Computers, Materials & Continua
DOI: 10.32604/cmc.2022.029400
Article

Tech Science Press

Content Based Automated File Organization Using Machine Learning Approaches

Syed Ali Raza[1,2], Sagheer Abbas[1], Taher M. Ghazal[3,4], Muhammad Adnan Khan[5,6], Munir Ahmad[1] and Hussam Al Hamadi[7,*]

[1]School of Computer Science, National College of Business Administration & Economics, Lahore, 54000, Pakistan
[2]Department of Computer Science, GC University Lahore, Pakistan
[3]Center for Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Selangor, Malaysia
[4]School of Information Technology, Skyline University College, University City Sharjah, Sharjah, 1797, UAE
[5]Riphah School of Computing & Innovation, Faculty of Computing, Riphah International University Lahore Campus, Lahore, 54000, Pakistan
[6]Department of Software, Pattern Recognition and Machine Learning Lab, Gachon University, Seongnam, 13120, Gyeonggido, Korea
[7]Cyber-Physical Systems, Khalifa University, Abu Dhabi, 127788, UAE
*Corresponding Author: Hussam Al Hamadi. Email: hussam.alhamadi@ku.ac.ae
Received: 02 March 2022; Accepted: 07 April 2022

Abstract: In the world of big data, it's quite a task to organize different files based on their similarities. Dealing with heterogeneous data and keeping a record of every single file stored in any folder is one of the biggest problems encountered by almost every computer user. Much of file management related tasks will be solved if the files on any operating system are somehow categorized according to their similarities. Then, the browsing process can be performed quickly and easily. This research aims to design a system to automatically organize files based on their similarities in terms of content. The proposed methodology is based on a novel strategy that employs the charactaristics of both supervised and unsupervised machine learning approaches for learning categories of digital files stored on any computer system. The results demonstrate that the proposed architecture can effectively and efficiently address the file organization challenges using real-world user files. The results suggest that the proposed system has great potential to automatically categorize almost all of the user files based on their content. The proposed system is completely automated and does not require any human effort in managing the files and the task of file organization become more efficient as the number of files grows.

Keywords: File organization; natural language processing; machine learning

1 Introduction

Classification of documents such as customer feedback, product reviews and e-mails etc., is conventionally considered as an essential task in various fields like information retrieval, extraction and recognition etc. [1]. Initial classes for the documents can be known or unknown initially. Yet, the classification task can abridge the task of information processing and may also escalate the performance of information processing systems [2]. Different machine learning approaches like clustering, classification and optimization have been utilized previously for this purpose [3]. Generally, these approaches use either textual content or the document's overall structure to categorize them into different classes [4]. However, most of the research in document classification is limited to the classification or clustering of plain text documents. Yet, very little concentration has been given to the automatic organization of different user files stored on computer systems based on their content. Previously multiple file organizers and file managers are available for different operating systems, but they do not manage files in an automated way or categorize files based on file extensions. This research paper addresses the file organization problem by considering the content of the user file and categorizing textual files of different extensions on any computer system.

Most of the clustering algorithms [5,6] available so far are data-centric, which means that they are not ideal in recognizing labeled clusters [5]. These algorithms are not adoptable for grouping files on computer systems [6] since they generally group documents into non-labeled clusters [7]. Optimization algorithms, for example, genetic algorithm, Ant Bee and particle swarm optimization, have also been used to identify optimal cluster count [8]. The fitness function in such approaches is to find the semantic similarity of the documents in a given cluster [9,10]. These approaches are limited to pre-defined forms and are not generalizable to domain-independent documents [11–13]. A few attempts have also been made for document clustering based on ontologies [14–17]. A significant limitation of these systems is that as soon as the ontology size expands, it becomes difficult and time-consuming to assign clusters to new documents. The observed computational complexities of all of these benchmarking models are nonlinear [18]. And cluster count and cluster optimality are questionable due to the curse dimensionality reduction by semantic relevance [10,19]. These models are least significant to define optimal cluster count for document set with fewer divergence [7]. Since the complexity of the traditional evolutionary strategies like Genetic Algorithm (GA), the process complexity is observed as nonlinear [10]. A few attempts have been made to fine-learning through supervised training after the unsupervised pre-training [20,21]. Likewise, supervised pre-training has been shown supportive in different image processing problems [22,23]. Ross et al. [24] observed that supervised pre-training on extensive scarce data with fine-tuning on minor problem-specific datasets improves classification results.

The purpose of this research is to minimize the human efforts to the point where a user doesn't have to worry about first analyzing the document and then storing it into the folder of similar documents. Furthermore, addressing the issues faced in document clustering and providing the solution is also a significant focus of this research. The organization and management of digital documents and folders and their non-digital counterparts have remained the focus of study for a long time. However, current file organizers and file managers available for different operating systems either do not manage files in an automated way or categorize files based on file extensions [9]. Dinneen et al. [12] reported that file management simulates relatively unsupported activity that invites attention from different disciplines with wide significance for topics across computer sciences.

Another significant aspect of file organization and categorization is how the file categories or folders are named, as generating descriptive and meaningful names for both files and folders aids users

find, apprehend and organize files [13]. On the other hand, this task is also tricky, particularly if the user wants to generate concise and unique names [25]. Users usually tend to demonstrate substantial creativity in file naming [14]. However, the file naming patterns are recognizable such as files are named to display the document they represent, their purpose, or a relevant creation date or deadline [15] but may also contain characters to expedite sorting of the files. This research addresses the file organization problem by considering the content of the file document and using it to categorize textual files of different extensions on any computer system. The system will work as a file manager. When the user adds a document, it will create folder categories, generate names of the folder, and categorize files into relevant folders automatically. The proposed system will help to reduce the efforts in managing documents in an automated and intelligent manner.

2 Materials and Methods

The task of automatic file organization is one of the most critical and time-consuming issues. Most non-technical users often do not store files in separate folders. Eventually, main folders like desktop, document or downloads in the Windows operating system appear to bear hundreds of files. As the number of files in any particular folder increases, it becomes more challenging to manage these files. The proposed system handles all these issues and is the first attempt towards an automatic file management system for operating systems. The proposed architecture consists of two different modules, named global and localize modules, as shown in Fig. 1. The global module is responsible for the overall learning of categories and category labels. At the same time, the local organizer classifies and stores the files in relevant folders based on the similarity of the files. Global module passes the trained model of neural network and the parameter settings for data preprocessing, feature extraction and truncated singular value decompositions. K-Means clustering approach is used for initial clustering of files which is fine-tuned with Adam optimizer to predict final classes. Once the number of documents in any folder increases to a certain threshold, the local organizer calls the global organizer, which may restructure the folders and rearrange the files into relevant folders.

2.1 Global Module

The global module is subdivided into Preprocessing module, Feature Extraction Module, Dimensionality Reduction Module, Clustering Module, Analysis Module, Topic Extraction Module and Neural Network training module.

2.1.1 Preprocessing Module

Most of the files in the computer system are usually classified as unstructured documents; therefore, almost all of the files must be pre-processed before the actual clustering begins. The preprocessing of the files includes multiple steps such as Filtering, Stop Word Removal, Tokenization and Stemming of document words.

Filtering: The task of filtering is to clean the data for all links, punctuation marks, digits and unnecessary white spaces.

Stop Word Removal: The task of this sub-module is to remove stop words that don't hold any valuable meanings and increase the length of the document unnecessarily from a text analysis perspective. Natural Language ToolKit (NLTK) [16] library has been used in this research to eliminate all stop words.

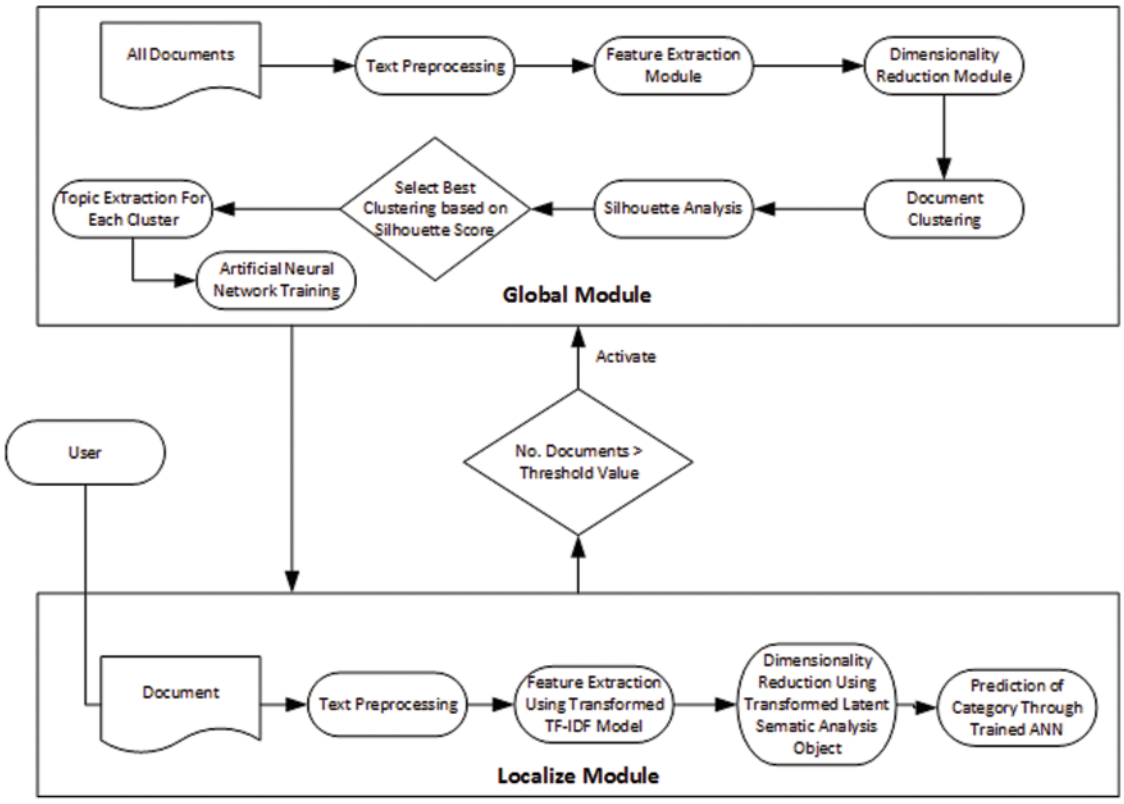

Figure 1: Proposed architecture for automatic file organization system

Tokenization: Once stop words are removed from the document, tokens are assigned to all remaining words to analyse the text better. The tokenization task is also done by using the NLTK library.

Stemming: Stem provides the root word of every token to find important terms used in the document. Snowball stemming algorithm [26] is one of the most popular stemming approaches for English Language documents used in this research to stem all tokens. Once stemming is done, a Stem Token Dictionary (STD) is maintained for topic extraction modules. Each entry in this dictionary holds a stem with all corresponding tokens.

2.1.2 Feature Extraction Module

Once pre-processing of the files is done, the essential tokens from the files are extracted. For this purpose, it is necessary to present all files in a suitable form. The vector space model is one of the most common approaches representing documents as a vector space [20]. To find the similarity between different files frequency of the words is used. In this research, each file Fi is located in an n-dimensional vector space where all n dimensions are extracted using the Term Frequency Inverse Document Frequency (TFIDF) vectorizer. It means that all components of a vector reflect a term within the given file. The value of each component is determined by the degree of association among the corresponding file and associated term. The TFIDF value can be calculated as mentioned in Eq. (1).

$$TFIDF = tf(t) * idf(t) \quad (1)$$

Tf (t) is the number of times the term t appears in a file divided by the total number of terms in the file, and idf(t) is the loge of a total number of files divided by the total number of files with term t. Term frequency finds out the frequency for each unique word and divides it with a number of words in that file, making each file having a unit length, resulting in no biasness towards any file. If the value of TFIDF is higher, it means that the associative term is rare, and if the value of TFIDF is smaller than the weight, it means that the associative term is more common. A minimum threshold mindf and maximum threshold maxdf are being used to check if any term appears in the number of documents more remarkable than the maximum threshold or lower than the min threshold, then that term is pruned. The threshold value for mindf is 2, and the maxdf is 0.5. The algorithm for the feature extraction module is mentioned below in Algorithm 1.

The output of the TFIDF vectorizer with the threshold as mentioned above values for mindf and maxdf is a TFIDF matrix of shape (18846, 32329) where 18846 are the no. of documents 32329 are the features extracted by TFIDF vectorizer.

1. *For all Files F_i (where i= 1 to total number of files) extract all terms tj (where j=1 to total number of terms*
    a. *For all terms tj Compute TFIDF Vectorizer (where max_{df}=0.5, min_{df}=2 and maximum number of features=200000)*
2. *For all training set learn the vocabulary and IDF then transform document-term matrix*

Algorithm 1: Algorithm for feature extraction module

2.1.3 Dimensionality Reduction Module

The files on a user computer system may contain very high dimensional data, and it becomes a formidable problem while applying statistical or machine reasoning methods on such data. For instance, the feature extraction module in this research identified 32329 features from the given dataset, which is a very high number. It is challenging to train a clustering or classification algorithm for this number of input variables. Therefore, it is necessary to reduce extracted features before a clustering or classification algorithm can be successfully applied to the dataset [27]. There can be two possible ways to perform dimensionality reduction. One way is to keep the most appropriate variables from the original dataset. The second way is to exploit the redundancy of the features in different ways and find a smaller set of new variables. In the second approach, a smaller set of new variables is identified. Each set is the combination of the input variables containing the same information as the input variables.

In this research, V × f document-term matrix D is very important where V is not equal to f, and it is implausible that D is symmetric. The first task in dimensionality reduction is to find Singular Value Decomposition (SVD). For the decomposition D, let X be the V × V matrix where columns of the matrix are orthogonal eigenvectors of DDT, and Y be the f × f matrix where columns are the orthogonal eigenvectors of DTD. These X and Y represent the left singular vector and right singular vector, respectively. Denote by DT the transpose of a matrix D is given in Eq. (2).

$$D = \mu \sum X^T \quad (2)$$

where the eigenvalues of DD^T are the same as the eigenvalues of DTD, the orthonormal columns X, Y can be represented as Eqs. (3) and (4).

$$X^T X = I \tag{3}$$

$$Y^T Y = I \tag{4}$$

The next step is to find the Truncated SVD, which is the subtraction of low-rank approximation from SVD. Although the SVD can solve this approximation given a V × f matrix X and a positive integer k, it is essential to find a V × f matrix Xk of rank as a maximum k to minimise the matrix difference, as computed in Eq. (5).

$$A = X - X_k \tag{5}$$

The value of A is the measure of inconsistency between the matrixes Xk and X. To find the Truncated SVD, this inconsistency needs to be minimized while limiting Xk to have ranked at most k as mentioned in Eq. (6).

$$min_{z\,|rank\,(z)=k}\ ||X - Z||_F = ||X - X_k||_F \tag{6}$$

2.1.4 K-Means Clustering

The k-means clustering algorithm [21] is widely used in document clustering problems and is acknowledged as an efficient approach to clustering large data sets [19]. The K in this algorithm is a pre-defined or user-defined constant, and the objective of the algorithm is to create a partition of data based on features into k clusters. K-means algorithm performs well on linear time complexity for the varying number of documents and especially gives better performance on a large number of documents [28]. As it is clear from the previous discussion, thousands of user-generated files are stored on any computer system; therefore, the K-Means algorithm can provide better and efficient initial clusters [29]. The main idea of K-Means is to define centroids for all clusters. These centroids are represented by k and are formed so that all features in a cluster are closely related to the centroid in terms of similarity function. This similarity of features can be measured through various methods, including cosine similarity and Euclidean distance, to all objects in that cluster [30]. The algorithm's output is a set of K cluster centroids where each k has a corresponding label L. The clusters are updated after μ_k the set of centroids is obtained, to hold the points nearest to each centroid. The centroids are recomputed as the mean of all cluster points on μ_k. This process is repeated unless no change in the centroids and the assignments of clusters is observed.

2.1.5 Silhouette Analysis

The major problem with K-Means clustering is that the number of centroids is defined before the clustering begins [31]. In file categorization, it becomes almost impossible to predict the number of folders/categories before the assignment of clusters. Therefore, silhouette analysis is used in this research to identify the quality and separation distance between the resulting clusters. Silhouette analysis results in a range of [−1, 1] where coefficients of the clusters near positive 1 specify that the current cluster is at a reasonable distance from its neighboring clusters. If the value returned is 0, it specifies that the current cluster is close to the boundary and lies precisely on the boundary between two neighboring clusters. A negative value specifies that current objects have been allocated to the wrong cluster. One way to examine the impression mentioned above is by observing the Average Silhouette Coefficient (ASC) evolution for each cluster. If the value of ASC is high, it specifies that

the clustering is good, whereas a small ASC value indicates that it is uncertain to which cluster these elements belong.

Furthermore, a negative ASC value specifies that the elements in consideration do not belong to a relevant cluster. The Silhouette Coefficient is calculated using the average distance between clusters distance represented by 'a' and the average distance with the nearest cluster represented by 'b' for each sample. The Silhouette Coefficient for a sample can be calculated using Eq. (7).

$$S = \frac{(b-a)}{\max(a,b)} \tag{7}$$

The number of clusters 'K' is selected with having the highest ASC, as mentioned in Eq. (8).

$$ASC = S_{avg}(X, Labels) \tag{8}$$

Once ASC is computed for all clusters, the next step is to use a threshold value as a cut-off point for selecting the total number of clusters as computed through Eq. (9).

$$\theta_{ASC} = \left| \frac{min_{ASC} + max_{ASC}}{2} \right| \tag{9}$$

where min_{ASC} represents the minimum value of all corresponding ASC and max_{ASC} is the value of maximum ASC among all corresponding ASC. Every ASC score is compared with the θ_{ASC} value and the value of K is set till the point where the condition ASC score $> \theta_{ASC}$ remains true.

2.1.6 Topic Extraction

Topic extraction is an essential step for the overall working of the proposed system. Therefore, in this research, a new strategy for topic extraction is proposed, as mentioned in Algorithm 2. First of all, top terms per cluster are selected through the centroid of the cluster. Then, since dimensions have been reduced for training purposes, all features are inverse transformed to get the original feature map. Then, the top 10 terms from each TFIDF vectorizer are selected, and for each term, the lemma is obtained using the NLTK library. Once the lemma is obtained, it is replaced with its hypernym to get the most generic term. Finally, the most frequent hypernym is selected as the topic of the cluster—Tab. 1 give a glimpse of extracted topics from the given dataset.

- *Initialize centers $c_1, \ldots, c_n \in R$*
- *Initialize all clusters from $C_1, \ldots, C_k$*
- *repeat until there is no further change in clusters*
  - *for each j: Cj ← {x ∈ S whose closest center is c_j}*
  - *for each j: c_j ← mean(Cj)*

Algorithm 2: Proposed algorithm for topic extraction

Table 1: Examples of extracted labels using proposed algorithm of topic extraction

Cluster	Terms (Stems)	Topic
5	gun control crime law weapon firearm cimin kill would peopl	instrument

(Continued)

Table 1: Continued

Cluster	Terms (Stems)	Topic
9	christian religion believ belief atheist god say faith bibl peopl	establishment
13	drug test caus day effect also one doctor medic patient	medical_practitioner
16	card driver video monitor vga use window bus color mode	video_display
19	drive disk scsi hard floppi ide mb use control problem	work
20	jesus god Christian church sin bibl Christ word one say	sacred_writing
24	space orbit nasa launch shuttl mission satellit cost would earth	location

2.1.7 Adaptive Moment Estimation

Adaptive Moment Estimation (ADAM) [32] optimizer is selected in this research because ADAM is direct to implement, is efficient in terms of computational cost, and requires very little memory. Furthermore, ADAM is an invariant to diagonal rescaling of the gradients and is suitable for problems containing large datasets or have a large number of parameters. The method is also suitable for non-static goals and problems with very noisy and/or sparse gradients. As the files stored on the computer system are sparse and typically contain many parameters and data, ADAM can provide better results than other gradients' variants. ADAM computes individual adaptive learning rates for different parameters from approximations of first and second moments of the gradients where mt and vt are approximations of the mean and non-centred variance of gradients, respectively, as computed in Eqs. (10) and (11).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t \tag{10}$$

$$v_t = \beta_t v_t + (1 - \beta_2)\, g^2{}_t \tag{11}$$

As mt and vt are initiated as vectors of 0's, the values of mt and vt are biased towards 0, mainly when the rate of decay is low and at the time of initial steps. However, these biases are counteracted through bias-corrected approximations of a first and second moment, as mentioned in Eqs. (12) and (13).

$$\hat{m}_t = \frac{m_t}{1 - \beta^t{}_1} \tag{12}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta^t{}_2} \tag{13}$$

These bias-corrected approximations are then used for parameters updating, which yields the update rule as mentioned in Eq. (14).

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{14}$$

2.2 Localize Module

The global module acts as a background process while the user interacts with the localize module. The localize module works as a user interface and is responsible for receiving files from the user, preprocessing the files, extracting features, and reducing dimensions for better analysis. After dimensionality reduction, localize module classify the received file based on the model and parameter

settings received through the global module. The preprocessing, feature extraction, and dimensionality reduction of file in localize module is similar to the global module.

3 Results and Discussion

The results of the proposed methodology are evaluated in this section. First of all, the experimental setup is explained, followed by a comparative account of the results generated from different levels of training of the proposed system. Then, the results are discussed with the deployment of the system with actual user-generated files. It is a known fact that every individual can have files of different nature and counts stored on their computer system; therefore, it is complicated to generate a benchmark dataset for file categorization problems. For experimental purposes, initial training of K-Means clustering and ADAM optimizer is done on 20 Newsgroup data set [33], which have been used widely for applications related to text analysis. 18846 subset documents were selected from the dataset on various topics as listed in Tab. 2. Each subset consists of randomly chosen documents from various newsgroups. All the documents header and main titles were removed to remove biasness in topic extraction and clustering.

Table 2: Statistics on experimental data

Sr. No.	Topics	Total No. of Docs.	Sr. No.	Topics	Total No. of Docs.
1	alt.atheism	799	11	rec.sport.hockey	999
2	comp.graphics	973	12	sci.crypt	991
3	comp.os.ms-windows.misc	985	13	sci.electronics	984
4	comp.sys.ibm	982	14	sci.med	990
5	comp.sys.mac	963	15	sci.space	987
6	comp.windows.x	988	16	soc.religion	997
7	misc.forsale	975	17	talk.politics.guns	910
8	rec.autos	990	18	talk.politics.mideast	940
9	rec.motorcycles	996	19	talk.politics	775
10	rec.sport.baseball	994	20	talk.religion	628

After preprocessing and feature extraction task is completed, this K-Means algorithm was trained on this dataset with no. clusters ranging from 2–40. For analysis of the cluster quality, ASC for each K-Means result was computed, as shown in Tab. 3. The values in Tab. 3 expresses the ASC concerning several clusters generated by the K-Means Algorithm. The cutoff point is computed using Eq. (4), which is 0.003. Therefore, the number of clusters selected for K-Means was set to 26 because the value of cluster 27 decreases from the computed threshold.

Table 3: Average silhouette coefficients with respect to number of clusters generated by k-means algorithm

Clusters	2	3	4	5	6	7	8	9	10	11	12	13	14
ASC	0.0092	0.0066	0.0066	0.0068	0.0078	0.0075	0.0075	0.0067	0.007	0.0056	0.0055	0.0061	0.0059
Clusters	15	16	17	18	19	20	21	22	23	24	25	26	27
ASC	0.0051	0.0101	0.0046	0.0061	0.0098	0.0075	0.0065	0.0048	0.0046	0.0095	0.0068	0.0097	−0.0065

(Continued)

Table 3: Continued

Clusters	2	3	4	5	6	7	8	9	10	11	12	13	14
Clusters	28	29	30	31	32	33	34	35	36	37	38	39	40
ASC	0.0082	0.0072	−0.0164	−0.0122	−0.0087	−0.0115	0.0006	−0.0111	−0.0109	0.0021	−0.0086	−0.0078	−0.0091

Once the number of clusters was identified dimensionality reduction algorithm was applied through truncated SVD. Truncated SVD takes n components and transforms the feature matrix such that for 100 <= no. of components <= 1000 and then ASC is computed for each value of n components for 2 <= K <= 26.

It can be observed from Tab. 4 that the ASC decreases as the dimensions of feature vectors increases. Therefore, the best K is selected based on the highest value of ASC, which is found with 100 components with the highest value of 0.074 for K = 26. The plot of ASC for different dimensions of the feature vector is shown in Fig. 2, which depicts that ASC is inversely proportional to the feature dimensions. Therefore, the value of ASC decreases when the feature dimensions are increased.

Table 4: Different dimensions with respect to the number of clusters generated by the K-means algorithm

	No of clusters												
Dim.	2	3	4	5	6	7	8	9	10	11	12	13	14
100	0.031	0.033	0.034	0.037	0.037	0.04	0.042	0.044	0.047	0.051	0.054	0.053	0.056
200	0.019	0.014	0.023	0.023	0.025	0.027	0.028	0.028	0.031	0.033	0.031	0.035	0.037
300	0.015	0.012	0.018	0.017	0.019	0.02	0.021	0.022	0.024	0.023	0.025	0.027	0.025
400	0.012	0.01	0.013	0.014	0.016	0.018	0.018	0.019	0.02	0.021	0.023	0.021	0.023
500	0.011	0.009	0.012	0.011	0.014	0.014	0.017	0.016	0.019	0.017	0.021	0.02	0.021
600	0.01	0.009	0.011	0.012	0.013	0.014	0.013	0.016	0.014	0.015	0.014	0.017	0.016
700	0.01	0.009	0.011	0.011	0.012	0.013	0.012	0.015	0.015	0.015	0.017	0.013	0.016
800	0.01	0.008	0.01	0.01	0.012	0.013	0.014	0.014	0.015	0.011	0.016	0.012	0.013
900	0.009	0.008	0.01	0.01	0.011	0.012	0.013	0.013	0.012	0.015	0.012	0.014	0.013
1000	0.009	0.008	0.009	0.01	0.011	0.012	0.012	0.011	0.013	0.014	0.012	0.013	0.012

	No of clusters											
Dim.	15	16	17	18	19	20	21	22	23	24	25	26
100	0.059	0.062	0.062	0.063	0.064	0.066	0.065	0.067	0.067	0.072	0.07	**0.074**
200	0.038	0.041	0.039	0.04	0.041	0.041	0.041	0.045	0.043	0.045	0.046	**0.048**
300	0.029	0.028	0.031	0.029	0.032	0.033	0.033	0.032	0.034	0.033	0.036	**0.038**
400	0.022	0.025	0.022	0.025	0.024	0.028	0.025	0.009	0.026	**0.03**	0.03	0.025
500	0.019	0.021	0.021	0.017	0.022	0.023	0.024	0.023	0.023	0.022	0.021	**0.025**
600	0.017	0.019	0.018	0.02	**0.021**	0.02	0.021	0.019	0.02	0.021	0.019	0.021
700	0.015	0.016	0.019	0.017	0.017	0.018	0.018	0.018	0.017	0.017	**0.02**	0.02
800	0.016	0.015	0.014	**0.017**	0.015	0.015	0.017	0.015	0.016	0.015	0.017	0.01
900	0.013	0.014	0.015	0.016	0.015	0.015	0.016	0.014	0.017	0.014	**0.019**	0.016
1000	**0.016**	0.013	0.014	0.013	0.015	0.014	0.013	0.015	0.016	0.015	0.016	0.015

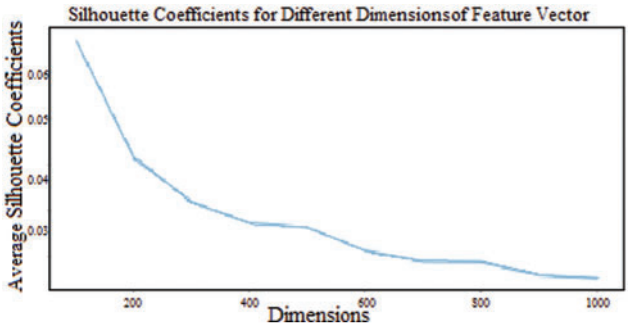

Figure 2: Plot for ASC for different dimensions of the feature vector

Once the clusters were obtained, the neural network was trained with 26 output classes and 100 input neurons. Different settings for hidden layers were used for 10 <= neurons <= 100 with 1 Hidden layer. The training results are mentioned in Tab. 5. It can be observed from the table that the training loss is smallest when the number of neurons is 40 in 1 hidden layer. The training loss starts increasing when the number of neurons is more than 40 in the hidden layer. The best result obtained with 40 neurons is training loss is 0.008, and validation loss is 0.181.

Table 5: Training and validation loss with different number of neurons in first hidden layer with ADAM optimizer

Neuros in hidden layer	Samples	10	20	30	40	50	60	70	80	90	100
Training	13192	0.214	0.0024	0.01	0.008	0.004	0.003	0.002	0.002	0.001	0.002
Validation	5654	0.315	0.182	0.187	0.181	0.194	0.212	0.237	0.229	0.24	0.24

Tab. 6 provides the result of accuracy with the same network settings. Again, it can be observed that accuracy for the neurons is best with 40 neurons in the first hidden layer where loss is negligible primarily, i.e., 0.999 and the validation accuracy is 0.946.

Table 6: Training and validation accuracy with different number of neurons in first hidden layer with ADAM optimizer

Neuros in hidden layer	Samples	10	20	30	40	50	60	70	80	90	100
Training	13192	0.996	0.998	0.998	0.998	0.998	0.998	0.996	0.997	0.994	0.998
Validation	5654	0.888	0.912	0.926	0.924	0.928	0.921	0.926	0.922	0.923	0.924

Based on the results of hidden layer 1, the neural network was trained again with two hidden layers where the neurons in the first hidden layer were 40 and the number of neurons in the second hidden layer was 10 <= neurons <= 100. Tab. 7 shows that the training loss is almost negligible when the number of neurons in the second hidden layer is 30, and the loss starts increasing when the number of neurons is more than 30 in the 2nd hidden layer. The best result achieved with 40 neurons in hidden layer 1 and 30 neurons in hidden layer 2 shows the 0.008 training loss and 0.378 validation loss. Tab. 8 provides accuracy with the same network settings. The accuracy for the neurons is best with 40 and

30 neurons in the first and second hidden layer, respectively, where training accuracy is 0.998, and the validation accuracy is 0.926.

Table 7: Training and validation loss with 40 neurons in first hidden layer and different number of neurons in second hidden layer with ADAM optimizer

Category name	Accomplish	Consider	Location	Medical practitioner	Modify	Move	Take
No. of files	6	40	8	13	21	101	15

Table 8: Training and validation accuracy with 40 neurons in first hidden layer and different number of neurons in second hidden layer with ADAM optimizer

Neuros in hidden layer	Samples	10	20	30	40	50	60	70	80	90	100
Training	13192	0.924	0.998	0.999	0.999	0.999	0.999	1.0	1.0	1.0	0.999
Validation	5654	0.891	0.942	0.942	0.949	0.948	0.948	0.946	0.945	0.94	0.946

Once the training was completed, the proposed system was tested by adding multiple files to the system. Around 204 files of various extensions such as .pdf, .docx, .txt, .pptx on different topics were provided to the system and results were obtained as mentioned in Tab. 9.

Table 9: Category name and user files stored by the proposed system after training

Neuros in hidden layer	Samples	10	20	30	40	50	60	70	80	90	100
Training	13192	0.022	0.008	0.008	0.006	0.007	0.006	0.013	0.012	0.015	0.007
Validation	5654	0.567	0.454	0.378	0.384	0.401	0.461	0.405	0.454	0.471	0.455

It was observed that six files were stored in the folder "Accomplish", where the content of these files included different results of students. Around 40 files were moved in the folder "consider" by the proposed system. Most of these files included a list of students admitted to institutes or other notices for different events. The folder "location" received eight files, and the content of these files was related to maps and geographical details about different places. The folder "medical practitioner" received 13 files, and most of the files included content related to psychology, communication skills or related the field of medical. Around 21 files were moved in folder "modify", where most of the novels were moved to this folder, including adventure stories or science fiction. Around 101 files were moved in folder "move", and all of the files were related to programming, course books of computer science and different articles and research papers on artificial intelligence. Finally, 15 files were moved in folder "take", where most of the files in this folder included content related to programming related slides or code.

These results depict that although the names of different categories are not appropriate due to initial training with dataset of news articles, the number of files stored in different systems is different. However, most of the files having similar content are stored in the same folder. This limitation of having inappropriate names can be dealt with few pieces of training with user-generated files. It is

evident that every user may have files of different natures, so predicting category names is almost impossible. Still, when users start uploading the files in the system, it can automatically learn and update according to the users' requirements. It is important to note that all of this categorization process is entirely automated, which means that users only have to provide files to the system. It is the system's responsibility to recognize the content of the file and manage files accordingly.

Another experiment was conducted with 222 files, including 204 files used in the previous experiment and both global and localize modules were executed. The results obtained after this experiment was exciting. The first thing observed was that the proposed system computed the threshold ASC value for K was three as computed using Eq. (9) for Tab. 10.

Table 10: Average silhouette coefficients with respect to number of clusters generated by k-means algorithm on original feature space dimensions (222 × 24573)

No. clusters	2	3	4	5	6	7	8	9	10
ASC	0.05	0.05	0.016	0.048	0.046	0.043	0.069	0.013	0.017

Based on Tab. 10, the proposed system selected 3 clusters and executed the dimensionality reduction module for several components 100 to 222. However, the number of components for this module cannot exceed the number of documents given for training. As a result, the best value obtained was 0.07 in 100 dimensions with 3 clusters, as mentioned in Tab. 11.

Table 11: Different dimensions with respect to the number of clusters generated by the k-means algorithm

	No. clusters	
Dimension	2	3
100	0.067	0.07
200	0.05	0.05
222	0.05	0.05

The next task was to train a neural network for three output classes. The neural network achieved the best performance with ten neurons in both the first and second hidden layers. However, the maximum accuracy achieved with the validation set was 70% which is very low compared to the first experiment. This depreciation of accuracy was mainly because of the smaller dataset size and the diversity of documents in the given dataset. The 222 files were user-generated files on various topics like course books, novels, programming code, student's marks sheets etc. After training, the proposed system categorized these files into three different folders, as mentioned in Tab. 12. Seventeen files were moved in the folder "consume". In comparison, most of these files had content like a list of students admitted in institutes and students marks sheets etc. 181 files were moved in folder "information" where most of the files in this folder included content related to books on psychology, communication skills, course books of Computer Science particularly books and research papers on artificial intelligence and novels. Twenty-four files were moved in folder "quantity", where all of the files in this folder were programming related slides or code.

Table 12: Category name and user files stored by the proposed system after training

Consume	Information	Quantity
17	180	24

4 Conclusion

In the current digital world, where most of the data available has been digitalized and is available in different files, managing these heterogeneous files is becoming a tedious task. Most computer users create, copy or download tens of files daily and as the number of files on a system grows, it becomes more difficult to arrange these files. The motivation behind this research is to design a system for categorizing files based on their similarities consequently. This research is the first step towards automatic file management based on the content to the best of our knowledge. The results suggest that the proposed system has great potential to automatically categorize almost all of the user files based on their content. The proposed system is completely automated and does not require any human effort in managing the files. As soon as the number of files grows, the predictions of labels and categorization tasks become more efficient. One of the significant limitations of the proposed system is that it can only categorize text-based files. The future versions of this system will incorporate an image, audio and visual files for categorization.

Acknowledgement: Thanks to our families & colleagues who supported us morally.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Murugan, H. Chelsey and N. Thomas, "Cluster analysis: Modeling groups in text," in *Advances in Analytics and Data Science*, Berlin, Germany: Springer, pp. 93–115, 2019.

[2] A. Ruksana and C. Yoojin, "An improved genetic algorithm for document clustering on the cloud," *International Journal of Cloud Applications and Computing*, vol. 8, no. 4, pp. 20–28, 2018.

[3] L. Laxmi, K. Krishna and M. Andino, "Charismatic document clustering through novel K-means non-negative matrix factorization algorithm using key phrase extraction," *International Journal of Parallel Programming*, vol. 46, no. 3, pp. 1–19, 2018.

[4] F. H. Syed and H. Muhammad, "A K-means based co-clustering algorithm for sparse, high dimensional data," *Expert Systems with Applications*, vol. 118, no. 1, pp. 20–34, 2019.

[5] S. Kamal, B. Atanu and S. Patra, "A semantic similarity adjusted document co-citation analysis: A case of tourism supply chain," *Scientometrics*, vol. 125, pp. 233–269, 2020.

[6] H. Andreas, M. Alexander and S. Steffen, "Ontology-based text document clustering," in *Third IEEE Int. Conf. on Data Mining*, Washington, pp. 1–4, 2003.

[7] C. Carlos, M. C. Henry, U. M. Richar, M. Martha, L. Elizabeth *et al.,* "Clustering of web search results based on the cuckoo search algorithm and balanced Bayesian information criterion," *Information Sciences*, vol. 281, no. 1, pp. 248–264, 2014.

[8] C. Claudio, O. Stanislaw, R. Giovanni and W. Dawid, "A survey of web clustering engines," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–38, 2009.

[9] B. Nadia and A. Francisco, "Cluster validation techniques for genome expression data," *Signal Processing*, vol. 83, no. 4, pp. 825–833, 2003.
[10] P. Kandasamy and M. N. A., "Hybrid PSO and GA models for document clustering," *International Journal of Advanced Soft Computing Applications*, vol. 2, no. 3, pp. 302–320, 2010.
[11] S. Pierre, K. Koray, C. Soumith and L. Yann, "Pedestrian detection with unsupervised multi-stage feature learning," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Washington, pp. 1–7, 2013.
[12] J. D. Dinneen and C. A. Julien, "The ubiquitous digital file: A review of file management research," *Journal of the Association for Information Science and Technology*, vol. 71, no. 3, pp. 23–28, 2019.
[13] C. Jerome, M. Jonathan and R. Michele, "File naming in digital media research: Examples from the humanities and social sciences," *Journal of Librarianship and Scholarly Communication*, vol. 3, no. 3, pp. 1260–1269, 2015.
[14] C. John, "Creative names for personal files in an interactive computing environment," *International Journal of Man-Machine Studies*, vol. 16, no. 4, pp. 405–438, 1982.
[15] H. Ben, D. Andy, R. Palmer and M. Hamish, "Organizing and managing personal electronic files: A mechanical engineer's perspective," *ACM Transactions on Information Systems*, vol. 26, no. 4, pp. 23–29, 2008.
[16] L. Edward and B. Steven, "NLTK: The natural language toolkit," in *ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, Philadelphia, pp. 1–9, 2002.
[17] B. Yungcheol and L. Yillbyung, "Form classification using DP matching," in *ACM Symp. on Applied Computing*, New York, pp. 1–5, 2000.
[18] B. Andrew and W. Marcel, "Fine-grained document genre classification using first order random graphs," in *Proc. of Sixth Int. Conf. on Document Analysis and Recognition*, Seattle, pp. 23–26, 2001.
[19] K. J. Anil, M. Narasimha and F. Patrick, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 1–69, 1999.
[20] A. Aizawa, "An information-theoretic perspective of TF–IDF measures," *Information Processing and Management*, vol. 39, no. 1, pp. 45–65, 2003.
[21] H. John and W. Marianne, "Algorithm as 136: A K-means clustering algorithm," *Journal of the Royal Statistical Society*, vol. 28, no. 1, pp. 100–108, 1979.
[22] S. Yang, M. Lin and T. Xuezhi, "Weakly supervised class-agnostic image similarity search based on convolutional neural network," *IEEE Transactions on Emerging Topics in Computing*, vol. 22, pp. 13–18, 2022.
[23] P. Chen, L. Leida, W. Jinjian, D. Weisheng and S. Guangming, "Contrastive self-supervised pre-training for video quality assessment," *IEEE Transactions on Image Processing*, vol. 31, pp. 458–471, 2021.
[24] G. Ross, D. Jeff, D. Trevor and M. Jitendra, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, Columbus, pp. 12–16, 2014.
[25] H. Vincent and S. Smail, "Automatic generation of ontologies : Comparison of words clustering approaches," in *15th Int. Conf. Applied Computing*, Budapest, 2018.
[26] P. Martin, "Snowball: A language for stemming," 2001. [Online]. Available: http://snowball.tartarus.org/texts/. [Accessed 15 February 2021].
[27] A. Farzana, S. Samira and S. Bassant, "Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)," *Computer Science Review*, vol. 40, pp. 54–67, 2021.
[28] S. Michael, K. George and K. Vipin, *A Comparison of Document Clustering Techniques*, Minnesota: University of Minnesota, pp. 45–52, 2013.
[29] K. R. Rajendra, "An effective approach for semantic-based clustering and topic-based ranking of web documents.," *International Journal of Data Science and Analytics*, vol. 5, no. 4, pp. 269–284, 2018.
[30] T. Lorenzo, S. Martin and F. Andrew, "Efficient object category recognition using classemes," in *11th European Conf. on Computer Vision*, Crete, pp. 26–33, 2010.

[31] W. Shenghui and K. Rob, "Clustering articles based on semantic similarity," *Scientometrics*, vol. 111, no. 2, pp. 1017–1031, 2017.
[32] P. K. Diederik and L. B. Jimmy, "ADAM: A method for stochastic optimization," in *Int. Conf. on Learning Representations*, San Diego, pp. 43–48, 2015.
[33] "20 newsgroups data set," 2006. [Online]. Available: http://people.csail.mit.edu/jrennie/20Newsgroups/. [Accessed 10 February 2021].