized enough. Though the overall structure is reasonable, the system cannot be altered by users to satisfy their special needs and habits. Thus, the system is not very effective when it comes to details, and more personalized parts need to be created.

The accuracy of my method is proved to be very high – almost all the files are transferred to the most ideal places on the laptops. However, its practicability is relatively low. As mentioned before, the method does not contain choices for personalization. Therefore, all users have to follow the exact categorization structure designed by me. This is not practical enough as every person has different habits and needs. The users may find their personal needs unsatisfied in places where my habits conflict with theirs. In short, the program still has room for further optimization.

In order to improve the practicability of the program, I will add parts for personalization to the program [15]. As a result, the users can enter their special needs and habits for the program to follow. In addition, I will improve its capability of classifying the contents of pictures. Currently, the program is able to classify the pictures into over twenty types based on their contents. A refined training can help the program make further categorization within existing types, maximizing its effectiveness.

REFERENCES

[1] Rosch, Eleanor, and Barbara Bloom Lloyd, eds. "Cognition and categorization." (1978).

[2] Medin, Douglas L., and Evan Heit. "Categorization." Cognitive science. Academic Press, 1999. 99-143.

[3] Mock, Kenricj. "An experimental framework for email categorization and management." Proceedings of the 24th annual international ACM Sigir conference on research and development in information retrieval. 2001.

[4] Badashian A S, Mahdavi M, Afzali S H, et al. Supporting Multiple Categorization using Conceptual File Management[J]. American Journal of Scientific Research, ISSN, 2011: 129-136.

[5] Chen Y, Wang J Z. Image categorization by learning and reasoning with regions[J]. The Journal of Machine Learning Research, 2004, 5: 913-939.

[6] Truong B T, Dorai C. Automatic genre identification for content-based video categorization[C]//Proceedings 15th International Conference on Pattern Recognition. ICPR-2000. IEEE, 2000, 4: 230-233.

[7] Lin C C, Chen S H, Truong T K, et al. Audio classification and categorization based on wavelets and support vector machine[J]. IEEE Transactions on Speech and Audio Processing, 2005, 13(5): 644-651.

[8] Tuemmler, Brian. "Network shared drives: How to clean up files for better information management." Information Management 46.1 (2012): 26.

[9] El Naqa, Issam, and Martin J. Murphy. "What is machine learning?." machine learning in radiation oncology. Springer, Cham, 2015. 3-11.

[10] Rist, Robert S. "Program structure and design." Cognitive science 19.4 (1995): 507-562.

[11] Deng, Li, and Dong Yu. "Deep learning: methods and applications." Foundations and trends in signal processing 7.3–4 (2014): 197-387.

[12] Hao, Xing, Guigang Zhang, and Shang Ma. "Deep learning." International Journal of Semantic Computing 10.03 (2016): 417-439.

[13] Gu, Xiaodong, et al. "Deep API learning." Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering. 2016.

[14] Cormack, Richard M. "A review of classification." Journal of the Royal Statistical Society: Series A (General) 134.3 (1971): 321-353.

[15] Vesanen, Jari. "What is personalization? A conceptual framework." European Journal of Marketing (2007).