Mindanao Journal of Science and Technology Vol. 19 (1) (2021) 17-39

Incorporating Rule-based Pattern Recognition Approach for Document Structure Classification on Cloud-based Document Management System

Marisa M. Buctuanon*, John Leeroy A. Gadiane,
Franc Anthony Margallo and Paul Ryan Lucero
School of Computer Studies
University of San Jose - Recoletos – Basak Campus
Cebu City, 6000 Philippines
**marisamahilum@gmail.com*

Date received: August 4, 2020
Revision accepted: June 2, 2021

Abstract

Numerous documents in the form of emails, business letters, reports and transactions among others are created and received by many organizations. Thus, organizing them is a challenge. There are various ways to organize documents such as according to the header, sender, or content. However, following the rules in organizing documents might not be consistent with organizations like schools as this may vary from one person to another and is prone to human errors. The manual organization also requires a lot of time and may lead to difficulty in finding the documents' location. Thus, this study aimed to develop an intelligent document organizing system named Docudile. It is a self-organizing system that classifies each document and seamlessly places them in the computer directory using the rule-based pattern recognition for quick and accurate locating of documents. A cloud-based document management system with storage that syncs documents from local storage to cloud server was also developed to mitigate the inaccessibility of the documents when they are accessed from a remote area. Term Frequency-Inverse Document Frequency (TF-IDF) was used to retrieve the documents. Results showed that the system yielded 98 and 89% accuracy in classifying and retrieving the documents, respectively, based on the rule-based pattern recognition. Compared with Naïve Bayes Classifier and support vector machine accuracy results, it was found that using cosine feature similarity of the rule-based pattern recognition obtained a better accuracy in classifying school-related documents. Furthermore, this study recommends that supporting documents should go with the main document during the classification.

***Keywords**: automatic document classification, cloud-based document management system, document structure classification, rule-based pattern recognition*

1. Introduction

Documents serve as means of communication between and within organizations. They provide a way for organizations to give official statements, standardize business processes and show business progress, proofs of a transaction and transparency reports to name a few. A document is the lifeblood of every business operation. However, documents tend to go missing or are hard to find when there is no rule to follow to organize them in digital storage. This kind of document mismanagement may result in business failure. Integrating automatic document classification (ADC) on a file management system would increase productivity between organizations and decrease mismanagement risks (Goller *et al*., 2000). It labels documents based on their content vis-à-vis to one or more predefined categories (Chagheri *et al*., 2011). The document's properties, also known as metadata such as title, author, name, subject and keywords that identify its topic or content, reflect its categorizations and use (Dourish *et al*., 2000). Augmenting a rule-based framework for document structure classification that uses the document properties enables the provision of an improved ADC. Another bottleneck of document retrieval is when the document location is inaccessible at any time of the day. The document might be stored somewhere else, and when there is an urgent need to retrieve it, nothing can be done. Hence, finding a pattern to follow in automating the document categorization and retrieval in online storage was the direction of the present study.

The document structure is defined as the organization of a document into graphical constituents like sections, paragraphs, sentences, bulleted lists and figures (Power *et al*., 2003). This structure is the layout of a document containing a significant amount of information that can be used to classify it by type (Shin *et al*., 2001; Gulin and Frolov, 2016). Classifying the document based on its structure has given interest to many researchers. Chagheri *et al*. (2011) proposed an approach using structural elements of the document and not just its content. The document structure used the tags of the XML document. They utilized support vector machines (SVM) algorithm to classify the document. However, their study focused only on XML documents and not on scanned documents, which are commonly used by many organizations. Hence, the scanned document was emphasized in this study. Similar to the present work is the study of Dengel and Dubiel (1995), which described a system that is capable of learning the presentation of business documents' logical structures. Their system clusters the documents into structural concepts and induces a concept of hierarchy taken as a source for classifying future

input. The study of Al-Sabahi *et al*. (2018) introduced a different way of creating the concept of hierarchy that is applied for logical labeling.

In the present study, rule-based pattern recognition has been introduced, which simplifies the classification process. The documents did not undergo a training process as this study utilized the structure of the document to create a rule-based pattern recognition system. Thus, it eliminated the overhead costs.

The rule-based approach, which was employed in this study, has a knowledge part represented by production rules which consist of IF condition and THEN action or conclusion (Shaw and Fifer, 1986; Henderi *et al*., 2020). With the rule-based approach added on pattern recognition, these rules must provide uniqueness of the form feature definition of documents. If set up correctly, an accurate and thorough form feature identification is achieved (Babic *et al*., 2008). In pattern recognition, numerous systems differ from where a form feature or a specific pattern is identified. One of which is syntactic pattern recognition, wherein a set of a grammar containing some rules defines a particular pattern (Babic *et al.*, 2008). In the early 1960s and next two decades, a syntactic approach was successfully applied by Flasiński and Jurek (2014) for structurally-oriented recognition (SOR) problems. This proves that document structure classification – a type of SOR – can be solved through pattern recognition. However, this method has not been extensively explored. Most studies focused on document classification through the Bayesian approach, which ignores the document structure and concentrates more on the content of the document.

This study aimed to show how a rule-based approach on pattern recognition is used to classify the document based on its logical and content structure. The logical structure is composed of the document's heading and body sections. The content structure refers to the title, date and other functional zones of the document (Stede and Suriyawongkul, 2010). Also, this study exhibits a cloud-based document management system (CDMC), where the documents in the form of images are stored in and retrieved from a cloud storage platform (CSP). Dropbox was utilized as the system's CSP since it is fast syncing. That is to say, the system offers more than document classification and cloud storage. It can sync local documents to Dropbox and retrieve documents based on their content or filename. This study's utilization of the result of a rule-based pattern recognition to generate a directory structure contributes to the many uses of rule-based pattern recognition. This approach may be simple; however, it gives a practical way to properly store and easily retrieve

documents. This study also amplifies the reasons to use a rule-based pattern to classify school-related documents.

2. Methodology

Docudile is the name of the system developed in this study. It is made using Java Enterprise Edition, specifically, utilizing the Spring Framework and Hibernate Object/Relational Mapping (ORM). The framework enables the developer to program and configure the model without unnecessary ties to a specific deployment environment while ORM provides ease in defining and manipulating the database. Using the Dropbox SDK (Dropbox for Java Developers, n.d.), the system can access and manipulate the Dropbox account through user authentication. The system also integrates ABBYY® Cloud OCR SDK (Cloud OCR SDK, n.d.) for optical character recognition (OCR).

This study had four major phases in incorporating a rule-based pattern recognition approach for document structure classification: 1) image to text conversion, 2) data preprocessing, 3) document structure generation and 4) directory structure generation. Each phase is discussed further in the succeeding sections. Additionally, managing the documents in the cloud, which includes the syncing of documents from local storage to the cloud server of Docudile and retrieving the document using Term Frequency-Inverse Document Frequency (TF-IDF), is likewise elaborated.

2.1 Image to Text Conversion

Although digitalizing documents is recommended nowadays, many organizations still produce hard copies of documents to communicate with other organizations and within their premises. To save them digitally as an image, the user needs to scan the document first and upload it to the system. Figure 1a shows a sample of a scanned document uploaded to the system. The image undergoes OCR to convert image to text. This is not yet the pattern recognition; this phase only allows the system to process the document in a text format instead of an image.

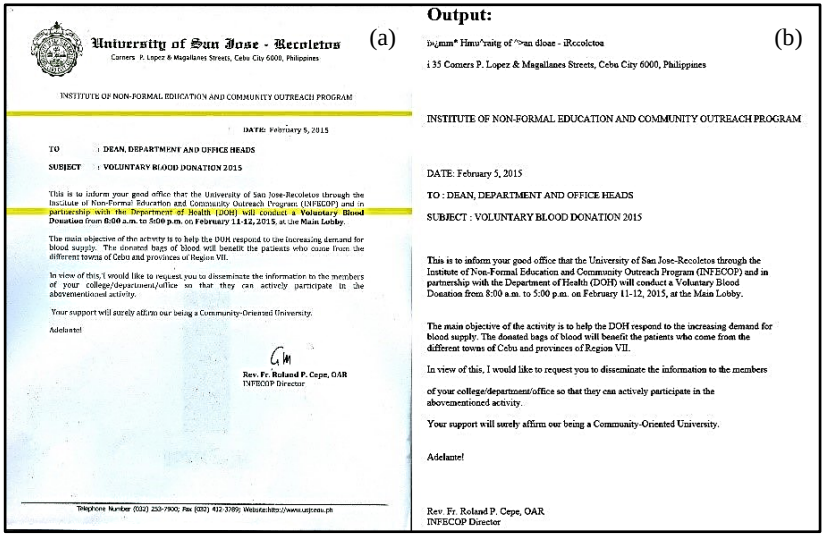

Figure 1. Sample output of the OCR API

There are several options available for developers when using the API, namely setting the language for recognition, font style used and image source. Since most of the documents are in English, the language setting was set into this language format. Knowingly, most of the documents are captured using flatbed scanners. Therefore, the source was set to "scanner." These settings are important because they aid the API in recognizing the documents correctly. The API returns the result in a single string format with only the carriage return and the new line as the separator between the lines. The system formats the result as shown in Figure 1b.

2.2 Data Preprocessing

Data preprocessing is a data mining technique that involves data cleaning, normalization, transformation and several other preparations to produce correct and useful data for further processing (García *et al*., 2015). In this phase, the OCR result undergoes removal of all characters that are identified as alphanumeric and whitespace. Removal of the occurrence of stop words then follows because these words are unrelated to the actual document. After which, the system identifies whether the words exist in the dictionary using Java WordNet Library (JWNL) (Extended Java WordNet Library, n.d.). Finally, stemming or acquiring the words' base forms (Singh & Gupta, 2017) is then applied through the same library. The system calls stemWordwithWord, a method in JWNL that is responsible for returning the

stemmed equivalent of the word. Table 1 exhibits the data preprocessing result shown in Figure 1. For sample demonstration, the content of the body shown in this table is only from the first paragraph of the document.

Table 1. Data preprocessing result

Heading
INSTITUTE OF THE NON-FORMAL EDUCATION AND COMMUNITY OUTREACH PROGRAM Date: February 5, 2015 Dean, Department Office Heads Subject: Voluntary Blood Donation 2015
Body
inform good office University San Jose-Recoletos Institute Non-Formal Education Community Outreach Program (INFECOP) partnership Department Health conduct Voluntary Blood Donation 8:00 a.m. 5:00 p.m. February 11-12, 2015, Main Lobby.

The basis on what pattern to follow is the logical and content structure of the documents. It is observed that the heading part of the document contains vital information as to how the documents are stored physically or digitally. The heading structure of the documents in many organizations, particularly schools, is either <OFFICE, DATE, TO, [THRU], SUBJECT> or <OFFICE, DATE, SALUTATION>, which serves as rules in classifying the documents. To further understand how these rules are applied to the system, the document structure generation section provides the process.

2.3 Document Structure Generation

This phase is also considered the information extraction (IE) process of the system in which the automatic extraction of structured information from unstructured sources is done (Sarawagi, 2007). The system only extracts the essential information of the document such as date, sender, recipient and subject.

2.3.1 Regular Expression

To recognize the structure of the document and identify the tag of each line in the heading, the system uses regular expression (regex). Regex is a pattern that matches one or more strings of characters (Briiggemann-Klein, 1993; Lee *et al.*, 2016). Regex was implemented to form a feature definition of a document and verify whether it has any match in the predefined pattern or rule. Table 2 illustrates the result of the regex process undergone by the heading section of

the document. Here, the system evaluates each line's tag as OFFICE, DATE, TO and SUBJECT.

Table 2. Regular expression result

Heading	
Text string	Tag
INSTITUTE OF THE NON-FORMAL EDUCATION AND COMMUNITY OUTREACH PROGRAM	OFFICE
Date: February 5, 2015	DATE
Dean, Department Office Heads	TO
Subject: Voluntary Blood Donation 2015	SUBJECT

Figure 2 displays the result done through the regex, where tags are labelled in each part of the heading.

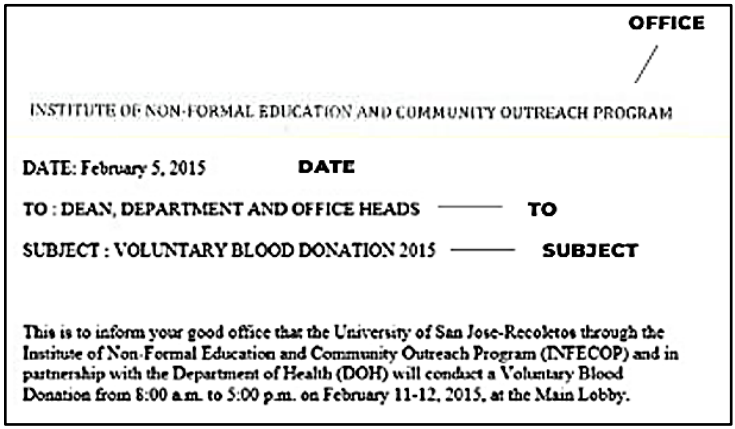

Figure 2. Regular expression result

Once the system recognizes the pattern of the document based on the result of the regular expression, the system then identifies whether the document is a memo, letter, or unclassified.

2.3.2 Feature Similarity Computation

The next process is to do feature similarity. In this study, feature similarity means finding how much a document is considered either a memo, letter, or unclassified using feature similarity computation. The IF-THEN rule-based pattern is applied to do the classification. If the document structure is <OFFICE, DATE, TO, [THRU], SUBJECT>, then the document is a memo. If the document contains <OFFICE, DATE, SALUTATION>, then it is

classified as a letter. Again, these patterns are observed every time a memo or letter is written or received by a school organization. These two types of documents are the only focus of the IF-THEN rules since they are most commonly used in academic institutions. Any document having none of these rules is considered unclassified. Unclassified documents do not have logical and content structures, which can only contain images or tables. Further discussion on unclassified documents is presented in the directory structure generation section.

Table 3 displays the result of feature similarity. The system employs Equation 1 to compute the feature similarity of the document to the two predefined structures, which are <OFFICE, DATE, TO, [THRU], SUBJECT> for a memo and <OFFICE, DATE, SALUTATION> for a letter. The tag that is enclosed in the square bracket means optional. After computation, Table 3 shows that the scanned document is classified as memo.

$$Feature\ similarity = \sum_{i=1}^{n} t_i \tag{1}$$

where *t* denotes the tags generated from the document from *i* to *n* or the total number of tags.

Table 3. Feature similarity result

FS	Tags
MEMO = 4	OFFICE, DATE, TO, SUBJECT
LETTER = 2	OFFICE, DATE

2.4 Directory Structure Generation

The document structure generation determines whether to generate a directory structure of the document or classify the document from the existing directory. If the directory structure of the document is already generated, the system traverses the said directory, where the document is then stored.

The tags that were acquired earlier during regex are employed in this process. These tags are the bases to form the hierarchical structure of the documents. The hierarchical structure is also referred to as the directory structure of the documents. Figure 3 is the result of the generated directory structure of the document.

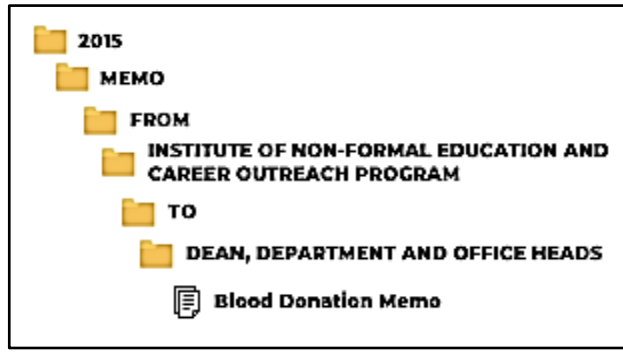

Figure 3. Hierarchical structure

The unclassified documents, including, but not limited to certificates and research, are stored in the "UNCLASSIFIED" directory. They are unclassified since they do not contain the pattern for a memo or a letter. However, they are still stored which can be also be used for document search and retrieval operations. Furthermore, the system performs an additional set of operations to manage the documents. The succeeding sections present the syncing of documents from local storage to the cloud and retrieving them through TF-IDF.

2.5 Local to Cloud Storage

The user can directly upload the documents to the cloud. However, if the internet is slow or fluctuating, uploading them to the cloud takes time. To resolve this, the system has the capability of syncing the local documents to the cloud in the background. Using Docudile, the user can upload the scanned documents and allow the system to store them locally in the meantime. Once the machine has an internet connection, the system performs the syncing of documents to the user's Dropbox account. By default, the system syncs the documents every 5 min using a jQuery script. The user can also manually trigger the syncing. The jQuery script has a setTimeout method, which calls a function every 5 s to check if it is already time to sync the documents to the cloud. The system acquires a list of all directories and documents with their corresponding metadata from the cloud storage. The metadata includes the storage path, file name, upload date and username of the uploader. Since multiple users can use one cloud account in an organization's particular office, the system logs this information for future inventory. However, the system can only retrieve all directories and metadata during the scheduled time for syncing. Beyond that, the system does not retrieve the local changes. The

system compares its own saved list with the current state of the local storage when syncing occurs. The system deletes any document in the cloud that are unfound in the local storage. It also overwrites the documents in the cloud with a timestamp that is behind the timestamp of the equivalent document in the local storage. Afterward, the system only syncs documents that are not present in the cloud. This optimized process helps the system avoid redundancy and the heavy load in document syncing.

2.6 Document Search and Retrieval

The system can also handle the document management system's default operations like the store, replace and delete. These functionalities were taken care of by the default libraries bundled with Java and the I/O library of Apache. One of the most relevant features of this system is the document search and retrieval functionality. The study utilized the TF-IDF algorithm to capture the need for easy and fast retrieval of documents upon searching. TF-IDF shows the relevance of the search keys to the documents. It calculates the values for each word in a document through an inverse proportion of the word frequency in a document to the percentage of each word that appears (Ramos, 2003). Here, the search and retrieval do not only find a document based on its filename but also its contents. These features involve two phases: generating of document index and actual searching of the document. The first phase happens during the uploading of the document. This computes the IDF values of each index. The TF (term, document) is computed by counting the number of times the term appears in a document. After calculating each index's IDF values using Equation 2 and its TF, the TF-IDF values are then obtained using Equation 3. Table 4 illustrates the sample bag of words present in the database, their IDF values, term frequency values, and TF-IDF values. The TF-IDF values of each index are stored in the database to optimize the search process. After calculating the TF-IDF values, the document length is also calculated (Equation 4) since it plays a vital role in the search process.

$$IDF(term) = \log_e\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ in\ it}\right) \quad (2)$$

$$TF\text{-}IDF(term, document) = TF(term, document) \times IDF(term) \quad (3)$$

Table 4. TF-IDF calculation

Term T	Blood	University	Donation	Student	Emergency
IDF (term)	3.9657	0.3945	6.9657	5.211	1.3219
TF (term, document)	1	1	1	0	2
TF-IDF (term, document)	3.9657	0.3945	6.69657	0	2.6438

$$Document\ length\ (document) = \sqrt{\sum_{t \in d} TF\text{-}IDF(term, document)^2} \quad (4)$$

The next phase is for the user to look for a particular document. Supposing the user inputs "Blood Donation Student." The input undergoes a tokenization process where each term represents t in Table 5. The term frequency of each term is calculated using Equation 5. After getting the term frequency, the TF-IDF value of each search term is calculated (Equation 3). Finally, the search length is attained using Equation 6.

Table 5. Term frequency result

Term T	Blood	Donation	Student
TF (term)	0.33	0.33	0.33
Search TF-IDF (term)	1.3219	2.3219	1.737
Search Length		3.186831254	

$$TF(terms) = \frac{No.\ of\ times\ term\ t\ appears\ in\ the\ search\ query}{Total\ number\ of\ terms\ in\ the\ search\ query} \quad (5)$$

$$Search\ length\ (search) = \sqrt{\sum_{t \in s} TF\text{-}ID(t)^2} \quad (6)$$

After calculating the actual search length, the system uses the document length, the TF-IDF values of the document indices and TF-IDF values of the search terms to compute each document's score relative to the search terms, using the cosine similarity measure (Equation 7). The higher the cosine similarity value, the higher is the relevance of the document to the search

terms. Table 6 shows a sample result of TF-IDF, document length and cosine similarity computation of five documents. These documents are ranked based on the cosine similarity.

Table 6. TF-IDF, document length and cosine similarity scores

Documents	TF-IDF			Document length	Cosine similarity value
	Blood	Donation	Student		
2-7-15 Voluntary Blood Donation 2015.jpg	5.28771238	4.64385619	0	25.73779761	0.216681766
06-09-15 Appointments. jpg	0	0	1.736965594	24.39067252	0.038815
06-15-15 Appointments. jpg	0	0	0	30.43764674	0
06-29-15 Excuse of CAS Students.jpg	0	0	3.473931188	23.5164332	0.080515948
08-14-15 Meeting of University Days and Intramurals.jpg	0	0	0	24.11034345	0

$$Cosine\ similarity\ (document, search) = \frac{TF\text{-}IDF\ (term, document) \times TF\text{-}IDF(term, search)}{documentLength\ (document) \times searchLength\ (search)} \quad (7)$$

Before the system retrieves the documents with a TF-IDF score greater than 0, the system also checks if the search terms exist in the document as a phrase.

Table 7 shows a sample result after checking whether the documents contain the search word “Blood Donation Student”. In this example, only 2-7-15 Voluntary Blood Donation 2015.jpg has a phrase that matches the search phrase. 2-7-15 Voluntary Blood Donation 2015.jpg will have higher relevance in the document ranking. To do this, the final TF-IDF score of 2-7-15 Voluntary Blood Donation 2015.jpg is added with 1, resulting in a TF-IDF value of 1.216681766.

After calculating the TF-IDF scores of each document, the system sorts the values in descending order. This results in 2-7-15 Voluntary Blood Donation 2015.jpg as the most relevant. Figure 4 displays the search result.

Table 7. Search phrases result

Documents	Search phrases		
	"Blood donation"	"Donation student"	"Blood donation student"
2-7-15 Voluntary Blood Donation 2015.jpg	1	0	0
06-09-15 Appointments.jpg	0	0	0
06-15-15 Appointments.jpg	0	0	0
06-29-15 Excuse of CAS Students.jpg	0	0	0
08-14-15 Meeting of University Days and Intramurals.jpg	0	0	0

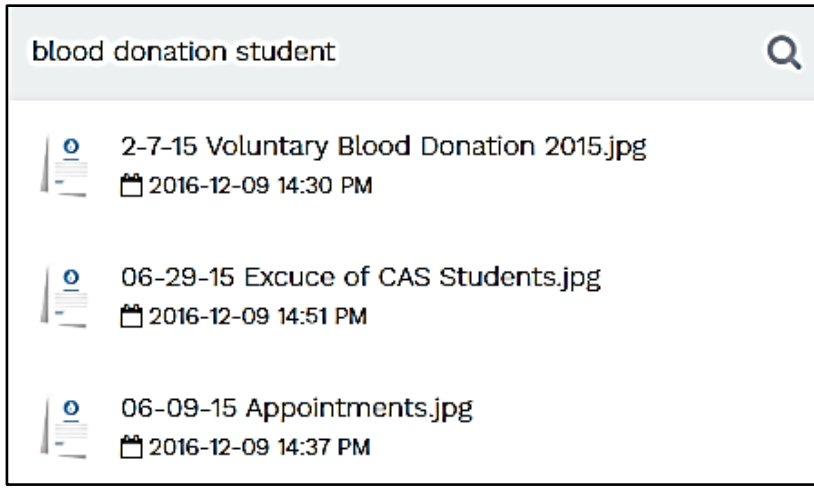

Figure 4. Document search and retrieval result

3. Results and Discussion

3.1 User Interface

The succeeding sections illustrate the different user interfaces interacted by the user when navigating through Docudile. For the user to access the documents uncompromisingly, the files are saved in cloud storage by using Dropbox. At the registration phase, the user needs to allow the system to sync the documents to the cloud. Clicking the button shown in Figure 5 redirects

the user to the authentication page of Dropbox, asking permission if the system can access the user's Dropbox account to manage documents.

Figure 5. Cloud authentication

Figure 6 displays the user's information such as name, email address and organization to where the user belongs. The information shown is merely for demonstration purposes; it does not reflect any official organization. Figure 6 also presents the number of documents stored and its storage cloud size.

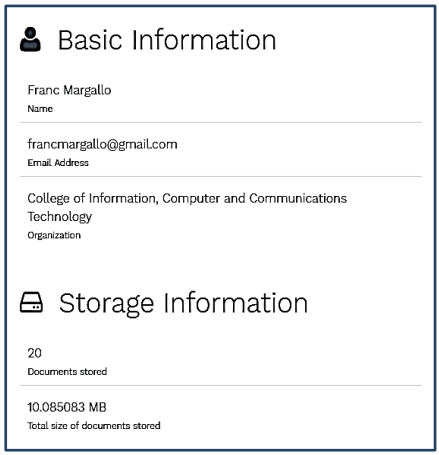

Figure 6. Basic information

Figure 7 serves as the uploading area of the user. The user can either click the upload area or drag and drop scanned documents to the upload panel. The middle portion of Figure 7 has been magnified to clearly illustrate the different statuses of the files during and after uploading. The status could be pending, processing and complete. The user knows if the uploading is complete when the file path of the scanned document is displayed.

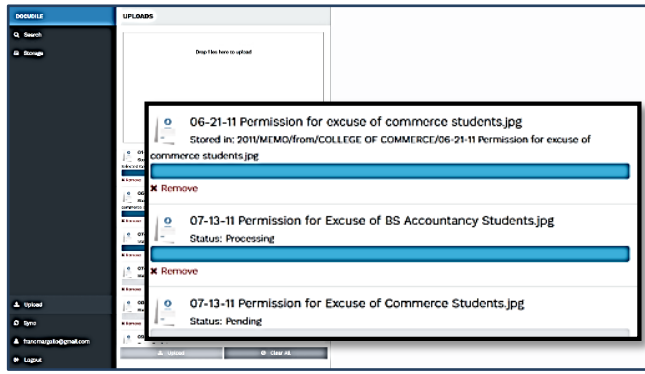

Figure 7. Upload page

Figure 8 is the main page shown to the user once the user has successfully logged in to the system. The document viewer page shows the different folders containing the documents uploaded directly to the system and those documents synced into the user's Dropbox account. The leftmost pane shows a tree view, which serves as the hierarchical structure or the directory structure of document storage. The user can navigate through it by clicking the expand-and-collapse icons of the folders. Additionally, the system also provides functionalities like downloading and deletion of documents.

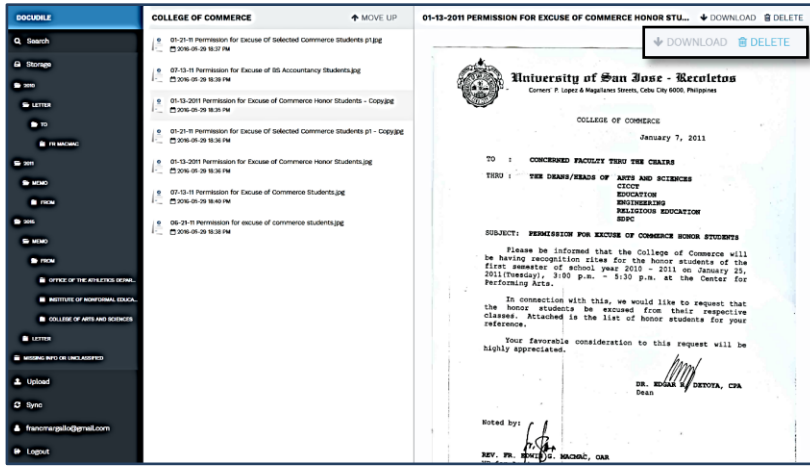

Figure 8. Document viewer page

Figure 9 shows a sample of search results. In this example, the user is looking for an excuse letter from students in a particular department. The user can search the documents by entering keywords or terms associated with the

documents. The left side shows the search result while the right-side pane displays the preview of the selected document.

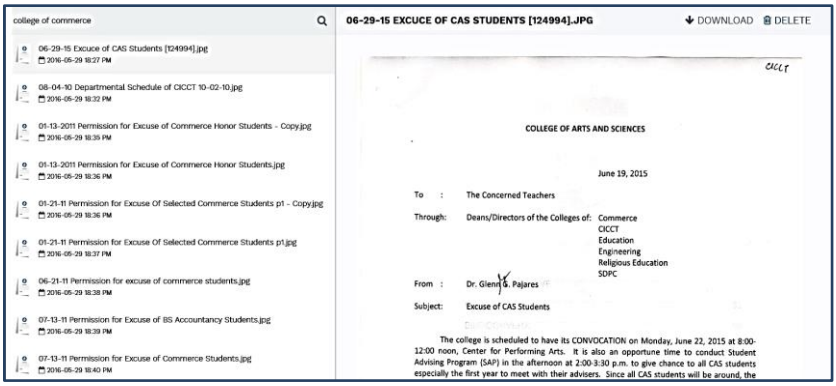

Figure 9. Search result

3.2 Accuracy Testing

The accuracy testing focused on the correctness of the generated hierarchical structure and the search result. The extracted information and the result of the classification dictate the identification of the hierarchical structure of the document. Any failure in the aforementioned process results in missing details or unclassified documents.

3.2.1 Rule-based Pattern Recognition

There were 198 scanned memos, 172 scanned letters and 30 unclassified documents in the form of images. They were randomly selected and shuffled in every test. Each test contained memos, letters and unclassified ones. As shown in Table 8, there were 196 letters classified as letters, 165 memos as memos, and 30 unclassified documents as unclassified. However, two letters and seven memos were unclassified. Upon inspection of these documents, stamps were seen on the heading part of the document, which disrupted the OCR result and caused the system not to detect the expected structure. Exposure to dirt also makes the scanned image blurry. There was no problem with the unclassified documents since documents that did not follow the hierarchical structure defined by the system were automatically labeled as unclassified. Some unclassified documents contained a title and a table only. However, some of them were supporting documents to memos and letters. Getting the average of the accuracy result of the letter, memo and unclassified

documents, the system yielded an accuracy test result of 98% in classifying the documents based on the rule-based pattern recognition.

Table 8. Rule-based pattern recognition accuracy test result

		Predicted label		
		Letter	memo	Unclassified
True Label	Letter	196 (99%)	0	2 (1%)
	Memo	0	165 (96%)	7 (4%)
	Unclassified	0	0	30 (100%)

The study also conducted accuracy testing with Naïve Bayes Classifier and Support Vector Machine (SVM). These are two known machine learning (ML) algorithms that can classify documents. In this case, the hierarchical structure's generation was ignored since these two ML algorithms' goal was to classify whether a document was a letter or memo based on its content only. The study utilized the scikit-learn library in Python to do the classification. Figures 10 and 11 show the sample code conducted for both ML algorithms. The 70-30 rule was used to split the dataset of 198 scanned memos, 172 scanned letters and 30 unclassified documents. The splitting of the dataset was not used in the rule-based pattern recognition since the method only needed the predefined rules to classify the documents. The skipping of the training process hastened the overall process of this study, which is an advantage of the rule-based pattern recognition. The accuracy results of the Naïve Bayes Classifier and SVM were 78 and 86%, respectively (Figures 10 and 11). From these results, it can be inferred that the rule-based pattern recognition classifies better than these two ML algorithms. The content of the documents, which was the bases of training and testing for both ML algorithms, was not enough to classify these documents. Since these documents were almost identical except for their structure, the use of the rule-based pattern in this study is highly recommended.

```
from sklearn.pipeline import Pipeline
text_clf = Pipeline([('vect', CountVectorizer()),('tfidf', TfidfTransformer()),('clf', MultinomialNB()),])
text_clf = text_clf.fit(train_data, train_target)
text_clf
```

```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', MultinomialNB())])
```

```
import numpy as np
predicted = text_clf.predict(test_data)
np.mean(predicted == test_target)
```

```
0.7777777777777778
```

Figure 10. Naïve Bayes Classifier accuracy result

```
from sklearn.linear_model import SGDClassifier
text_clf_svm = Pipeline([('vect', CountVectorizer()),('tfidf', TfidfTransformer()),('clf-svm', SGDClassifier(loss='hinge', p
_ = text_clf_svm.fit(train_data, train_target)
predicted_svm = text_clf_svm.predict(test_data)
np.mean(predicted_svm == test_target)
```

```
0.8571428571428571
```

Figure 11. SVM accuracy result

3.2.2 Document Search and Retrieval

The sample search inputs for accuracy checking of document search and retrieval are shown in Table 9. Different inputs were taken into consideration including, but not limited to, filename, segment of the filename and phrase or word from a document. The expected and actual results are scanned documents with file extensions png, jpg, and jpeg files. Because of the dynamic input, the actual result may not be necessary the first document that appears in the search result. However, all of the expected documents are ranked higher than those documents that are irrelevant.

Table 10 exhibits the precision and recall of 400 documents stored in Docudile. These documents have already undergone the TF-IDF process during upload, which has been mentioned in the methodology section. The result of the TF-IDF process was saved in the database for fast search and retrieval of documents. Every time a new document is added to the system, the system updates the TF-IDF values stored in the database. The search keywords were calculated on the fly to find the relevant documents using cosine similarity. There were two test cases conducted in this study as shown in Table 10. T1 used the filename or substring of the filename as search keywords. T2 utilized the content of the document as search keywords. For each search keyword, a maximum of 10 documents was displayed to the user. Table 10 displays the search and retrieval result with precision, recall and F1 scores. There were 40 search keywords for every test case. Every search input returned relevant documents, non-relevant documents and total relevant documents stored in the database. Precision was used to get the ratio of the relevant returned documents to the total returned documents. The recall was utilized to get the ratio of the relevant returned documents to the total relevant documents stored in the database.

Table 9. Accuracy test result of document search and retrieval

Test no.	Search keywords	Expected	Actual	Result
T1	CAS Student World of Music 2015	2-7-15 CAS Student 2015.jpg	2-7-15 CAS Student 2015.jpg	Correct
T2	Enrollment 2nd semester 2021	Enrollment Duty.png	Enrollment Duty.png	Correct
T3	2nd Sem 2020 class roster	Class of 2nd Sem 2020.jpeg	Class of 2nd Sem 2020.jpeg	Correct
T4	Grading System of CICCT	Grades.png	Grades.png	Correct
T5	CS Elec1 Change of Grade	Change of Grade.png	Change of Grade.png	Correct
T6	Business Permit of Startup 2018	06-15-15 Appointment.jpeg	06-15-15 Appointment.jpeg	Correct
T7	Demonstration on programming simulation	Business.jpg	Business.jpg	Correct
T8	Letter to the Vice-President 2010	Letter to the Vice-President.jpg	Letter to the Vice-President.jpg	Correct
T9	Curriculum Vitae Marisa Buctuanon	CV.jpeg	CV.jpeg	Correct
T10	Office of the President 2020	Letter from the President.jpg	Letter from the President.jpg	Correct
T11	Important Letter corresponding to the health protocol of USJ-R	A response letter.jpeg	A response letter.jpeg	Correct
T12	Memorandum Agreement with Google	12-2-18 Meeting with Google.png	12-2-18 Meeting with Google.png	Correct
T13	Trust Fund for the school year 2019-2020	Fund Release.jpg	Fund Release.jpg	Correct
T14	Handbook Manual for Employees	06-2020 Handbook for Employees.jpg	06-2020 Handbook for Employees.jpg	Correct
T15	Financial Statement of CICCT	2020 Financial Statement.png	2020 Financial Statement.png	Correct
T16	Vacation Leave of John Leeroy Gadiane	IT Department File Leave.jpg	IT Department File Leave.jpg	Correct
T17	Accenture Online Job Recruitment	OJR.jpeg	OJR.jpeg	Correct
T18	Non-disclosure Agreement with RITTC	Non-disclosure Agreement with Company A.jpg	Non-disclosure Agreement with Company A.jpg	Correct
T19	Employee Training for Research	10-1-17 Training.jpg	10-1-17 Training.jpg	Correct
T20	Discussion with the Attorneys and the members of the panel	08-14-15 Meeting for Tuition Fee Increase and Other Matters. jpg	08-14-15 Meeting for Tuition Fee Increase and Other Matters. jpg	Correct

To compute the precision, recall and F1 scores of the result, Equations 8, 9 and 10 were used, respectively. F1 score determined the accuracy of the TF-

IDF implementation of the system. F1 score is more useful when there is an uneven document vis-à-vis the search input. On average, the system was able to return 89% of the relevant documents. The major contributing factor to this was the number of relevant documents returned by the system. Since the system displays only the top 10 relevant documents, not all relevant documents are shown to the user.

Table 10. TF-IDF precision and recall summary

Test no.	No. of search keywords	Total relevant documents returned	Total Non-relevant documents returned	Total relevant documents in database	Precision	Recall	F1 Score
T1	40	400	0	504	100%	84%	91%
T2	40	72	28	496	93%	81%	86%
Average					97%	82%	89%

$$Precision = \frac{Relevant\ documents\ returned}{Relevant\ document\ returned + Non\text{-}relevant\ documents\ returned} \quad (8)$$

$$Recall = \frac{Relevant\ documents\ returned}{Total\ relevant\ documents\ in\ the\ database} \quad (9)$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (10)$$

3.3 Performance Testing

Testing the performance of the main features of the system really matters for the user to ensure its efficiency. Table 11 shows the average user time, CPU time and system time. User time is the time spent by the user navigating through the feature. CPU time is the amount of time executed to perform the instructions for the feature. System time is the operating system's time for running or executing the feature. There were 10, 25 and 50 documents fed for each feature. Each time spent per batch of documents is recorded. Table 11 presents the average result of the testing.

Uploading a new document to the system undergoes different processes, and it depends on the machine where the system is running. Since OCR is not CPU intensive but internet dependent, the results may vary depending on the connection speed. The time it took for the OCR API to recognize each character in the document is acceptable – an average of 0.51 s. Although CPU time and system time are important to the overall process, the most relevant result here is the user time. Users do not focus on the background process. Rather, they anticipate that the system gives a fast result. Moreover, the factor

affecting the performance of the upload was the pre-processing of the text from OCR since the libraries used are taking some system resources. Syncing to Dropbox would highly rely on the internet connection. The current ping from the host machine to Dropbox was 204 ms on average with the highest and lowest speed of 209 and 202 ms, respectively. The upload speed was 0.49 Mbps, which was taken from a speed test website.

Table 11. Performance testing result

Feature tested	Average user time (s/document)	Average CPU time (s/document)	Average system time (s/document)
Optical character recognition performance	0.51	0.65	0.15
Upload document performance	41.19	43.06	1.88
Sync to Dropbox	0.02	0.02	0.001

4. Conclusion and Recommendation

The study aimed to increase an organization's workflow by lessening the amount of time locating documents; hence, a system was developed. With the help of the rule-based pattern recognition, it is possible to generate a hierarchical structure to store the documents and classify them into letters, memos, or unclassified ones. The system yielded 98% accuracy for document classification. Compared with Naïve Bayes Classifier and SVM, the result of the feature similarity based on the rule-based pattern recognition was way better than the said two ML algorithms. On the other hand, the accuracy test of TF-IDF in finding and retrieving searched documents yielded 89% due to the system's user interface. However, this concern can be addressed by allowing more than 10 results to display to the user. The results of the performance testing found that the system took time to complete the whole process – from uploading the scanned document, generating the hierarchical structure up to cloud syncing. The system undergoes several processes that need numerous resources, and some of which were system intensive and internet dependent. Hence, if internet connectivity is unavailable or intermittent, the system offers local storage to augment the on-the-fly uploading of documents. The incorporation of the rule-based pattern recognition in classifying documents based on structure and storing them

automatically is one of the reasons why this algorithm can offer means for document classification and storage. Although supporting documents are expected not to be classified, future works should integrate other types of documents including certificates and research in the rule-based pattern recognition process.

5. References

Al-Sabahi, K., Zuping, Z., & Nadher, M. (2018). A hierarchical structured self-attentive model for extractive document summarization (HSSAS). IEEE Access, 6, 1-8. https://doi.org/10.1109/ACCESS.2018.2829199

Babic, B., Nesic, N., & Miljković, Z. (2008). A review of automated feature recognition with rule-based pattern recognition. Computers in Industry, 59(4), 321-337. https://doi.org/10.1016/j.compind.2007.09.001

Briiggemann-Klein, A. (1993). Regular expressions into finite automata. Theoretical Computer Science, 120, 197-213.

Cloud OCR SDK. (n.d.). Retrieved from https://github.com/abbyy/ocrsdk.com

Chagheri, S., Calabretto, S., Roussey, C., & Dumoulin, C. (2011). Document classification combining structure and content. Proceedings of the 3rd International Conference on Enterprise Information Systems (ICEIS), Beijing, China, 8-11.

Dengel, A., & Dubiel, F. (1995). Clustering and classification of document structure – A machine learning approach. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, Quebec, Canada, 587-591.

Dourish, P., Edwards, W., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D.B., & Thornton, J. (2000). Extending document management systems with user-specific active properties. ACM Transactions on Information Systems (TOIS), 18(2), 140-170. https://doi.org/10.1145/348751.348758

Dropbox for Java Developers. (n.d.). Retrieved from https://www.dropbox.com/developers/documentation/java

Extended Java WordNet Library. (n.d.). Retrieved from https://github.com/extjwnl/extjwnl

García, S., Luengo, J., & Herrera, F. (2015). Data preprocessing in data mining. Switzerland: Springer International Publishing.

Goller, C., Löning, J., Will, T., & Wolff, W. (2000). Automatic document classification: A thorough evaluation of various methods. IEEE Intelligent Systems, 14, 145-162.

Gulin, V., & Frolov, A. (2016). On the classification of text documents taking into account their structural features. Journal of Computer and Systems Sciences International, 55, 394-403. https://doi.org/10.1134/S1064230716030102

Henderi, Aini, Q., Srenggini, A., & Khoirunisa, A. (2020). Rule based expert system for supporting assessment of learning outcomes. International Journal of Advanced Trends in Computer Science and Engineering, 9(1.2), 266-271. https://doi.org/10.30534/ijatcse/2020/3991.22020

Lee, M., So, S., & Oh, H. (2016). Synthesizing regular expressions from examples for introductory automata assignments. New York, NY, USA: Association for Computing Machinery.

Power, R., Scott, D., & Bouayad-Agh, N. (2003). Document structure. Computational Linguistics, 29(2), 211-260. https://doi.org/10.1162/089120103322145315

Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. Proceedings of the First Instructional Conference on Machine Learning, 133-142.

Sarawagi, S. (2007). Information extraction. Foundations and Trends in Databases, 1(3), 261-377. https://doi.org/10.1561/1900000003

Shaw, F., & Fifer, R. (1986). An evaluation of five pc-based expert system shells. Retrieved from https://apps.dtic.mil/dtic/tr/fulltext/u2/a197915.pdf

Shin, C., Doermann, D., & Rosenf, A. (2001). Classification of document pages using structure-based features. International Journal on Document Analysis and Recognition, 3, 232-247. https://doi.org/10.1007/PL00013566

Singh, J., & Gupta, V. (2017). A systematic review of text stemming techniques. Artificial Intelligence Review, 48, 157-217. https://doi.org/10.1007/s10462-016-9498-2

Stede, M., & Suriyawongkul, A. (2010). Identifying logical structure and content structure in loosely-structured documents. In A. Witt & D. Metzing (Eds.), Linguistic modeling of information and markup languages (pp. 81-96). Switzerland: Springer Nature.