EUL: A Unified Research Repository and SDG Classification Framework using KNN and Cosine Similarity

---

A Thesis Presented to

The Faculty of the School of Computer Studies of

University of San Jose-Recoletos

Cebu City, Philippines

In Partial Fulfillment

Of the Requirements for the Degree

Bachelor of Science in Computer Science

By

Bohol, Cristopher

Premacio, Paul Joshua

Dr. Ma. Lorna D. Miro

Adviser

May 2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This paper aims to develop an institution-wide digital research repository system that automatically classifies approved research papers into 17 UN Sustainable Development Goals (SDG). It emphasizes the research contributions for the empowerment of sustainable development by serving as a comprehensive knowledge repository and an SDG classifier. This also serves as a comparative study between three different methods in classifying research papers, these methods are namely: TFIDF with Cosine Similarity Scoring, TFIDF Only, and K-nearest Neighbor algorithm. This classifier is composed of TF-IDF, for text vectorization and knowing the relevance of a particular word in a corpus; cosine similarity to calculate the similarity scores of each document based on the TF-IDF vectors.

CHAPTER I

INTRODUCTION

RATIONALE OF THE STUDY

Classification is a process of assigning objects into different classes or categories based on shared qualities or characteristics. Humans have done this process even before the invention of modern computers, Medieval army commanders sorting their formation based on the roles of the unit, pikes up-front, archers at the back, or a renaissance doctor categorizing medicines on a shelf and labeling them. The goal is simple: to easily manage and analyze the information at hand. A simple task no doubt, but doing it manually, with a large amount of information, let's say documents, is a tedious task. Manually managing documents is not ideal, especially if there are hundreds of research documents from the repository. To improve this process, we enlist the help of machines in the form of Artificial Intelligence (AI).

The advancement of modern computing gave birth to AI and eventually its underlying fields, Machine Learning (ML), and Natural Language Processing (NLP). Artificial Intelligence technology has been around since the 1940s [1]. It's been fine-tuned throughout the past decades. There are a lot of real-world applications that use ML and NLP. For instance: chatbots, language translators, email classification and filtering. With

the help of AI and its related fields, document classification can now be done automatically [2].

In 2015, all United Nations member states adopted a resolution that calls for a shared blueprint for peace and prosperity, they call this, ‘The 2030 Agenda for Sustainable Development. At its heart are the 17 Sustainable Development Goals (SDGs), in which developed and developing countries around the world are called to participate. With this plan in mind, classifying research according to the UN SDGs makes the research presently relevant.

This study aims to develop a system that an educational institution can benefit from by creating a web and mobile application that serves as a repository of all institutional research.

THEORETICAL BACKGROUND

Machine Learning

ML is defined as a type of AI that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so [3]. This can be done by using algorithms or models to analyze and draw inferences from patterns in the data. There are three types of Machine learning: *supervised*, *unsupervised*, and *reinforcement learning*. (“Different Types of Machine Learning [A Detailed Guide]”)

"Supervised Learning is an approach to creating artificial intelligence (AI), where a computer algorithm is trained on input data that has been labeled for a particular output." (“What is Supervised Learning? ”) The model is trained until it can detect the underlying patterns and relationships between the input data and the output labels, enabling it to

yield accurate labeling results when presented with never-before-seen data [4]. It can be further down into two categories, ***classification,*** and ***regression*** algorithms.

A **classification** algorithm is defined as a technique that is used to identify the category of new observations based on training data. (“Classification Algorithm in Machine Learning ”) In Classification, a program learns from the given dataset or observations and then classifies new observations into a number of classes or groups [5]. Examples of classification are SVM and Naive Bayes.

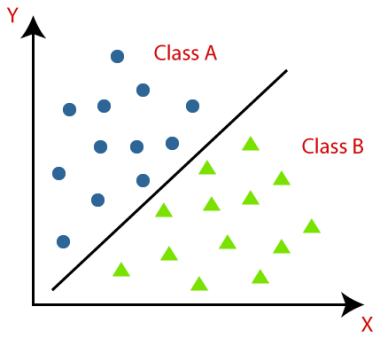


Figure 1 Sample Visualization of a Classification Algorithm

K-Nearest Neighbor Algorithm also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Term Frequency - Inverse Document Frequency (TF - IDF)

TF-IDF is an example of text vectorization. It counts the frequency of a word in a document to measure its relevance to a document in a collection of documents. Text Vectorization is the process of converting text into numerical representation [9].

Term Frequency (TF) is simply just the frequency of a word in a document.

Inverse Document Frequency (IDF) is the frequency of the word across all documents. This becomes the rarity of the word. The closer it is to 0, the more common the word is. Multiplying these two numbers of results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document [10].

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Figure 2 TF-IDF sample

Mathematically it can be described as,

$$tf\ idf\ (t, d, D) = tf\ (t, d)\ .\ idf\ (t, D)$$

Where,

$$idf\ (t, D) = log\ (\ \frac{N}{count\ (d \in D{:}t \in d)}\ )$$

$$tf\ (t, d) = log\ (1 + freq\ (t, d))$$

Natural Language Processing

Computers can't understand human language. It only understands 0s and 1s or binary information. For computers and humans to *communicate directly,* NLP is used. NLP is described as a branch of computer science—and more specifically, the branch of

artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can [11].

Cosine Similarity

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

REVIEW OF THE RELATED LITERATURE

Research paper classification systems based on TF-IDF and LDA schemes

A research paper classifier [13], uses TF-IDF as its text vectorization algorithm and LDA scheme in topic modeling. This study also uses the K-means clustering algorithm to finally classify its test data. While this paper is like EUL, the study in question only used the abstract part of the research as its source of data, while EUL uses the abstract, the rationale of the study, and the research methodology as its inputs.

Subject classification of research papers based on interrelationships analysis

A similar study of classifying research papers uses a novel supervised approach for subject classification of scientific articles based on an analysis of their interrelationships [14].

This study focuses more on scientific articles, citations, common authors, and common references to assign subjects to papers, whereas EUL accepts any form of the research topic if its authors are affiliated with USJ-R.

PROJECT OBJECTIVES

This study intends to develop a mobile and web application for managing USJ-R's research repository. Specifically, this study aims to:

- Have a single research repository for the entire USJ-R community.
- Classify papers based on the UN's 17 Sustainable Development Goals (SDG).
- Extract information from the research papers' relevant information like Approved Date, Title, Authors, Department, and Abstract.
- Give proper recommendation between three different SDG classification methods namely: TFIDF with Cosine Scores, TFIDF only, and KNN algorithm.

PROJECT SCOPE AND LIMITATIONS

EUL is a web and mobile development project study.

This study focuses on research papers from faculty and students of USJ-R. The papers are expected to follow USJ-R's approved format. It will extract needed information from source documents and classify the document according to UN SDGs.

A comparative study is also done between three different SDG classification methods. The goal of the comparative study is to determine what model is best suited for the SDG classification.

The study requires the device to be connected constantly to the internet as there are features of the system that fetches data from the cloud. As there is constant data processing and algorithms involved in the study, the machine that runs the processing should be at least a Ryzen 5 or intel core i5 with at least 8 GB of RAM. The accepted

format is only PDF files to give uniformity in the repository and the scraping tool the study is using is PDF-specific.

Its mobile component requires Android v4.1 (Jellybean) and above. The system only supports English as its main language. As of now, the application will only be used by USJR faculty and students.

SIGNIFICANCE OF THE STUDY

Presently, there is a lack of an institutional-wide research repository system. The study is not only a research repository but also a system that automatically classifies research in accordance with the United Nation's Sustainable Development Goals (UN SDG).

Figure 3 UN Sustainable Development Goal

Having a digital repository that classifies research papers based on UN Sustainable Development Goals promotes goal-oriented, and efficient knowledge discovery which enhances the institutions' contribution to sustainable development efforts on a global scale.

This also makes collaborations among departments and students much easier and more efficient because it is accessible through mobile and web applications.

Furthermore, the study will serve as a comparative framework for future SDG document classification studies in the university. The study

RESEARCH METHODOLOGY

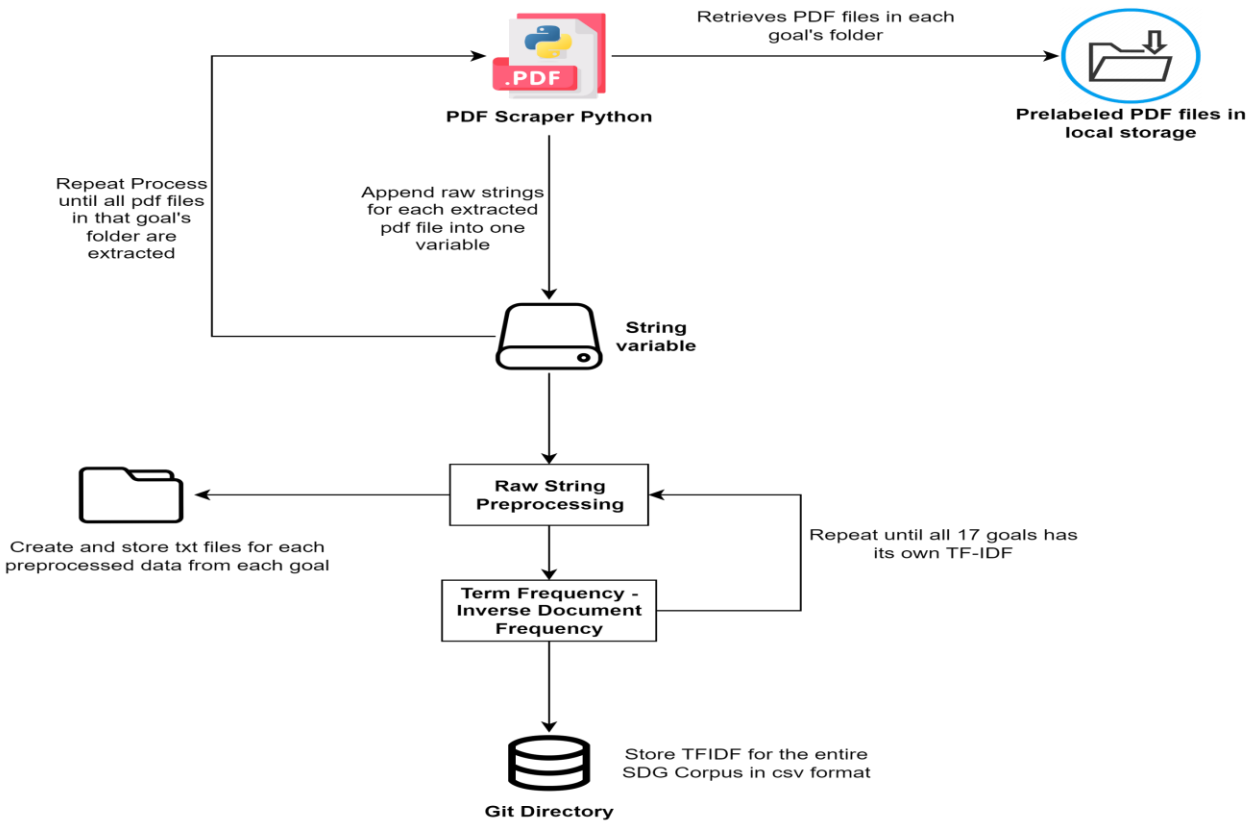


Figure 4: A conceptual framework for model training

Training the Model

The study uses a supervised technique in classifying documents and as such, the study already has predefined training data for each goal. Each goal consists of 5 pdf files that are manually classified as part of that goal. The entire training corpus consists of 85 pdf files. To limit the computational time of training the model, each pdf file in the training set is limited to only 3-5 pages. These pages are manually chosen to be the candidate for training as most of the relevant keywords are found on these pages. The pdf files are fetched directly from google scholar's wide variety of research journals and papers. Figure 5 shows the process of training the model based on predefined labels.

Once the training starts, the first step is the scraper will extract all the words in a single pdf file per goal. Once all pdf files in that goal have been scraped, the next process will start.

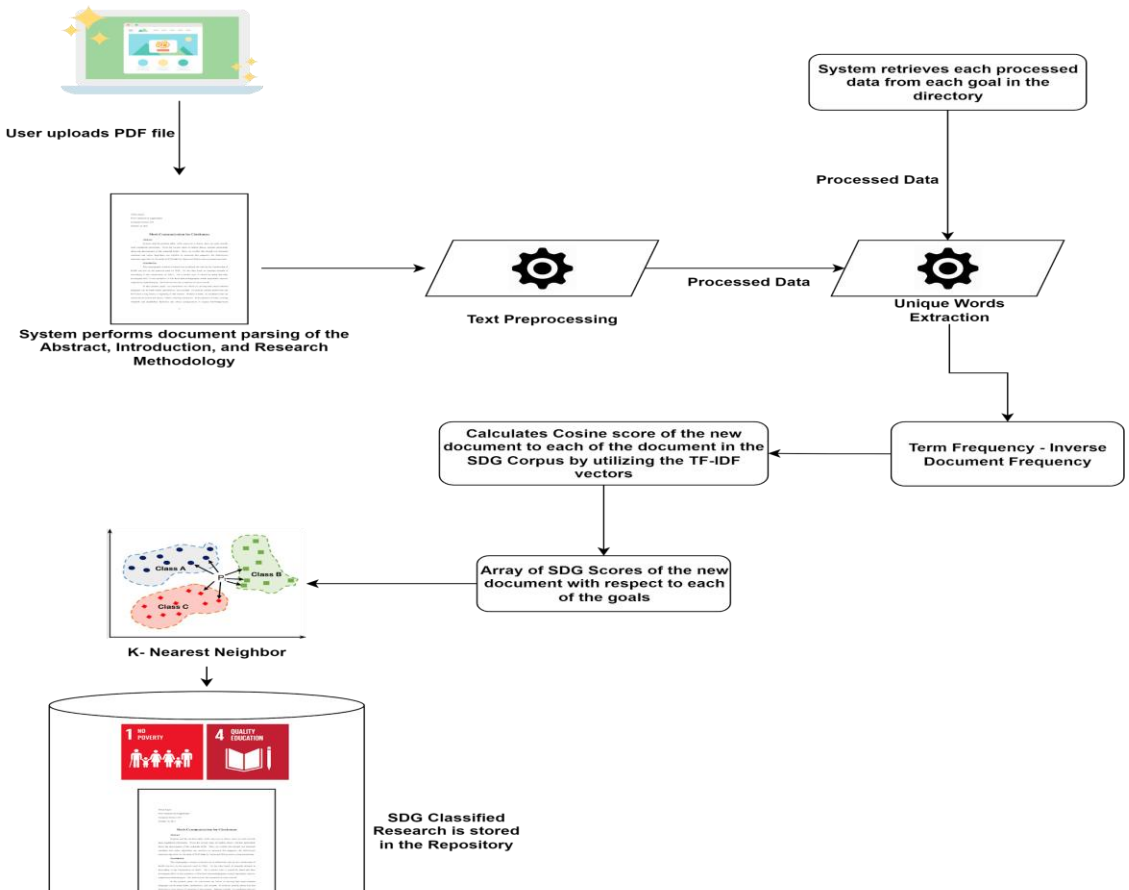
The second process is text processing, in this area, the appended string of all pdf files in one goal is cleaned of all unnecessary text, like stop words, punctuations, and numbers. Just like in any machine learning model, text preprocessing plays an integral part in the accuracy of the model as it removes unwanted or unimportant texts. Regular Expression and Lemmatization techniques are used in this process.

Another process is to store preprocessed strings and store them into their respective text files labeled with respect to their goal. This process is important in classification.

Once preprocessing is done, words are assigned a numerical value, representing the frequency of the word in the document. This process is called text vectorization. Term Frequency- Inverse Document Frequency is the most common form of text vectorization. There are other common techniques like bag of words, but the advantage of using TF-IDF is, it gives a numerical value or in this case, IDF, of a particular word in the entire corpus, meaning, it calculates how relevant that word is with respect to the SDG corpus. The lesser the value, the more common it is in the SDG corpus and vice versa.

Since the study needs to classify documents based on 17 UN SDG, the system generates TF-IDF for each SDG, and afterward creates a single TF-IDF that represents the entire SDG corpus.

After creating the TF-IDF for the SDG corpus, the result is then stored locally in a CSV format. This is a one-time process and can only be triggered once the system fails to locate the stored TF-IDF CSV file or by manually calling a function that trains the model.



Classification

The Abstract, the Rationale of the study, and the Research Methodology were chosen to be used as the primary source of input data since most of the relevant keywords needed for the system to classify can be found in these documents. In the first step, the user uploads a file (pdf), and the system will perform document parsing to extract texts from the Abstract, the Introduction, and the Research Methodology.

This appended data is composed of the extracted Abstract, Introduction, and Research Methodology and represents the new document to be classified. This appended data will undergo the same process when the system trains the model.

The system will perform the usual text preprocessing of the appended data. Once done, the system will retrieve the preprocessed data for all goals stored in the directory from the training process. These data retrieved from the directory alongside the preprocessed appended data are then used to extract unique words which will then be used to update the TF-IDF feature set.

After preprocessing, the system will now calculate the TF-IDF vectors for every 18 documents, the last document now represents the document to be classified.

Once the TF-IDF is done creating, the system will calculate the cosine similarity score of the last document to each of the documents in the TF-IDF. This will give an array of cosine scores that represent the similarity of the new document to each of the documents in the SDG corpus.

The cosine similarity scores will then be used as input to the KNN classifier. The KNN classifier performs the majority voting scheme to predict the labels or goals of the new document. The KNN classifier has a hyperparameter called K, this K represents the number of data points considered for classifying the new document. The optimal value of K depends on the nature of the data set. In this study, there are only 17 documents, and as such the value of K is set to 5.

Information Extraction

The system also incorporates information extraction every time the user uploads a PDF file. This technique automatically extracts the Approved Date, Research Title, Research Authors, and Department/School.

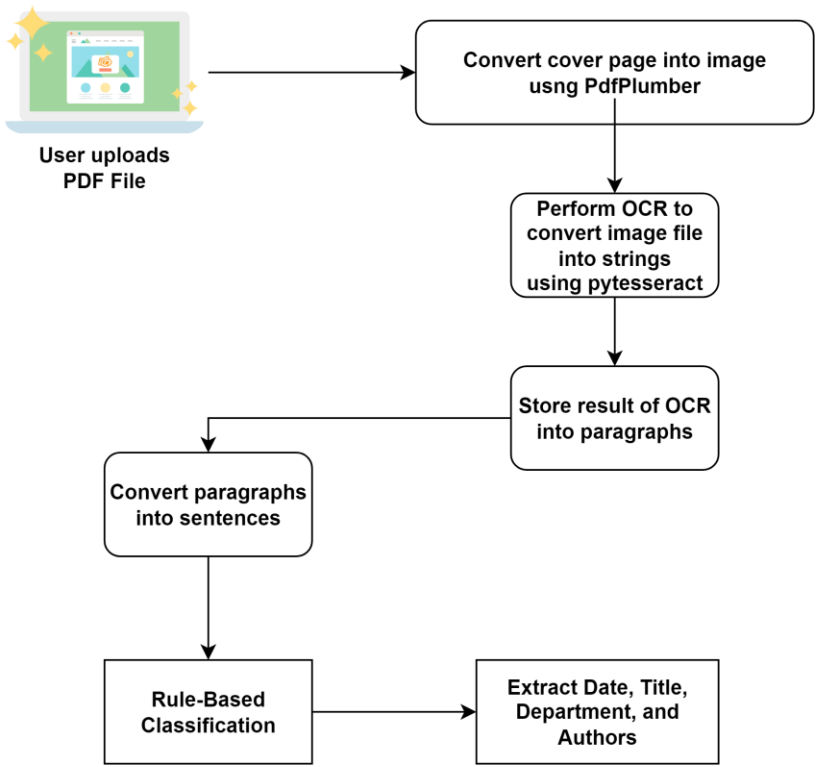


Figure 6 visualizes how the system extracts information when the user uploads a PDF file. The first step is to convert the cover page into an image using pdfplumber. Once the image has been created, the system performs OCR using pytesseract to convert the image file into strings. After performing OCR, the result is then transformed into paragraphs, after transforming the paragraphs are then split into sentences. After splitting, a rule-based classification is used to extract the Date, Title, Authors, and Department.

This process makes the extraction of the raw text accurate and removes unwanted spacing if the usual pdfplumber is used. This makes the rule-based identification method an accurate and clean result.

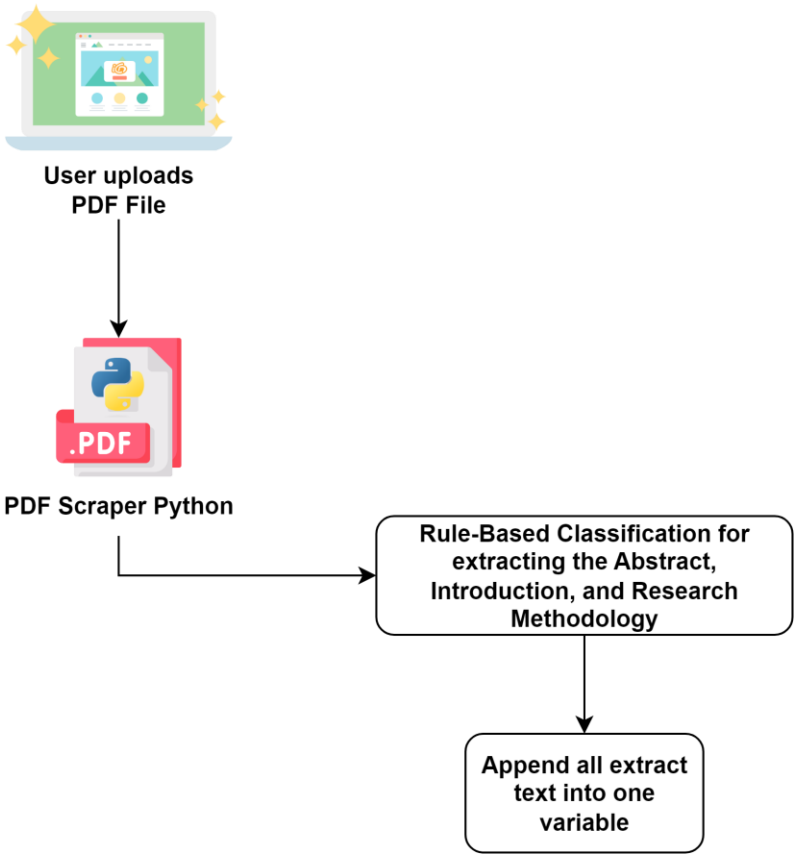


Figure 7 A conceptual framework for Extracting Abstract, Introduction, and Research Methodology

Figure 7 shows how the system extracts the Abstract, Introduction, and Research Methodology to be used as input to the SDG classifier. The pdfplumber is used here to extract text from pdf files instead of OCR since the result of extraction is clean.

A rule-based classification is then applied to identify which part of the extracted text is an Abstract, Introduction, and Research Methodology. To lessen the computational process, the system only limits the scraping to 15 pages as these 3 sections are usually found in these parts.

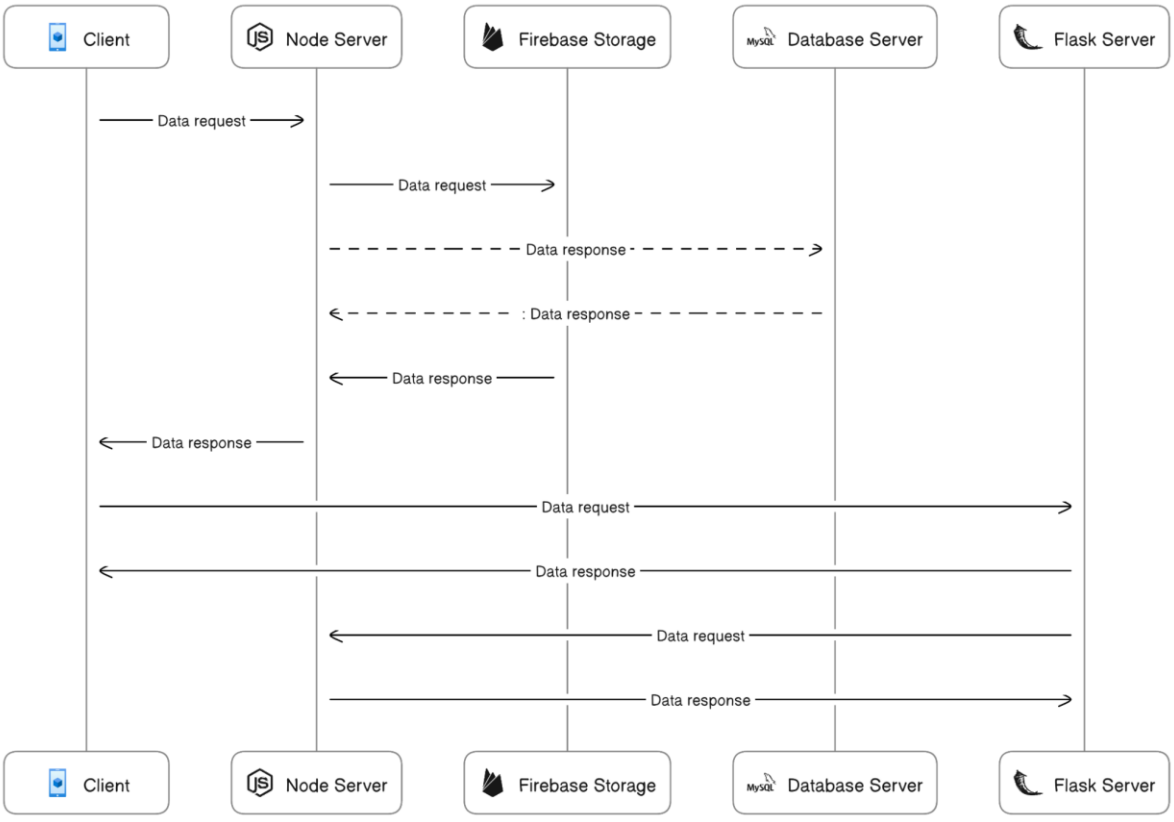


Figure 8 Data Request Diagram

Figure 8 shows the process on how the data are processed after requesting a service from the app. When user creates an account in the repository, it will first go to the Node Server, then to Firebase Storage for storing the profile picture and then finally into the database server. When classifying and extracting data, the applications will directly call to the Flask server, then the flask server calls the necessary scripts then outputs the results via JSON.

CHAPTER II

SOFTWARE REQUIREMENTS AND DESIGN SPECIFICATIONS

This chapter specifies the user and system requirements that are expected to be accomplished as well as the structure and process of achieving these. It contains sections for the Architectural Diagram, Use Case Diagram, Use Case Narrative, Activity Diagram, Class Diagram, and User Interface Design.

ARCHITECTURAL DIAGRAM

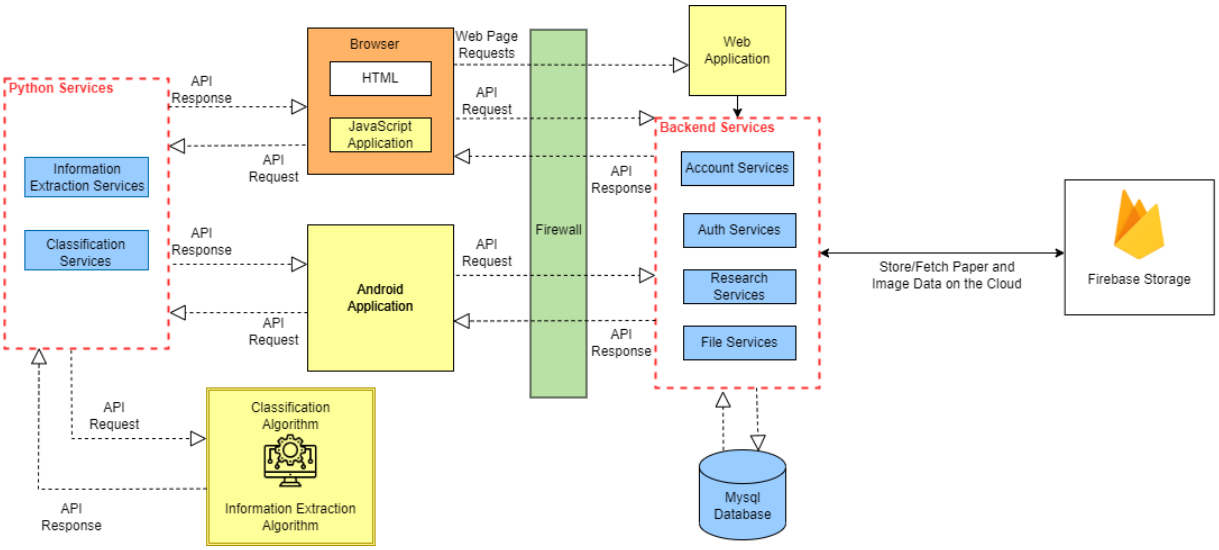


Figure x: Architectural Diagram of the System

Figure x illustrates the architectural diagram of the system which comprises the flow of the system with the different technologies involved in the process. The study uses two different frameworks for both frontend and backend, and another framework for the algorithm processes. Separate firebase storage is used in storing research papers and

the images from the users. The system utilized APIs to communicate between different frameworks with the help of a backend technology.

USE CASE DIAGRAM

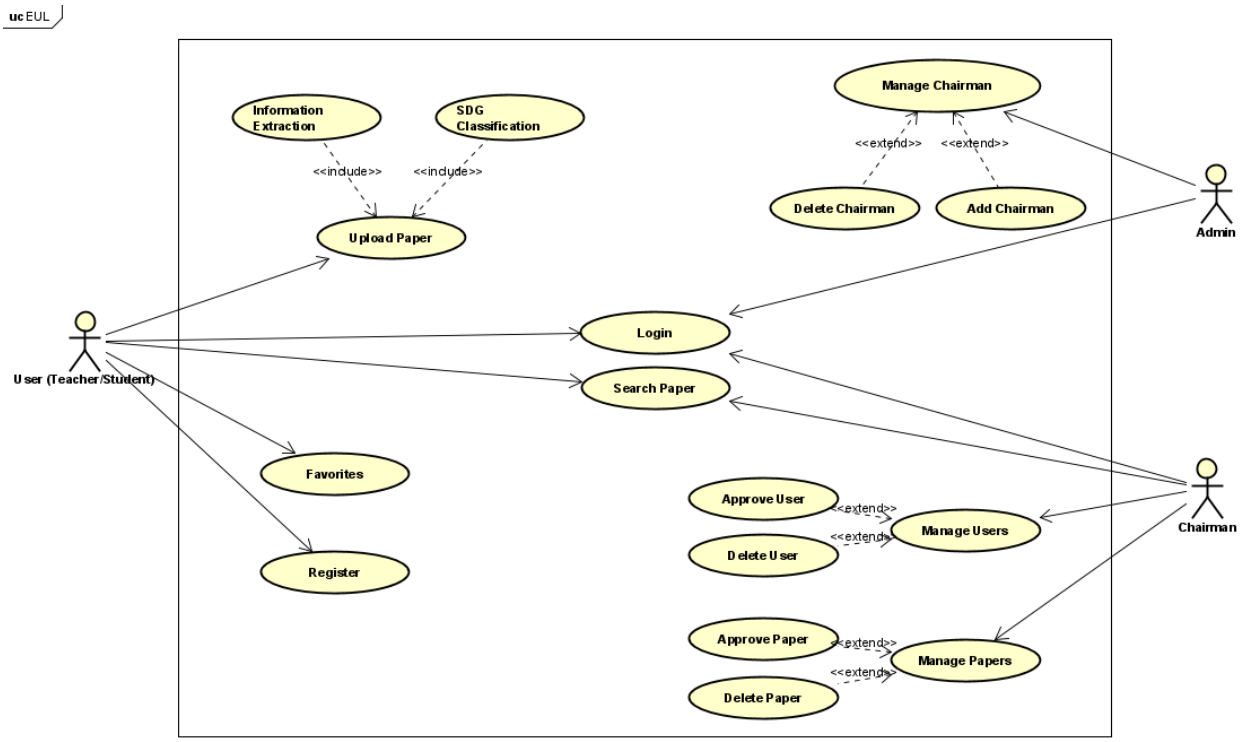


Figure 10: Use Case Diagram

Defines the different use cases in which the system performs when students, teachers, chairman and admin query the research repository on the Application. This diagram maps how different actors or users interact with our system.

USE CASE NARRATIVE

The use case narrative shows the main scenarios and alternative flows of a use case. It provides more details about the use case.

Table 1 Use Case 001 Register User

Use Case 001	Register User
Actors	Student, Teacher
Overview	Allows users to register for an account.
Precondition(s)	None
Postcondition(s)	System creates new user account
Flow of Events	
Actor Actions	System Actions
1. User navigates to registration page.	2. System presents the registration form.
3. User enters their user details.	5. System validates the registration details.
4. User clicks submit button.	6. System creates a new user account.
	7. System display Success Message.
Alternative Flow of Events: 5a. If the system detects if user exist.	
2. User needs to input again	1. System displays an error message.
3. User submits the user form	

Table 2 Use Case 002 Upload Research

Use Case 002	Upload Research
Actors	Student, Teacher
Overview	Actors can upload a soft copy of research to the database.
Precondition(s)	User is log in

Postcondition(s)	Research has classified and stored into the existing research repository.
Flow of Events	
Actor Actions	System Actions
1. User navigates to upload page.	4. The system calls flask to verify if research paper is acceptable.
2. User selects file to upload	5. The system will proceed to UC003
3. The user click the next button	6. Result from UC003 used to filled up form
7. User provides necessary details of the research paper.	9. System classifies the research paper.
8. User click submits the research paper	10. System stores the research paper in the database and cloud storage.
Alternative Flow of Events: 8a. Internet Connection has disconnected.	
2. User clicks submits research paper	1. The system will delete existing research details store in the database
Alternative Flow of Events: 8b. User clicks rest button	
2. User selects file to be uploaded	1. User will be redirected to first step with empty file selector

Table 3 Use Case 003 Classify Research

Use Case 003	Classify Research
Actor(s)	System
Overview	Automatically classifies paper based on UN SDGs. Top four goals are displayed.

Precondition(s)	The file uploaded passed the acceptance checker function.
Post Condition(s)	Returns classified goals to UC002
Flow of Events	
Actor Action(s)	System Response
	1. System extracts the Abstract, Introduction, and Research Methodology of the paper.
	2. System performs preprocessing of the extracted texts.
	3. System creates TF-IDF of the extracted texts.
	4. System scores the TF-IDF of the newly uploaded document using cosine scoring.
	5. System calls KNN classifier to predict SDG labels using the Cosine Score lists.
	6. System successfully predicted top four SDG goals
	7. System returns extracted information ton UC002

Table 4 Use Case 004 Remove Research

Use Case 004	Remove Research
Actor	Chairman
Overview.	Allows the chairman to remove research papers
Precondition(s)	User is log in as Chairman.
Postcondition(s)	System removes research details from database.

Flow of Events:	
Actor Actions	System Actions
1. Chairman selects the "Remove Research"	4. System deletes the research paper.
2. Chairman selects the research paper to remove.	5. System updates the repository accordingly.
3. Chairman enter password	
Alternative Flow: 3a. Chairman does not have permission to remove the research paper.	
Users need to enter password	

Table 5 Use Case 005 Approve Research

Use Case 005	Approve Research
Actors	Chairman
Overview	Allows the chairman to approve research papers.
Precondition(s)	User is log in as Chairman.
Postcondition(s)	Chairman approves the research paper
Flow of Events:	
Actor Actions	System Actions
1. Chairman selects the "Approve Research"	4. System updates the status of the research paper to "approved".

2. Chairman selects the research paper to approve.	
3. Chairman enter password	
Alternative Flow: 3a. Chairman does not have permission to approve the research paper.	
Users need to enter password	

Table 6 Use Case 006 Approve User

Use Case 006	Approve User
Actors	Chairman
Overview	Allows the chairman to approve user accounts.
Precondition(s)	User is log in as Chairman.
Postcondition(s)	Chairman approves user registration
Flow of Events:	
Actor Actions	System Actions
1. Chairman selects the "Approve User"	4. System updates the status of the user account to "approved".
2. Chairman selects the user account to approve	
3. Chairman enter password	
Alternative Flow: 3a. Chairman does not have permission to approve the research paper.	
Users need to enter password	

Table 7 Use Case 007 Remove User

Use Case 007	Remove User
Actors	Chairman
Overview	Allows the chairman to remove user accounts.
Precondition(s)	User is log in as Chairman.
Postcondition(s)	Chairman removes user from database
Flow of Events:	
Actor Actions	System Actions
1. Chairman selects the "Remove User" option.	4. System deletes the user account.
2. Chairman selects the user account to remove	5. System updates the user list accordingly.
3. Chairman enter password	
Alternative Flow: 3a. Chairman does not have permission to approve the research paper.	
Users need to enter password	

Table 8 Use Case 008 Add Chairman

Use Case 008	Add Chairman
Actors	Admin
Overview	Allows the admin to add a chairman.

Precondition(s)	User is log in as Admin.
Postcondition(s)	Admin created chairman account
Flow of Events:	
Actor Actions	System Actions
1. Admin selects the "Add Chairman"	4. System creates a new chairman account.
2. Admin provides the necessary details for the chairman account.	
3. Admin enter password	
Alternative Flow: 3a. Admin does not have permission to delete the chairman	
Users need to enter password	

Table 9 Use Case 009 Delete Chairman

Use Case 009	Delete Chairman
Actor	Admin
Overview	Allows the admin to delete a chairman account.
Precondition(s)	User is log in as Admin.
Postcondition(s)	Admin deletes account of chairman
Flow of Events:	
Actor Actions	System Actions
1. Admin selects "Delete Chairman"	4. System deletes the chairman account.

2. Admin selects the chairman account to delete	
3. Admin enter password	
Alternative Flow: 3a. Admin does not have permission to delete the chairman	
Users need to enter password	

Table 10 Use Case 010 Search Repository

Use Case 010	Search Repository
Actors	Student, Teacher, Chairman, Admin
Overview	Allows users to search for research papers.
Precondition(s)	User is logging in.
Postcondition(s)	User views the search results.
Flow of Events:	
Actor Actions	System Actions
1. User navigates to search page	3. System performs the search based on the entered criteria.
2. User enters the search criteria (keywords, authors, topics, etc.).	4. System retrieves and displays the search results.
Alternate Flows 3a. The search returns no results.	
1. The system displays a message indicating that no results were found	

Table 11 Use Case 011 Download Research Papers

Use Case 011	Download Research Papers
Actors	Student, Teacher, Chairman, Admin

Overview	Allows users to download research papers.
Precondition(s)	User is logged in.
Postcondition(s)	Research paper is downloaded.
Flow of Events:	
Actor Actions	System Actions
1. User selects the "Download Icon"	2. System initiates the download of the selected research paper file.
Exceptional Conditions: Research paper is not available for download:	
1. System detects that the selected research paper is not available for download.	
2. System displays an error message indicating that the research paper is not accessible.	

Table 12 Use Case 012 Bookmark Research Paper

Use Case 012	Bookmark Research Paper
Actors	Student, Teacher, Chairman, Admin
Overview	Allows users to bookmark selected research paper
Precondition(s)	User is logged in
Postcondition(s)	Research paper is bookmarked.
Flow of Events:	
Actor Actions	System Actions
1. User selects the "Bookmark Icon"	2. System adds the bookmark to the user's bookmark list.

Table 13 Use Case 013 Login

Use Case 013	Login
Actor(s)	Student, Teacher
Overview	To let the actors, access all features
Precondition(s)	none
Post Condition(s)	Redirected to homepage
Flow of Events	
Actor Action(s)	System Actions
1. User navigates to login page	2. System checks if the credentials store in local storage is empty
3. User needs to fill up necessary inputs (school id and password)	5. The system checks if account credentials entered exist.
4. The user clicks submit button	6.Actors will be redirected to the private homepage
Alternate Flow of Events: A2. user tokens exist in local storage	
1. The system will redirect to private homepage	
Alternate Flow of Events: A4: Credentials are not valid	
1. The system will display the error message then user needs to input again.	
2. If the submitted credentials are correct, redirect to the private homepage.	
3. If the system detects that the number of an attempt made by the user is equal to 3, then the system will notify the user who wants to reset the password.	
Alternate Flow of Events: A4: Credentials are valid, but the account was not yet approved by chairman	

1. System will redirect to approval page and display the error message to wait for a confirmation message.

ACTIVITY DIAGRAM

The activity diagram shows the workflow behavior of the system by describing the sequence of actions in the process.

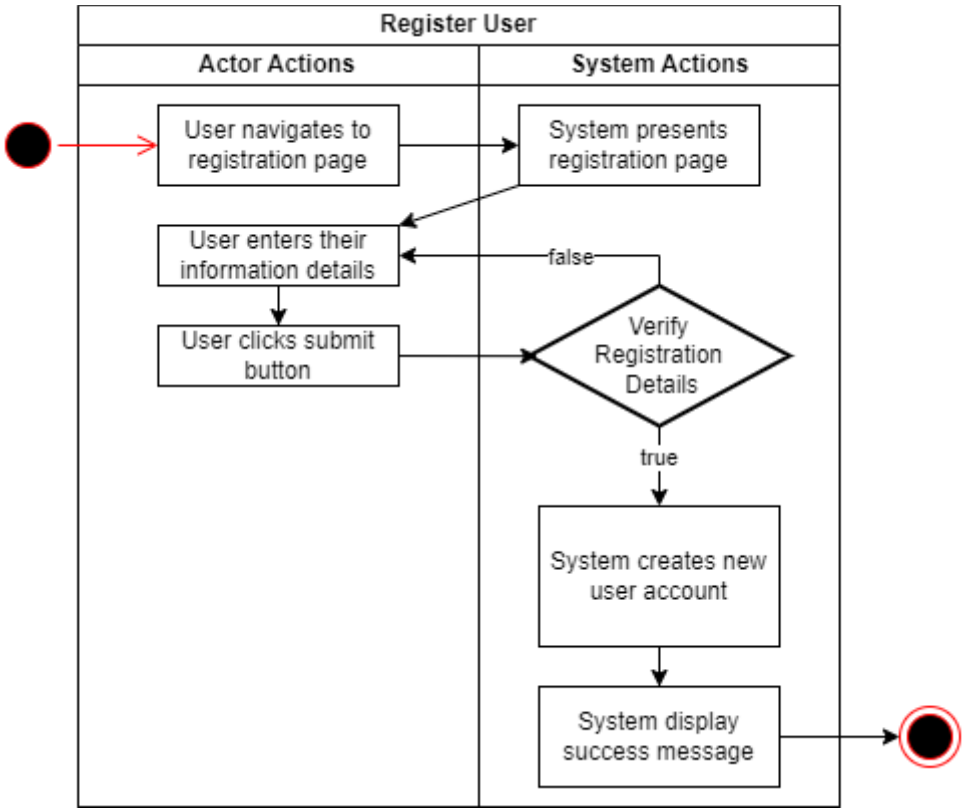


Figure 9 Register User

Figure 10 shows how the user can register through both the mobile and web application. The user clicks or taps the register tab, user will then be redirected to the registration page. The system will display text fields for the users to input their information. The system will then check if the input information is correct. If accepted, the system will create an account and display a success message.

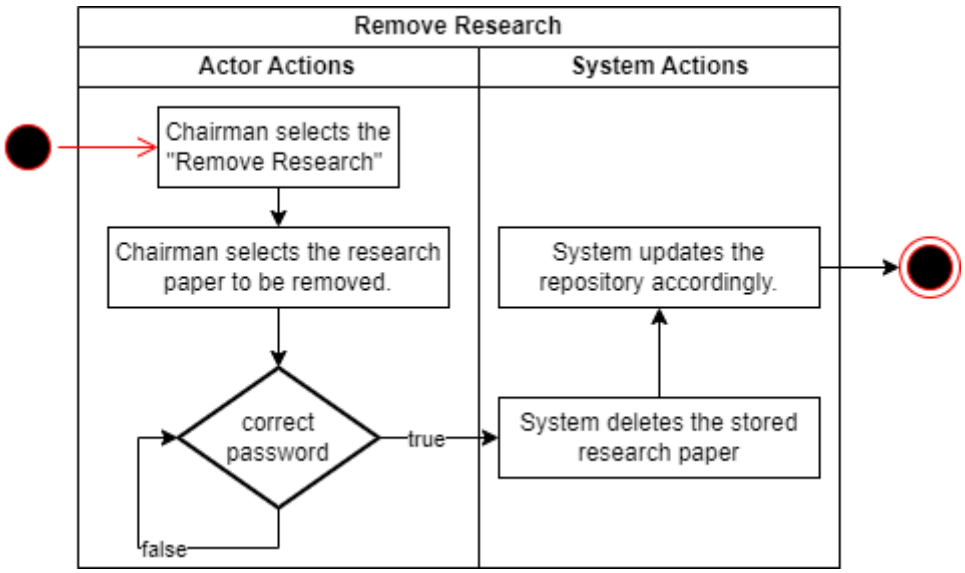


Figure 10 Remove Research

This depicts how the Chairman can remove the stored paper using the web application. This feature is only available on the web application. Chairman selects the remove research tab. Chairman will then select which research paper to be removed. System will then prompt the chairman to reenter the password for security purposes. If valid, the system will delete the chosen paper.

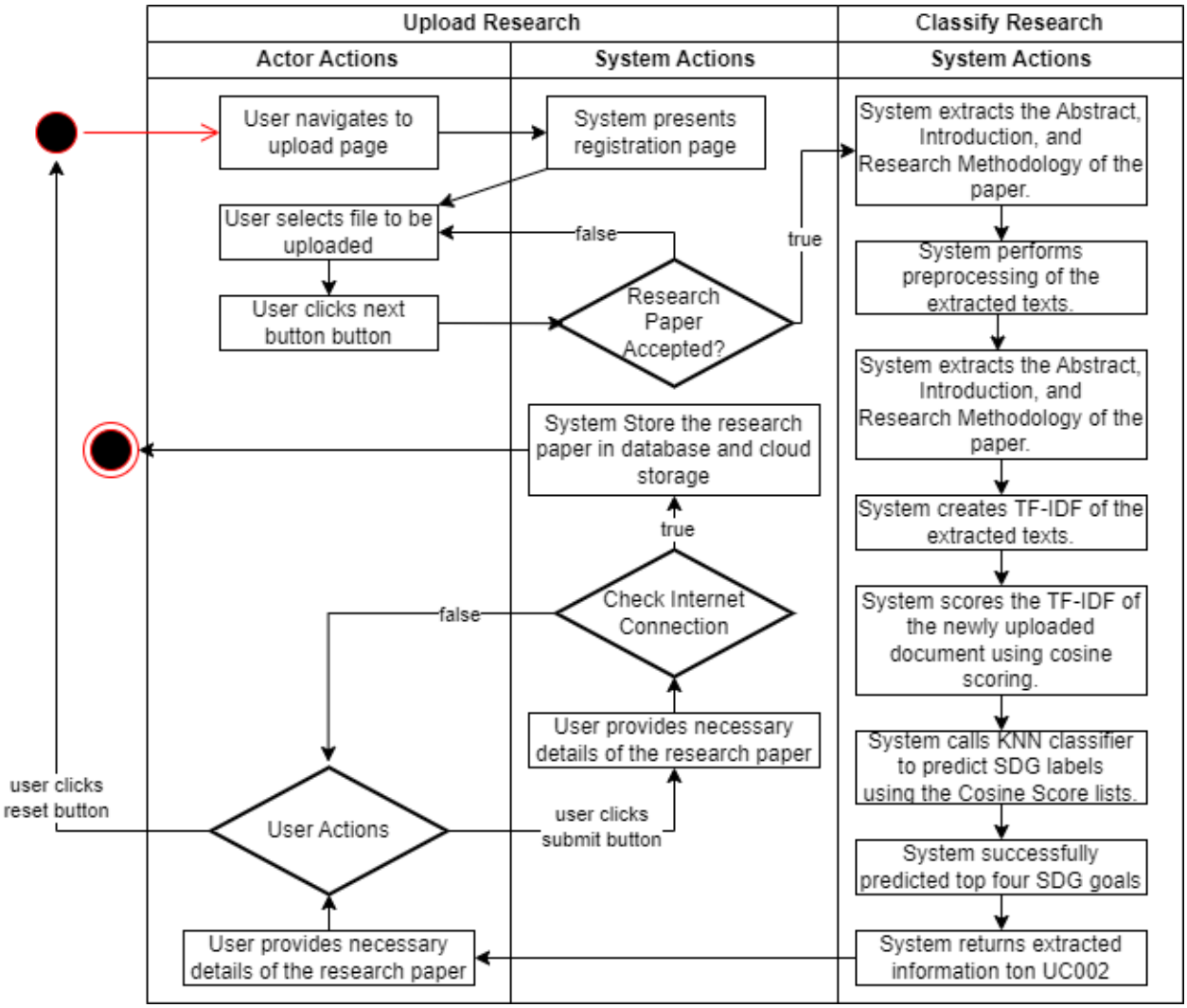


Figure 11 Upload and Classify Research

This diagram shows the process of uploading research paper and classifying it into different SDGs. In the mobile and web application, the user must click or tap first the upload button to be redirected into the upload screen. Once in the upload screen, the user can click pick research paper button to be redirected into the device's local storage where the PDF files are stored. Once the user picks a file, the system will then check for its acceptance, the system will scrape the paper and check for the endorsement section. If true, the system will then proceed to extract information from the paper. Once done, the

user can tap or click the classify button to proceed into the next screen. After information extraction, the system will then perform SDG classification, this will take around 30 seconds to finish and once it is done, the system will display the goals that the paper belongs to.

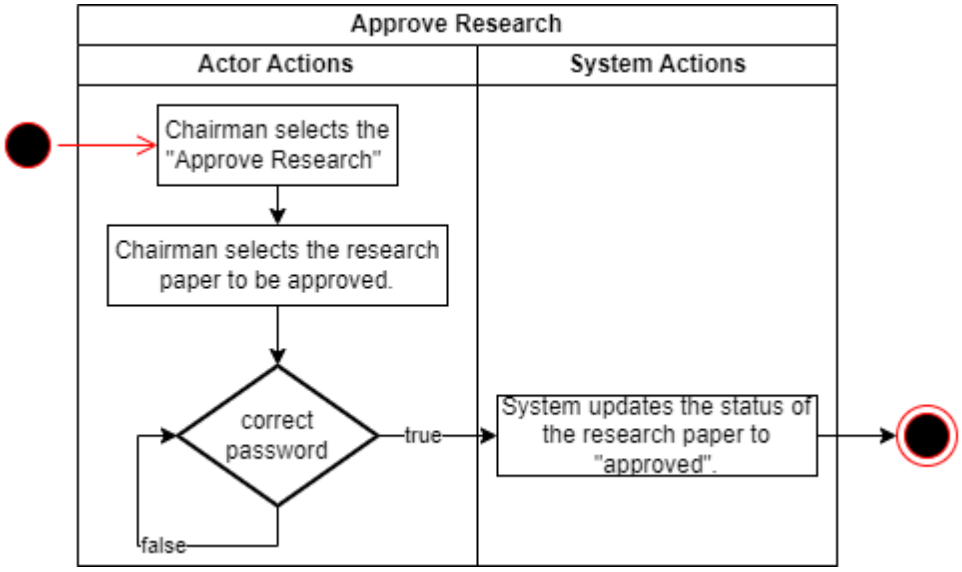


Figure 12 Approve Research

Figure 13 shows how the chairman can approve the research uploaded by the user after classifying. To avoid students uploading a lot of duplicate research, before it can be viewed by the user, it should be approved by the chairman of the department. The chairman can click the approved button to approve a research paper. This feature can only be accessed through the web application.

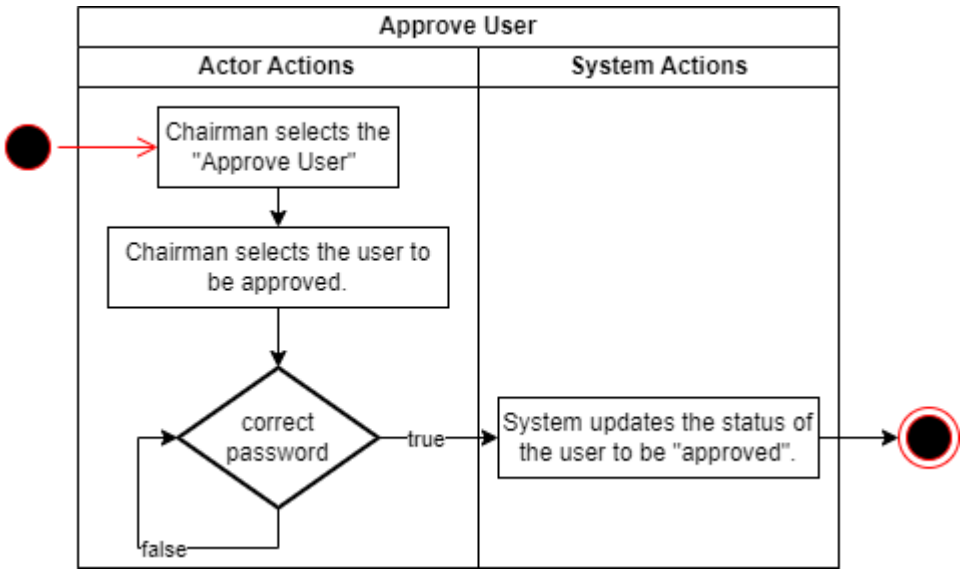


Figure 13 Approve User

After registering to EUL, the chairman must approve first the newly registered user. The chairman can click the approved button to approve the user. This feature is only available on the web application.

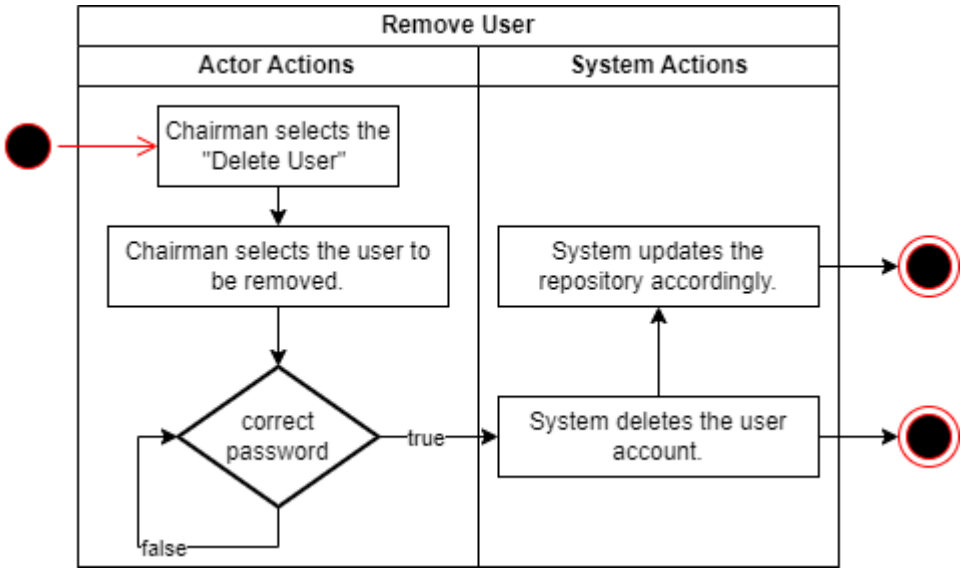


Figure 14 Remove User

Figure 15 shows how the chairman can remove the user. This feature can only be accessed through the web application. Just like any CRUD chairman functions, this also requires the chairman to input the password for security validations, once passed the chairman can now delete the student's account.

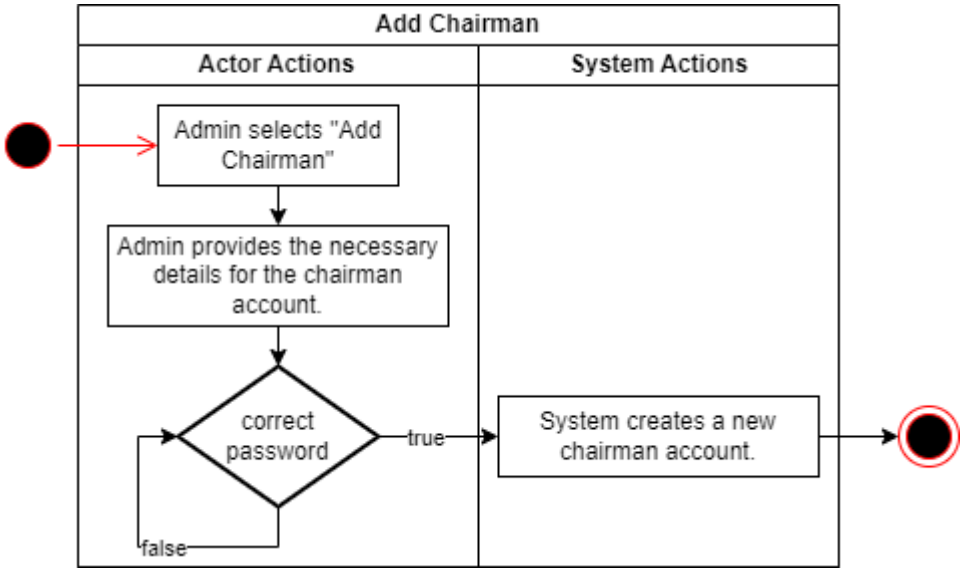


Figure 15 Add Chairman

The administrator of EUL can add chairman to the respective departments. This feature is only accessible by the admin in the web application. Admin clicks the Add Chairman button and system will perform validity checking, once passed, the system creates a new chairman account.

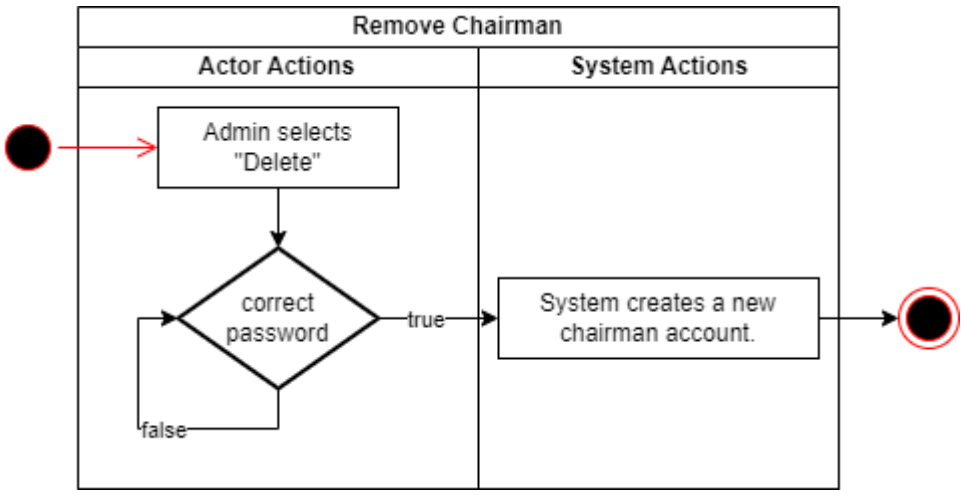


Figure 16 Remove Chairman

Figure 17 shows the process on how the admin can remove a chairman account. First admin can select the Delete button. The system will then perform validity checking, once passed the system will delete the picked chairman account.

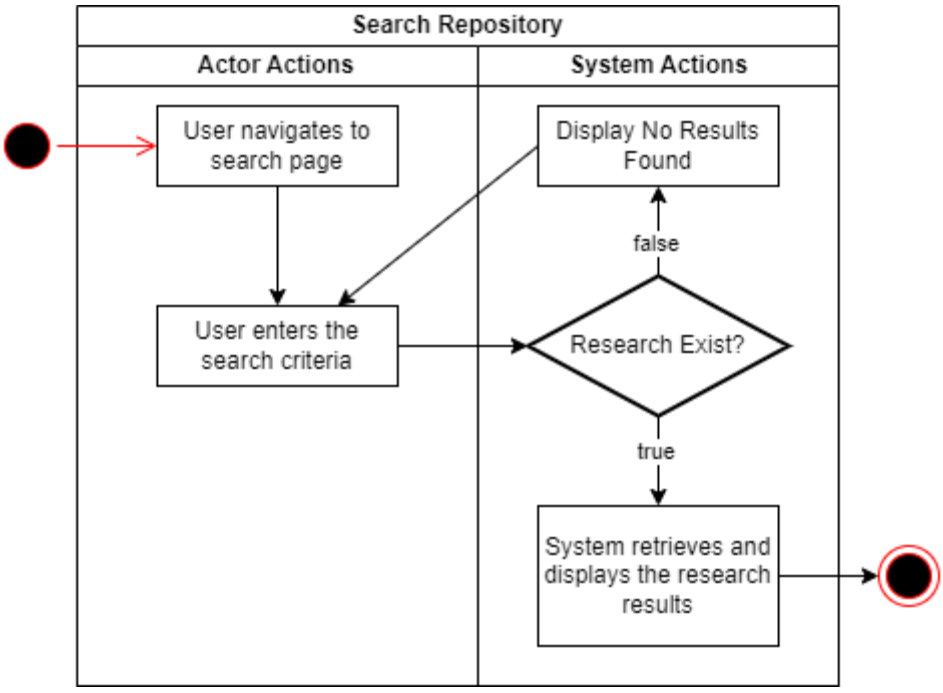


Figure 17 Search Repository

Users can search repository from both mobile and web application. Users can filter categories on which research papers will be displayed. If there's no result to the query, the system will display an appropriate message.

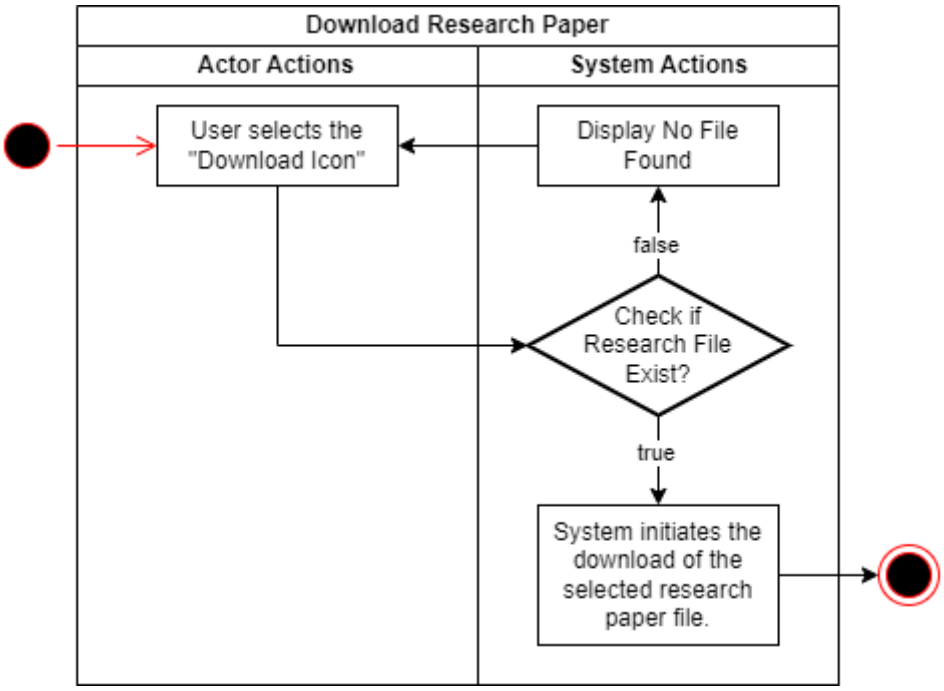


Figure 18 Download Research Paper

Figure 19 shows the process of downloading a research paper. This is only accessible in the mobile application. User clicks the download icon; system will then initiate the download process if the research file exists in the repository.

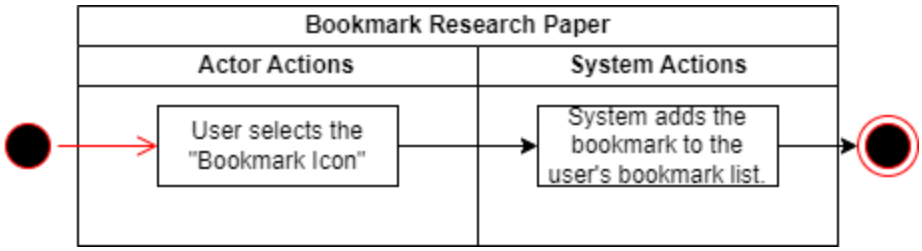


Figure 19 Bookmark Research Paper

Figure 20 shows how the user can bookmark a research paper. This feature can be accessed from both mobile and web applications. User picks a research paper, clicks, or taps on the bookmark icon and system will automatically bookmark the research paper. The paper can be accessed through the My Bookmarks tab.

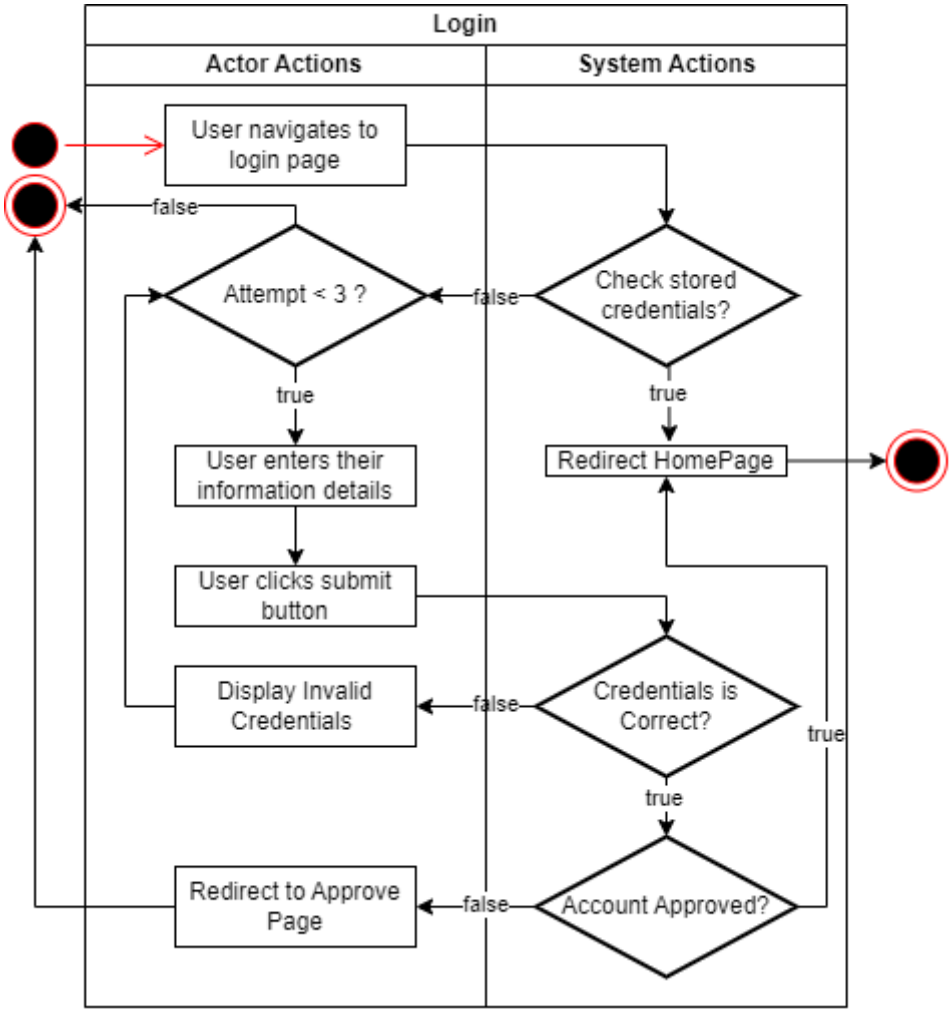


Figure 20 Login

Figure 21 shows how the users can login to the web and mobile application. Users need to enter their student ID and their password. System will then check for credentials. If an account is found, it will then check if it's approved or not, if approved, user will then be redirected to the application's dashboard.

CLASS DIAGRAM

The class diagram shows the structure of the system as classes with their attributes, operations, and relationships among other classes.

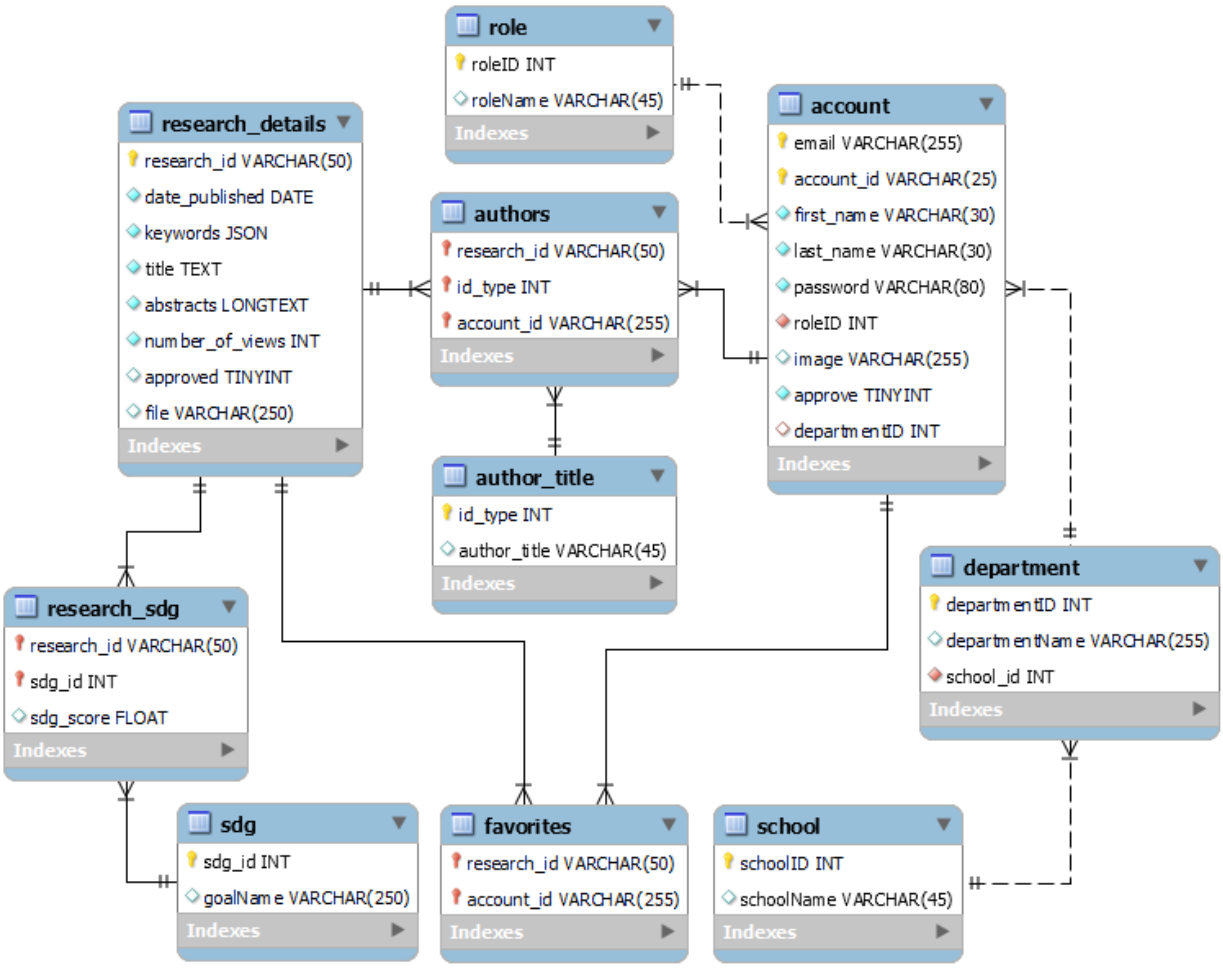


Figure 21 Class Diagram of EUL

Figure 22 illustrates the relationship between different classes in EUL. The research details class contains the information needed for a research paper. It is connected to the authors, role, author_title, research_sdg, and favorites. An account class is also connected to the role class, the role class contains the role id and the role name.

USER INTERFACE DESIGN

The user interface design is the process of creating interfaces that are expected to provide insights into how the user can interact with the system.

Mobile Application

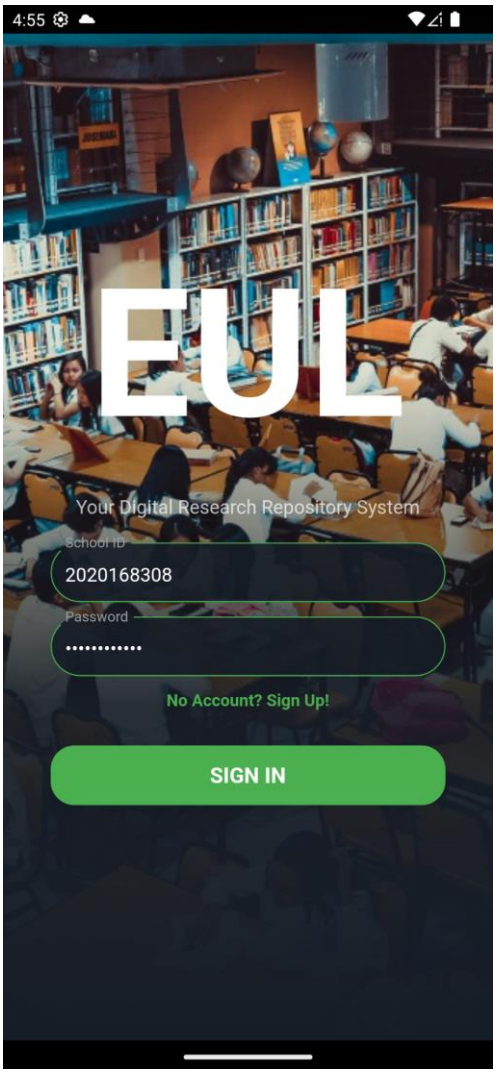


Figure 23 Login Page

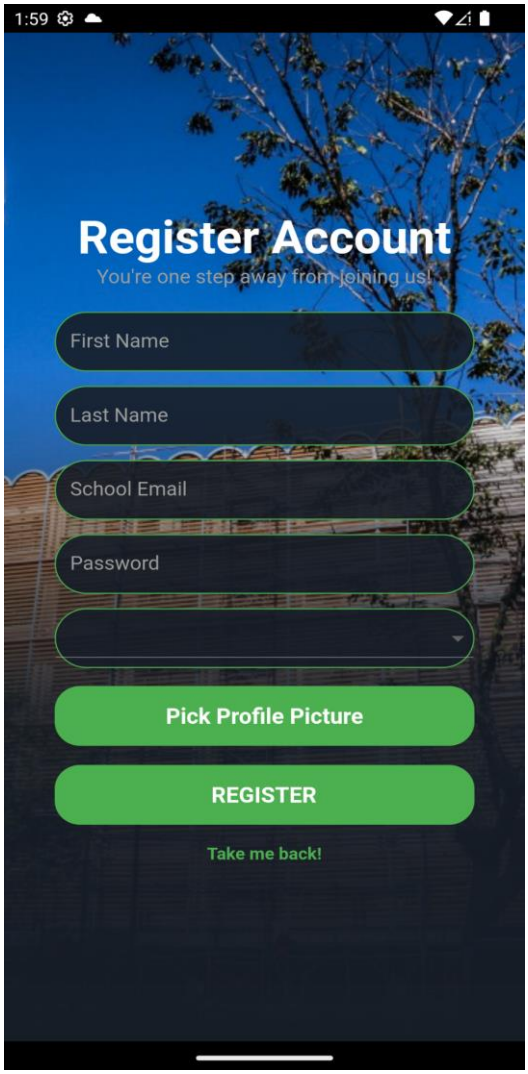


Figure 22 Registration Page

This screen shows how the user can create an account with EUL. Users can register by inputting the necessary information on the screen.

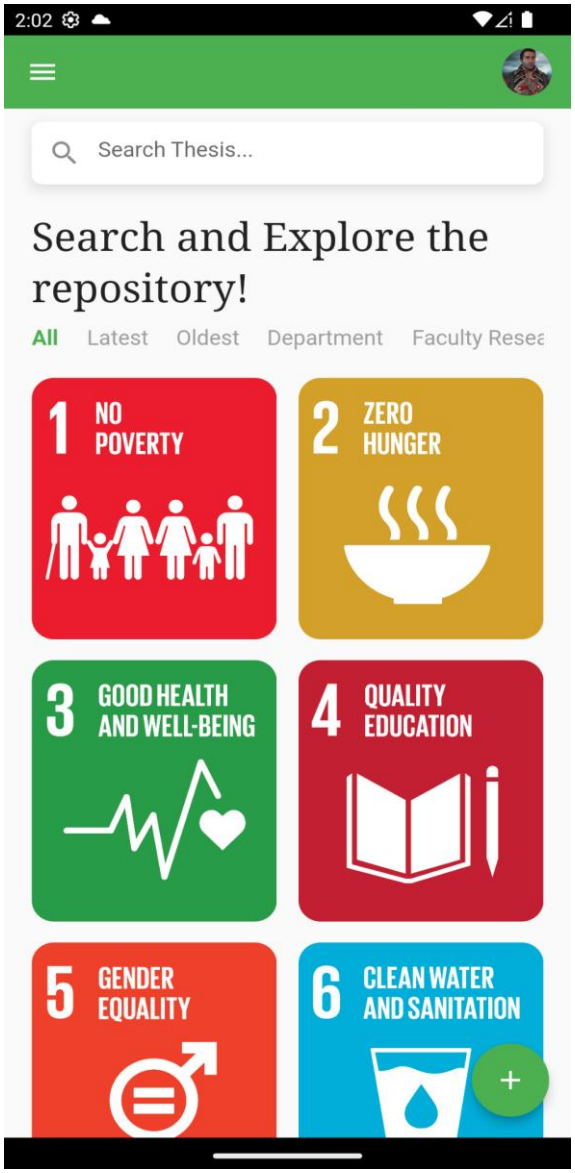


Figure 25 Home Page

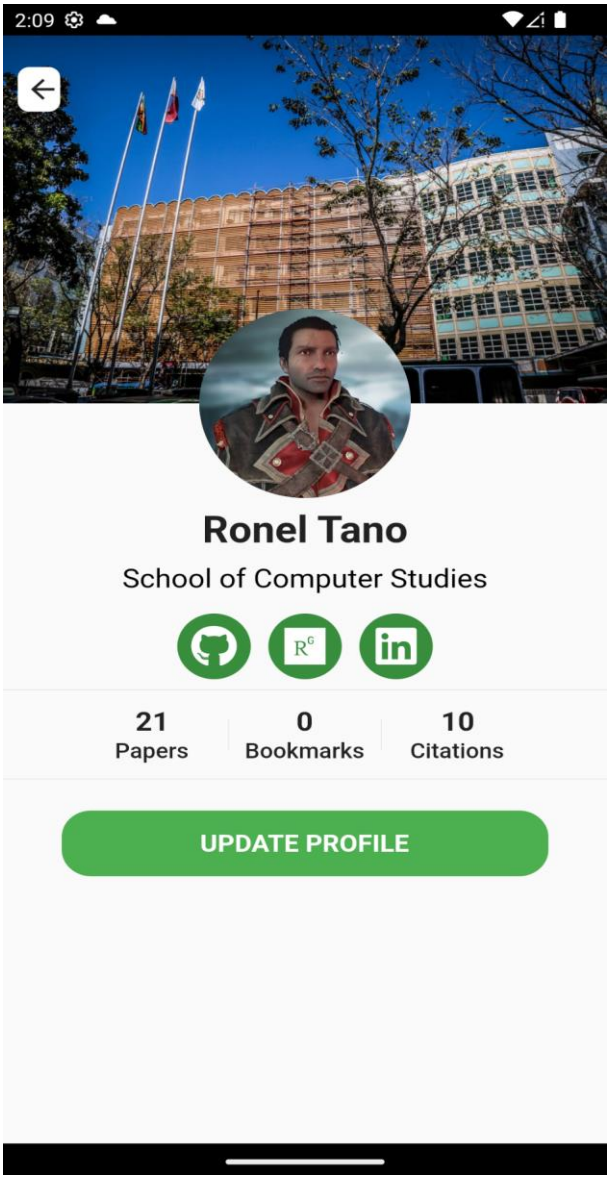
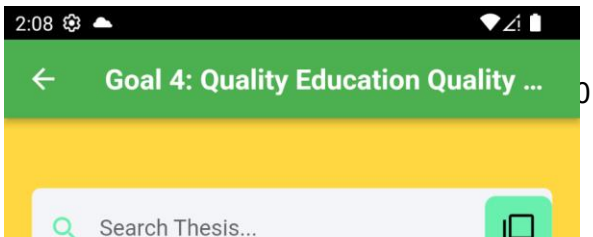


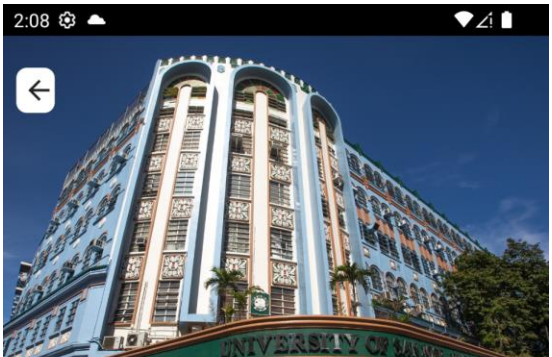
Figure 26 Profile Page

On the left side is the main dashboard of the mobile application, after login the user will be redirected to this screen. This screen shows all the 17 UN SDGs. Here users can also search for a particular research or sort papers based on categories.

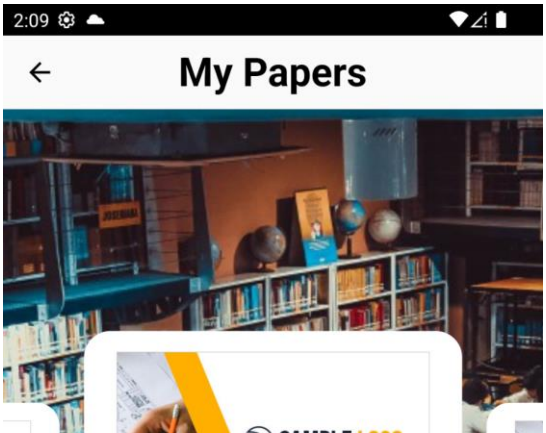
On the right side is the user's information as well as the number of research papers and bookmarks. Users can also update their profiles by changing their names.

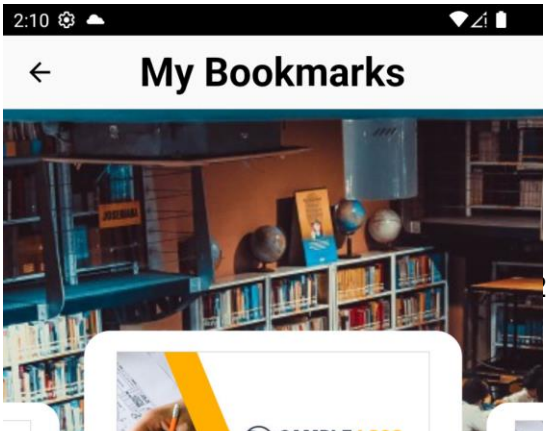


On the left side screen is shown when the user taps on a particular SDG icon. This screen contains the SDG name and description as well as a button to redirect users to research papers that are part of the goal. And in the right screen shows the dashboard when users click the button for viewing related papers from a particular SDG. This lists all research papers that are classified to that SDG.



On the left side screen is shown when users click a research paper. The user can view the title, goals, and abstract, users can also view the PDF file. There's also a button for bookmarks. And the right-side screen allows the user to view and read the pdf file for a particular research paper.





On the left side screen shows all the user's research papers. Users can navigate and choose which of the research papers the user wants to view. And the right screen allows the users to view their bookmarked research papers.

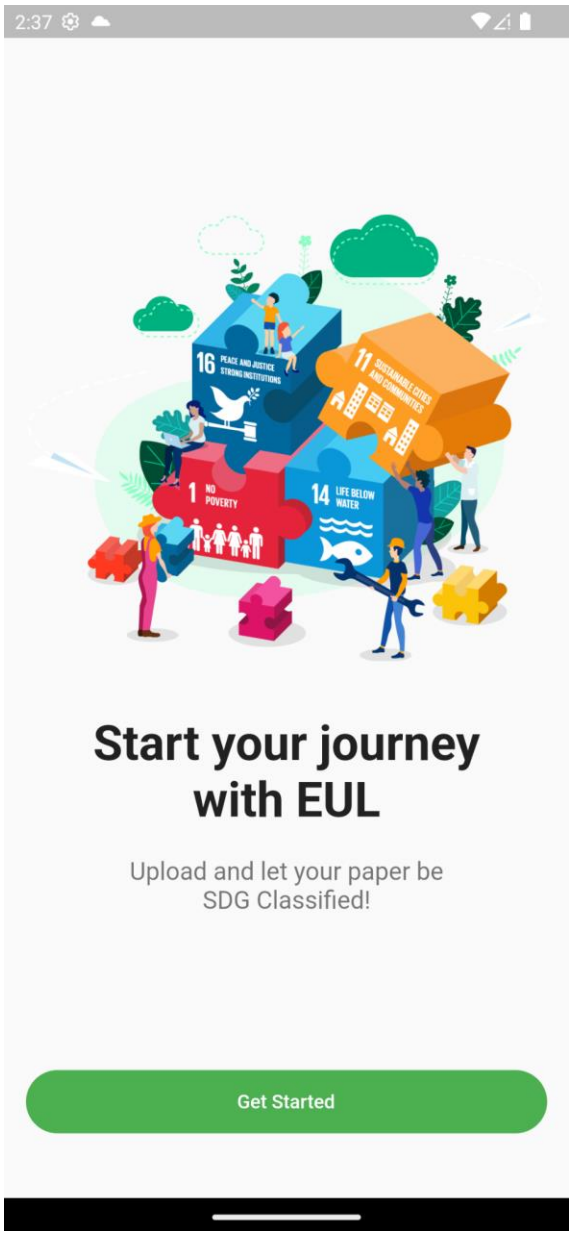


Figure 31 Upload Page

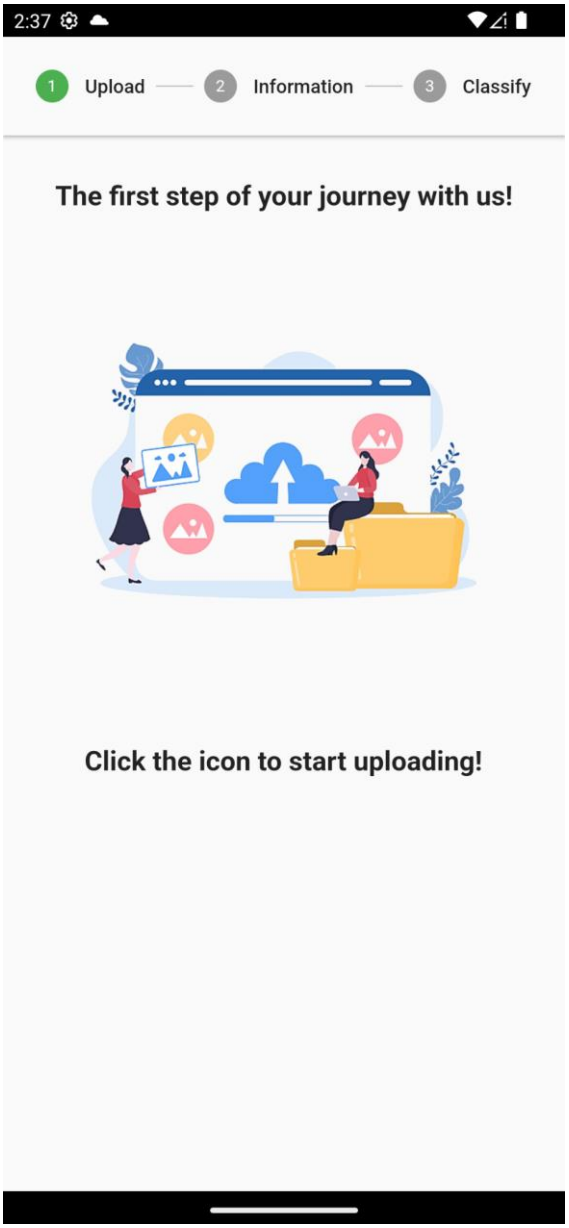
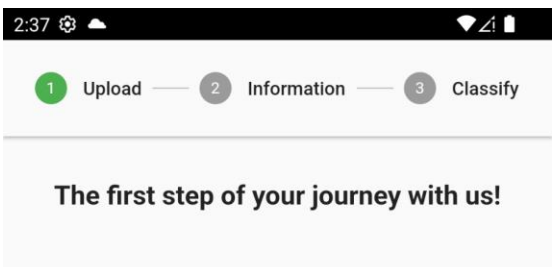
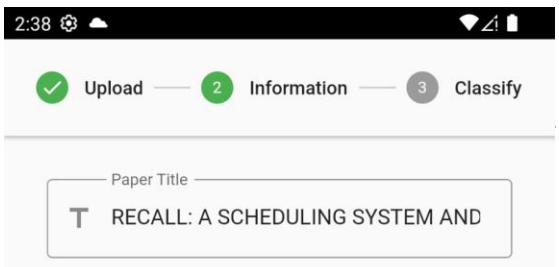


Figure 30 Upload Page

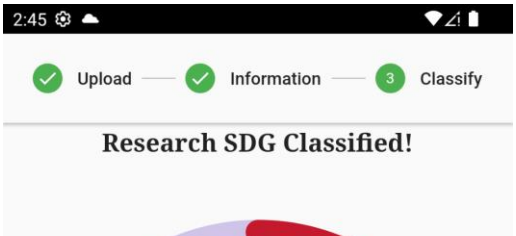
This screen is shown when the user taps the upload button in the main dashboard. Here the user is introduced to the upload process of EUL. This screen shows how the user can upload a PDF file into the system. Users can tap the image to start uploading research papers.





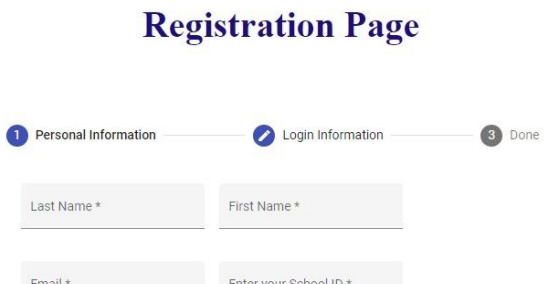
On the left side of the screen shows when the user picks a pdf file from their phone. If fetched from the system successfully a new button will appear to proceed to the next step.

On the left side of the screen shows to the user once the paper uploaded will be accepted by the system. This screen shows the extracted text from the uploaded file.



This screen is shown after the user clicks the button to classify research. This displays the top 4 goals predicted by the algorithm. The uppermost shows the highest score of all the goals.

Web Application



This shows the registration page of the web application. Here users can input their information including their school ID.

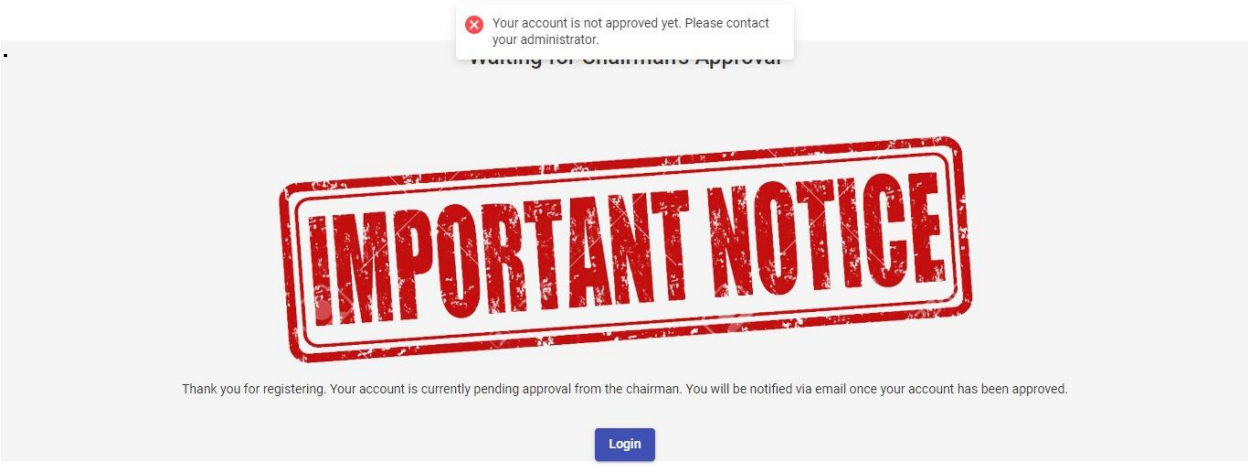


Figure 36 Pending Approval Screen

Once registered, users can't access yet their account. It must be approved first by the chairman of their school. This gives another layer of security for the application.

Figure 37 Web Dashboard

Once the user is approved for login, they will then be able to log in and be redirected into this screen. Here users can search, upload, view library, and view all the user's uploaded research papers.

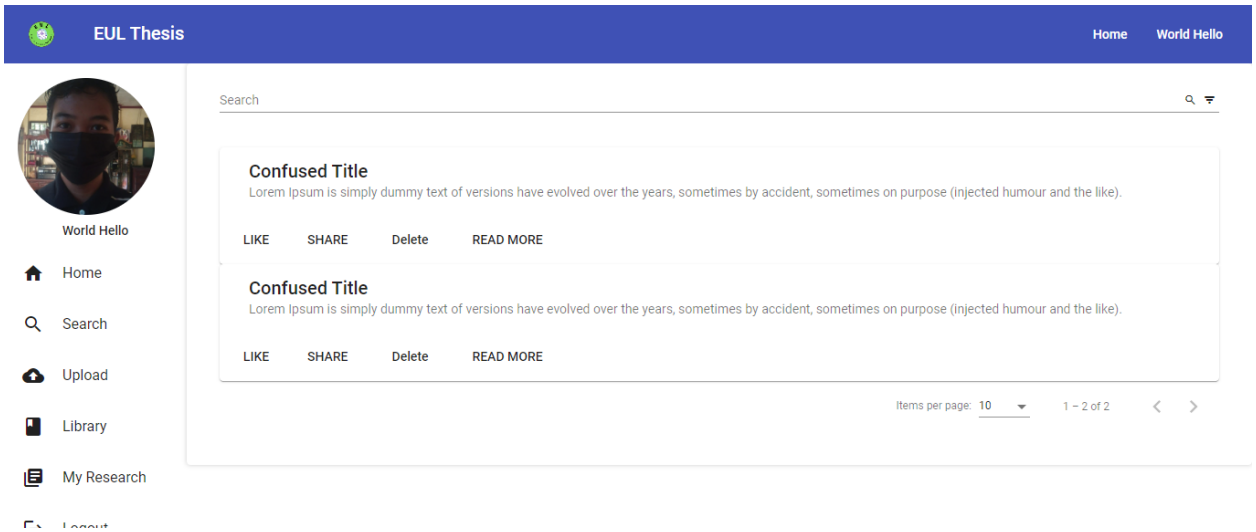
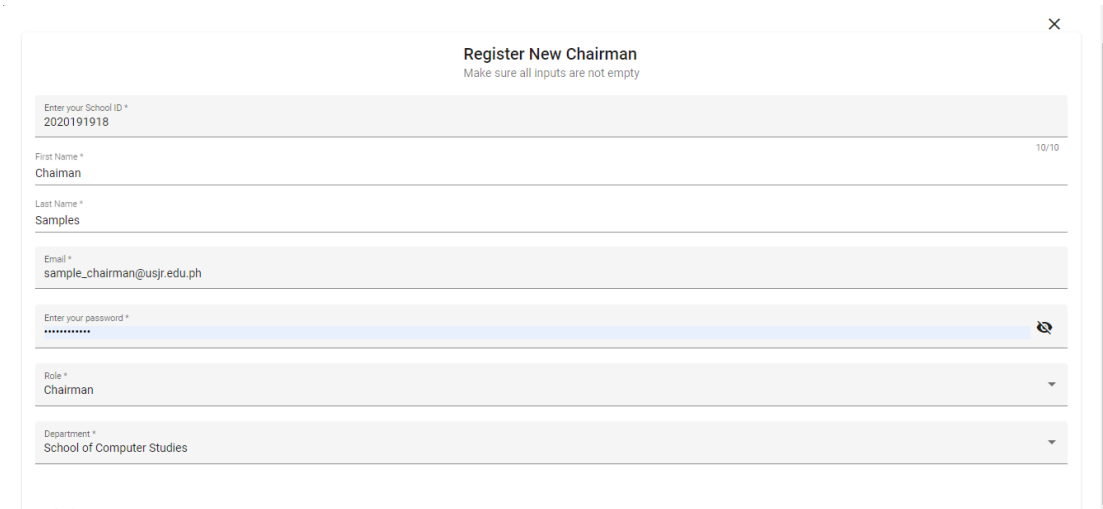


Figure 38 Web Search Screen

Users can search research papers based on different filters: Name, Department, or Goals.



Admin view for adding chairman account. Admin can add chairman account for a particular school. This decentralized the role of the admin and gives more emphasis on department-based decisions.

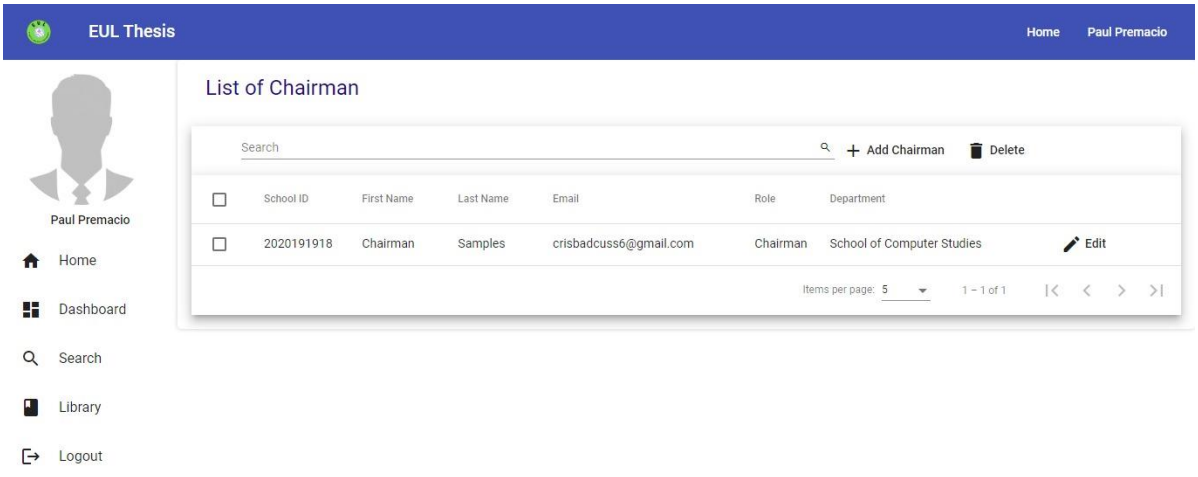
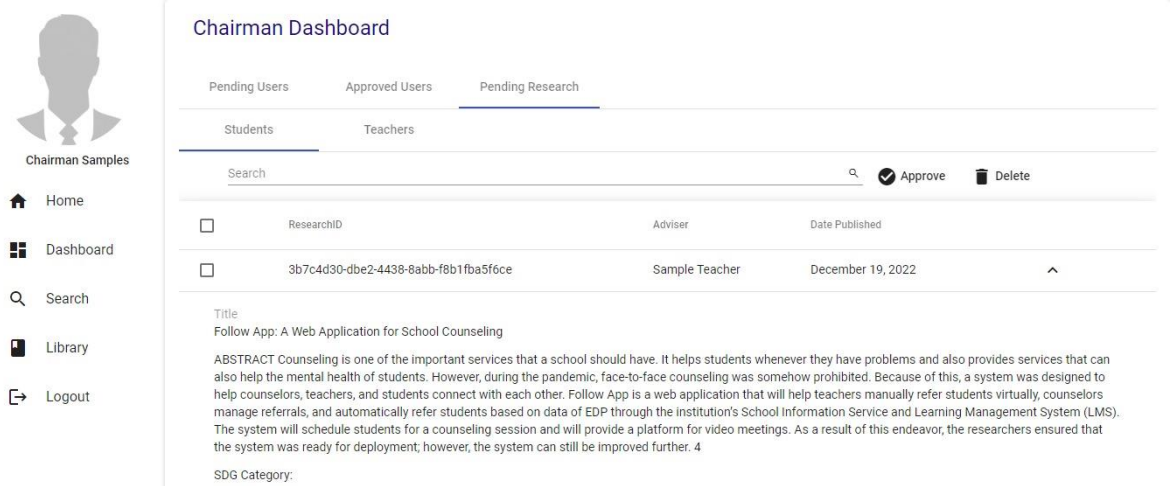


Figure 40 Admin View for List of Chairman

Admin view for adding chairman account. Admin can add chairman account for a particular school. This decentralized the role of the admin and gives more emphasis on department-based decisions.

This screen allows the admin to view all created chairman accounts. Here admin can perform CRUD operations for the account.



This is the chairman screen for approving research papers, here the chairman can view the uploaded research, and decides whether to approve the paper or not.

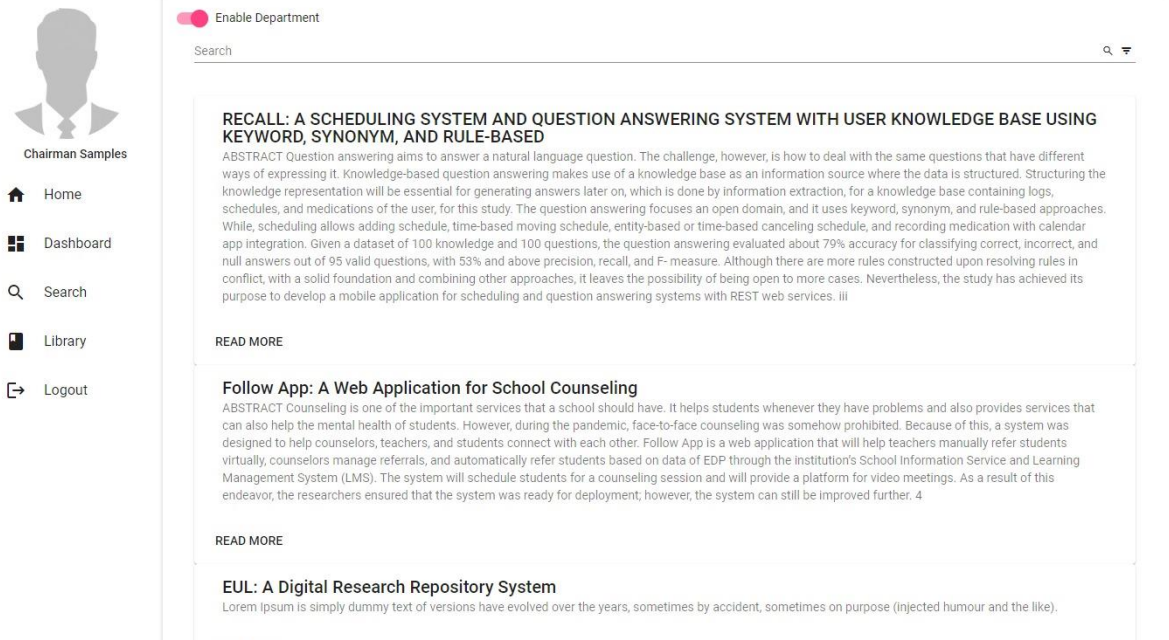


Figure 42 Search with Filter Chairman

Chairman accounts can search research papers from other departments. They can also disable the filter and only show papers from their respective departments.

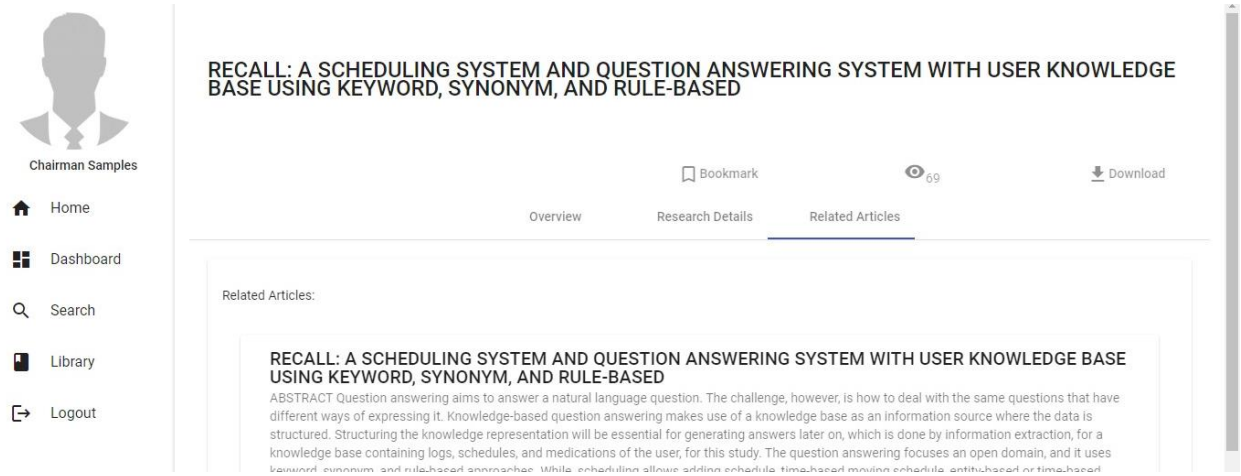
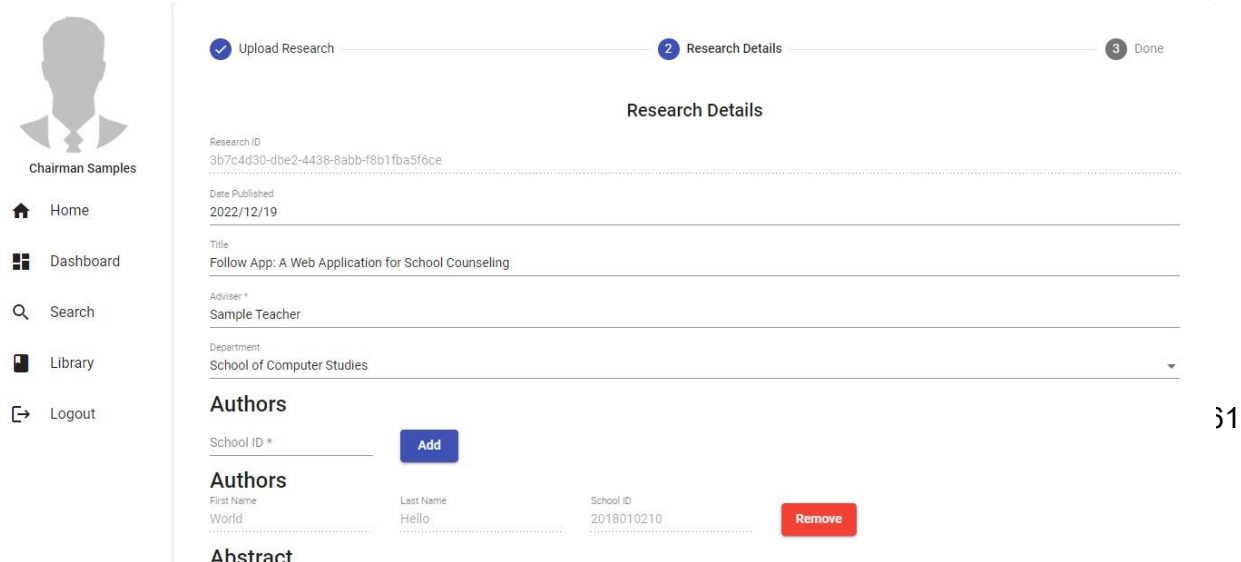
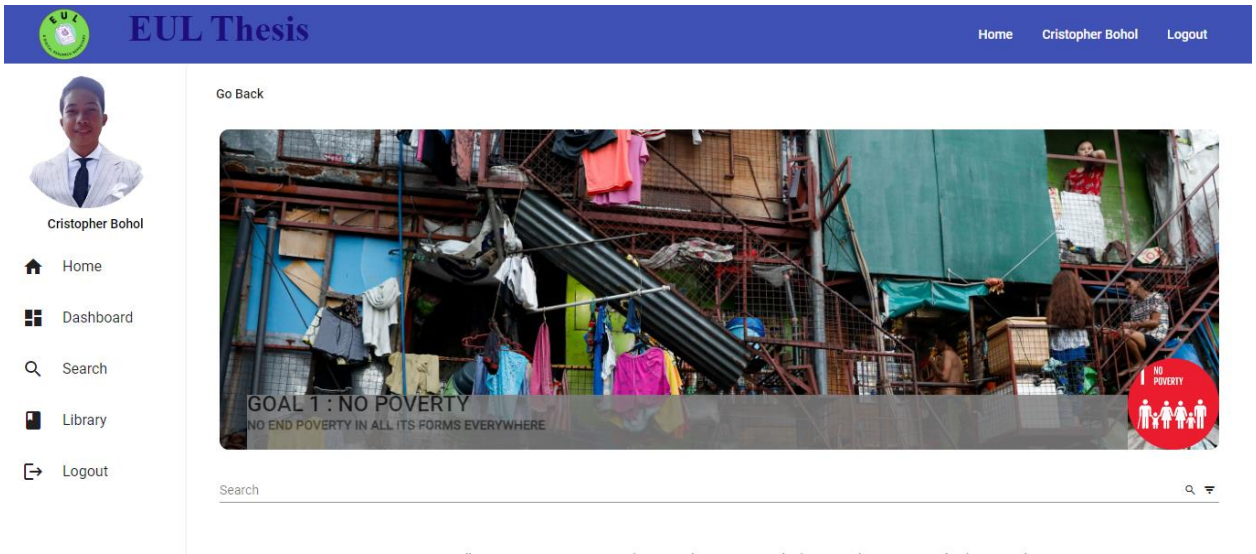


Figure 43 View Related Articles

This is where users can view papers that are related to the current viewed paper. These articles are based on the SDG categories.



This screen allows the users, chairman, students, or teachers, to upload their paper. This screen already displays the SDG Goals and the extracted information.



This is the screen when user clicks one of the goals in the dashboard. Here research papers from this goal are displayed with different filters.

CHAPTER III

SOFTWARE DEVELOPMENT AND TESTING

This chapter describes the implementation of the project in development, the various tools used to create and run the application, and testing methods to evaluate the question-answering results. It contains sections for the Development and Testing Process.

Development Software Platforms, Development Environments, and Tools

The system can be accessed via mobile and web platforms. Each platform has its own technologies implemented. The process is a hybrid of Agile and Prototyping Models. An initial prototype is created and refined as the development progresses.

For the web platform, these are the tools and development environment used:

- Visual Studio Code - a lightweight IDE commonly used by developers. This is used to run the REST API services and the web frontend.
- Node.js - a JavaScript framework used in this system as the backend environment.
- Angular 14 - a JavaScript framework used by this system as its front-end technology.

For the mobile platform, these are the tools and development environment used:

- Flutter Framework 3.3.7 - a relatively new framework for developing mobile, web, and desktop applications developed by Google. This framework uses Dart as its main language.

- http v0.13.4 - an open-source package by Flutter used primarily in handling http requests.
- json_serializable v6.3.1 - an open-source package for easy handling of JSON, for example converting JSON to Maps, or Maps to JSON.
- path_provider v2.0.11 - an open-source package for accessing and finding commonly accessed locations in the file system.

- Android Studio version 2021.2 - is used by this system to run the emulator for debugging and testing.
- Visual Studio Code - the primary IDE used by the developer for fast and smooth development.

For the Python Backend, these are the tools and development environment used:

- Visual Studio Code - IDE used for developing and running Python scripts.
- python 3.10.5 - the language used for creating the algorithms used by this system.
- Python Flask 2.2.3 - a Python framework for building web applications. This system used the framework in serving the Python scripts to be used by the web and mobile platforms in the form of http requests.
- nltk 3.7 - Natural Language Toolkit, a package used for handling all features related to natural language processing like lemmatization, POS Tagger, etc.
- pandas 1.4.3 - a python library for handling data analysis and structures. This library is used by the system primarily for converting dictionaries to CSV files.
- pdfplumber 0.7.5 - a Python library for handling all tasks related to PDF files.
- pytesseract 0.3.10 - a Python library used for handling OCR tasks.
- numpy 1.24.3 - a Python library used for handling mathematical operations on arrays. It is used for efficient calculations on arrays.

For Database Server:

- MySQL Workbench - digital filing cabinet for information, helps create, edit, view tables, run queries, manage user accounts, and server settings. Useful for database admins and developers.

DEVELOPMENT PROCESS

Preparing the Data for Training

The study uses a supervised learning algorithm to classify papers thus the first step is to carefully chose which papers fetched from google scholar or other trusted research portals to include in the training set. Here, it is already manually renamed to which goal it belongs.

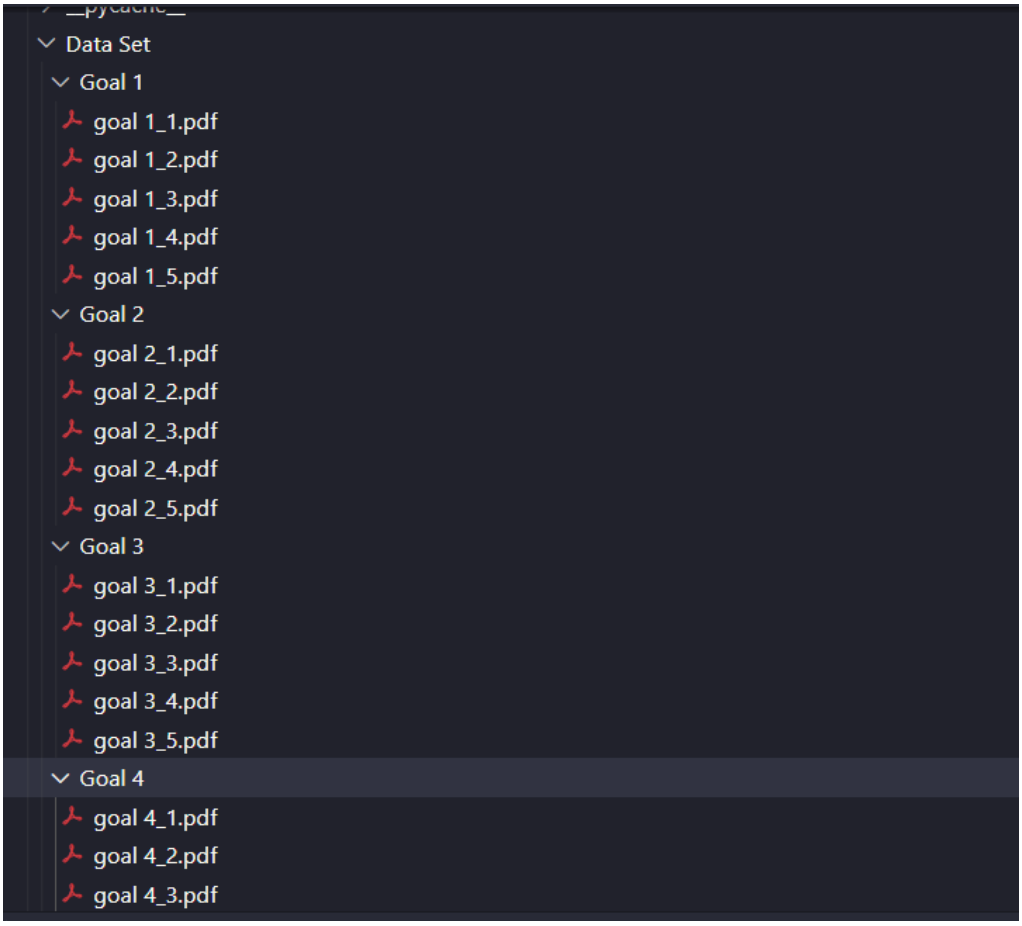


Figure 46 Directory of the Training Set with prelabeled PDF files

```
def extractAllPDF(self, goal):
    directory = (glob.glob("tfidf/Data Set/" + goal + "/*.pdf"))
    extractedText = " "
    finalText = " "
    for file in directory:
        with pdfplumber.open(file) as pdf:
            count += 1
            print(goal + " PDF #: " + str(count))
            for page in pdf.pages:
                extractedText = page.extract_text()
                print("Words Length: " + str(len(extractedText)))
                finalText = finalText + extractedText
    return finalText
```

Figure 47 Function for opening the directory on which the training data are located.

The function will automatically navigate to the defined directory with the name of the goal. Ex. tfidf/Data Set/Goal 1/. After locating the goal's folder, the function identifies which of the files inside the goal folder is a pdf. All the files are already in PDF format so everything inside will be used for extraction.

Figure 48 Goal 1 Folder with all its PDF files

It will iterate over each of the files, extract all strings inside and store it into the finalText variable. This is repeated until the function reaches the final folder which is Goal 17.

```
Goal 1 PDF #: 1
Words Length: 2770
Words Length: 1756
Words Length: 2845
Goal 1 PDF #: 2
Words Length: 3290
Words Length: 2928
Words Length: 2037
Goal 1 PDF #: 3
Words Length: 2413
Words Length: 3726
Words Length: 3578
Goal 1 PDF #: 4
Words Length: 4380
Words Length: 2503
Words Length: 3818
Goal 1 PDF #: 5
Words Length: 2885
Words Length: 2124
Words Length: 1774
```

Figure 49 Word Counts per PDF

This visualizes the word count for each PDF file inside the Goal 1 Folder. The PDF files are trimmed to only contain 3 pages each to maximize computational efficiency.

```
    Human SettlementsThe Evolution of Urban Water Management
To effectively address the emerging challenges in water management in cities, it is instructive to first
understand the historical evolution of urban water management regimes (i.e. infrastructure and
institutions). Traditionally, urban water has been managed in a technocratic way, based on principles of
predictability and control. Brown et al. (2009) investigated the evolution of urban water management
in cities over the last 200 years and considered a series of sustainable futures perspectives.5 As shown
in Figure 1, they developed a typology of six dominant water management regimes that represent a
nested continuum of socio-political drivers and service delivery responses, i.e. water supply, sewered,
drained, waterways, water cycle and water sensitive cities. This framework has been used to inform the
development of many urban water management strategies in Australia, (notably Water for Victoria6,
South Australia's Water for Good7), and internationally (e.g. Asian Development Bank (2013)8).
Figure 1: Evolution of Urban Water Management Regimes
(source: Asian Development Bank (2013) adapted from Brown et al 2009)
The first three regimes represent the historical development of water servicing in response to the need
to provide (i) clean and reliable water supplies, (ii) better public health outcomes and (iii) protection
from flooding. In these regimes, water services are provided through large, centralised infrastructure
and the associated yet siloed administrative/governance systems managed on the community's behalf
by utilities. The corresponding urban water infrastructure in almost all developed cities exhibit these
characteristics, with low community water literacy9 and urban water services largely invisible and typically
taken for granted.
Managing the water cycle in this segmented and linear way, whereby wastewater and stormwater
are swiftly channeled outside of the city and into receiving waterways, has given rise to a range of
unintended consequences, including environmental degradation.
5  Brown, R.R., Keath, N., & Wong, T.H.F. (2009), 'Urban water management in cities: historical, current and future regimes', Water, Science and
Technology, 59(5), 847-855.
6  http://delwp.vic.gov.au/water/a-new-water-plan-for-victoria
7  www.environment.sa.gov.au/files/.../water/water-for-good-full-plan.pdf
8  http://www.adb.org/sites/default/files/publication/30190/asian-water-development-outlook-2013.pdf
9  http://watersensitivecities.org.au/are-australians-water-literate/
3
A Framing Paper for the High-Level Panel on Water     This overwhelming demand on natural capital has left many cities with a reduced environmental capacity
to assimilate and process pollution, which in turn compromises water supply security (especially for
downstream urban environments) and urban liveability. The existing legacy of environmental damage
is especially problematic, with the majority of developed cities characterised by an ecological footprint
much larger than the physical footprint of the city.10 These challenges are further exacerbated by rapidly
increasing urban population that is not well supported by aging infrastructure and inadequate ongoing
investment in its augmentation, and the unpredictable and diverse nature of climate change impacts,
causing drought in some places (e.g: many Australian, Indian and Brazilian cities etc in recent times)
and severe flooding in others (e.g: the recent floods in Jakarta, Chennai, Houston, cities in Great Britain,
just to name a few).
The next two regimes, the Waterways and Water Cycle cities, reflect the response of urban water
systems (particularly in the developed world) to new challenges and socio-political drivers characterised
```

Figure 50 Raw Data

After going through all the pdf files in the goal's folder. This is what the raw extracted text looks like. As expected of raw data, it still contains a lot of unnecessary text.

Once this is done, the result will then be stored on a list. This process will be repeated until all goals are scraped and stored on a list.

```
for goal in goals:
    trainingData = self.extractAllPDF(goal)
    trainingDocs.append(trainingData)
```

Figure 51: Iterate each goal

The trainingDocs list represents the goals in the corpus, this is still raw data and has not been preprocessed yet.

Once all goals are appended to the list, the system performs text preprocessing.

```
preProcessedDocs = self.preprocess_documents(documents)
unique = self.getUniqueWords(preProcessedDocs, False)
```

Figure 52 Preprocessing and Getting Unique Words

The variable documents represent the list of unprocessed text from each goal. The list contains 17 items as there are 17 goals in the entire corpus.

```
def preprocess_documents(self, docs):
    stop_words = set(stopwords.words("english"))
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = []
    for doc in docs:
        tokens = word_tokenize(doc)
        filtered_tokens = [
            token.lower() for token in tokens if token.isalnum() and token not in stop_words]
        lemmatized_tokens.append([lemmatizer.lemmatize(
            token) for token in filtered_tokens])

    return lemmatized_tokens
```

Figure 54: Preprocessing Documents Function

Preprocessing includes removing stop words, punctuation, and numbers, converting to lowercase, and lemmatization. All of these are done with the help of the nltk library. Each of the items in the list of unprocessed text represents each goal, and this function iterates over that list and performs text preprocessing. Once done, the system will create a new list of preprocessed data.

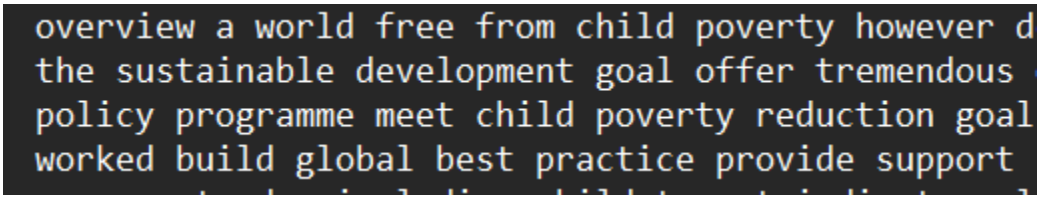


Figure 53 Snippet of Preprocessed Data

After preprocessing, the text is now normalized and ready to be used for the next step.

```
def getUniqueWords(self, preProcessedDocs, type):
    unique = {}
    for str in preProcessedDocs:
        for str2 in str:
            unique[str2] = 0
    return unique
```

Figure 54 Snippet of Preprocessed Data

The next step for training the model is the identification of the unique words in the entire 17 documents. These words represent the feature set.

	Value
overview	0
a	0
world	0
free	0
from	0
...	...
flower	0
color	0
categorize	0
venation	0
optical	0

Figure 55 Dictionary of Unique Words

These unique words are then stored in a dictionary data structure for easy retrieval and storage.

Text Vectorization

After creating the dictionary of unique words, the next step is the calculation of the TF-IDF. The first step in this process is to create the Term Frequency of each word with respect to its parent goal. All words in Goal 1 have their own Term Frequency, the same as Goal 2 until Goal 17.

```
for listOfTokens in preProcessedDocs:
    tf.append(self.getTermFreq(
        unique, len(listOfTokens), listOfTokens))
    tv.append(self.getTerm(unique, len(listOfTokens), listOfTokens))
```

Figure 56 Iterating over the entire Preprocessed List

The formula for term frequency is word frequency / total count of words in the document. In this case, Goal 1 has a total count of 4019 words.

```
            Value
overview        2
a               7
world          25
free            3
from            4
...           ...
flower          0
color           0
categorize      0
venation        0
optical         0

[10393 rows x 1 columns]
Total Number of Words in Goal 1:  4019
```

Figure 57 Term Vectors for Goal 1

This visualizes the term vectors of goal 1. Each word is assigned a value that represents the occurrence of that word in the Goal 1 document.

	Value
overview	0.000498
a	0.001742
world	0.006220
free	0.000746
from	0.000995
...	...
flower	0.000000
color	0.000000
categorize	0.000000
venation	0.000000
optical	0.000000

Figure 58 Term Frequency for Goal 1

After getting the occurrence of the words in Goal 1, the system will proceed in calculating its term frequency. By applying the formula mentioned before, the result will look like this. There are words that have zero values because these words are not present in the Goal 1 document.

```
def inverse(self, unique, preProcessedDocs, tf=[{}]):
    num_of_docs = len(preProcessedDocs)
    idf, finalIDF = {}, {}
    for str in unique:
        idf[str] = 0
    for str in idf:
        for str2 in tf:
            if str in str2:
                idf[str] = idf[str] + str2[str]
    for str in idf:
        finalIDF[str] = math.log10(num_of_docs / idf[str])
    return finalIDF
```

Figure 59 Code Snipper for IDF

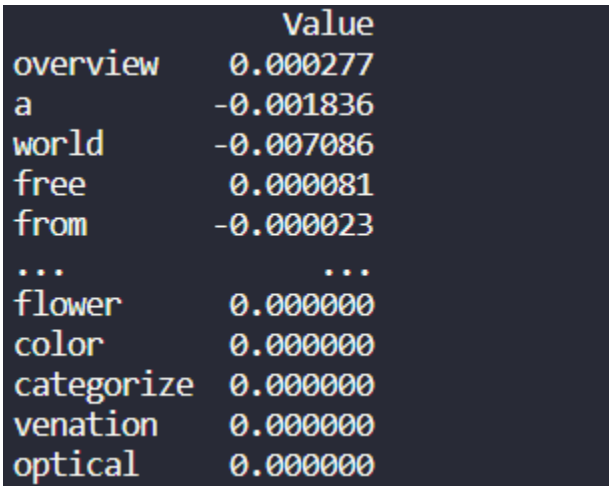
After getting all Term Frequencies in the entire list of processed data, the next step is getting the inverse document frequency.

	Value
overview	0.556303
a	-1.054358
world	-1.139179
free	0.109144
from	-0.023481
...	...
flower	1.255273
color	0.954243
categorize	1.255273
venation	1.255273
optical	1.255273

Figure 60 Result of Inverse Document Frequency for Goal 1

The formula for IDF is log(N/DF), where N is the total number of documents, and DF is the document frequency.

Inverse Document Frequency is a measure that helps identify the relevance of a particular term in the entire collection of documents. The lesser the value, the more common it is, and the higher, the more rare or unique the word is. As in the image snippet, the term *world* has an IDF of -1.139179, while the term flower has an IDF value of 1.2555, this means that the term *world* appears more frequently across the 17 documents and is very irrelevant, whilst the term flower appears more frequent across all documents.



	Value
overview	0.000277
a	-0.001836
world	-0.007086
free	0.000081
from	-0.000023
...	...
flower	0.000000
color	0.000000
categorize	0.000000
venation	0.000000
optical	0.000000

Figure 61 Final TF-IDF Values

After getting the IDF values, the final step is to get the Final TFIDF, which has a very straightforward calculation, TF x IDF. The image represents the TFIDF value for Document 1 which in this case is Goal 1, the term overview has a value of 0.000277, while terms like flower, color, etc. have values of 0, which means that these values don't have any relevance to Goal 1.

```
def calculateTFIDF(self, listofDict, idf, tf_idf):
    temp = {}
    count = 1
    for list in listofDict:
        temp = list
        for features in idf:
            if temp.__contains__(features):
                temp[features] = temp[features] * idf[features]
        tf_idf.append(temp)

    return tf_idf
```

Figure 62 TF-IDF calculation

The last process is to store the final TF-IDF for the entire SDG corpus into a CSV file. This CSV file visualizes how the TFIDF looks after appending all the 17 documents.

	overview	a	world	free	from	child	poverty	however	despite	urgency	availability	proven	approach	measure	respond	received	relatively	little	attention	global	struggle	the
0	0.000277	-0.00184	-0.00709	8.15E-05	-2.34E-05	-0.01535	-0.0491	-0.00134	-0.00037	0.000237	-0.00022	0.000119	-0.00145	-0.00191	0.000175	-5.00E-05	-6.22E-05	7.49E-05	-0.00014	-0.0079	0.000475	-0.0221
1	0	-0.00267	-0.00173	0	0	-0.00555	-0.00132	-0.00151	-0.00012	0	-5.08E-05	8.05E-05	-0.00098	-0.0006	0	-2.26E-05	-0.00013	0	-3.24E-05	-0.0024	0	-0.00916
2	0	-0.00504	-0.00306	0	0	-0.00031	0	-0.00214	0	0	-8.99E-05	0.000143	-0.00087	-0.00053	0	0	0	8.99E-05	-5.73E-05	-0.00098	0	-0.01867
3	8.65E-05	-0.00279	-0.00284	1.70E-05	-1.10E-05	-0.00959	-0.0014	-0.00251	-0.00063	0.000148	-4.68E-05	0	-0.00136	-0.00018	5.48E-05	0	-1.94E-05	0	-8.95E-05	-0.00358	0	-0.01867
4	0	-0.00686	-0.0121	3.74E-05	-8.04E-06	-0.00247	-0.00691	-0.00031	0	0	0	0	0	-0.0002	0.000121	-2.29E-05	0	0.000103	-6.57E-05	-0.0015	0	-0.02704
5	0	-0.00299	-0.00539	1.72E-05	-7.40E-06	-0.00049	-0.00018	-0.00212	-5.80E-05	0	-4.74E-05	0	-0.00023	-0.00019	0	-3.17E-05	-5.91E-05	4.74E-05	0	-0.00138	0	-0.01426
6	0	-0.00167	-0.00253	0	0	-0.00033	-0.00036	-0.00227	0	0	-9.54E-05	0	0	-0.00093	0.000223	0	-3.96E-05	0	-6.08E-05	-0.00625	0	-0.01563
7	0	-0.00269	-0.00087	2.79E-05	-1.20E-05	-0.00184	-0.00143	-0.00252	-0.00028	0	0	0	-0.0013	-0.00226	0	-1.71E-05	-6.38E-05	7.68E-05	-0.00024	0	0	-0.0214
8	0.000159	-0.0012	-0.00228	0	0	0	-0.00128	-0.00179	0	0	-8.60E-05	0	-0.00166	0	0.000101	-3.83E-05	0	0	-0.00011	-0.00031	0	-0.01739
9	0	-0.00285	-0.00476	5.37E-05	0	-0.00152	-0.00221	-0.00066	-0.00072	0	0	0	0	0	0	0	-6.14E-05	0	-9.44E-05	-0.00188	0	-0.01213
10	0	-0.00265	-0.00597	0	-1.97E-05	-0.00043	-0.00094	-0.00113	-7.71E-05	0	-0.00025	0	-0.00274	-0.00037	0	-5.61E-05	-2.62E-05	6.31E-05	-8.04E-05	-0.00413	0	-0.01447
11	0	-0.00408	-0.00206	0	0	0	0	-0.00092	-0.00019	0	-0.00016	0	-0.00112	-0.0003	0	-1.73E-05	-3.22E-05	0	0	-0.00621	0	-0.01738
12	0.000119	-0.00293	-0.00488	0	0	0	0	-0.00077	-0.00016	0	-0.00019	0.000102	-0.00062	-0.00013	0	0	-0.00011	0	-4.11E-05	-0.0082	0	-0.01689
13	0	-0.00233	-0.0028	0	0	-0.00025	-0.00028	-0.00132	-0.00018	0	-0.00015	0	-0.00036	0	0	0	-3.07E-05	0	0	-0.0035	0	-0.02262
14	0	-0.00255	-0.00138	0	-4.72E-06	-0.00021	-0.00045	-0.00325	-7.40E-05	0	-0.00073	0	-0.00088	-0.00047	0	-2.69E-05	0	0.000182	-7.72E-05	-0.0033	0	-0.01092
15	0	-0.00148	-0.00124	8.51E-05	-7.33E-06	-0.00176	-0.00035	-0.00126	-0.00017	0	0	0.000149	-0.00102	-0.0011	5.49E-05	-2.09E-05	0	0	-0.00012	-0.00051	0	-0.02283
16	0	-0.00362	-0.00554	0	0	0	-0.00032	-0.00205	-0.00021	0	-0.00034	0	-0.00021	-0.00017	0	0	0	0	0	-0.00439	0	-0.01647

Figure 63 TF-IDF in CSV format

In the CSV file, each row represents each goal in SDG. Here we can visualize properly the relevance of a particular word in the entire SDF corpus.

Information Extraction

Abstract, Introduction, and Research Methodology

After creating the TF-IDF for the entire SDG corpus, the system is now ready to accept new documents from users. Classifying papers based on everything it contains isn't ideal, as it will greatly affect the computational performance of the classifier. The solution for this is to extract sections of the paper that contain information that greatly describes the paper. Abstract alone isn't enough as there are papers that have fewer words in the abstract. To augment this, the Introduction and Research Methodology are also extracted.

```
def main_logic(self, filename):
    appendedData = ""
    abstract = self.getFromPDFAbstract(filename)
    introduction = self.getFromPDFIntro(filename)
    method = self.getFromPDFMethod(filename)
    appendedData = abstract + introduction + method
    return {'abstract': abstract, 'introduction': introduction, 'method': method, 'appendedData': appendedData}
```

Figure 64 Extracting Data Set for Classifying New Document

It is also important to remember that not all papers in the institution contain all the sections, some papers only have an Abstract, and an Introduction but no Research Methodology. As long as both the Abstract and the Introduction are present, the classifier will still predict accurately the SDG labels.

```
def getFromPDFAbstract(self, filename):
    count = 1
    finalText, final_abstract = " ", " "
    limitPages, currentPage = 10, 0
    with pdfplumber.open('assets/upload/' + filename) as pdf:
        for page in pdf.pages:
            extractFromPDF = page.extract_text()
            finalText = finalText + extractFromPDF
            checkAbs = self.getAbstract(finalText, count)
            if (checkAbs):
                final_abstract = finalText
                final_abstract = self.cleanString(final_abstract)
                break
            if (currentPage == limitPages):
                break
            count += 1
            currentPage += 1
            final_abstract = " "
            finalText = " "
    return final_abstract
```

To improve the computational time in extracting the Abstract, the search is limited to only 10 pages as most Abstract in the institution's format can be found around these sections.

```
def getAbstract(self, processedText, page):
    count = 0
    pageAbstract = 0
    abstract = False
    if (("ABSTRACT" in processedText or "Abstract" in processedText)
        and ("TABLE OF CONTENTS" not in processedText and "Table of Contents" not in processedText)):
        if (count == 0):
            abstract = True
            pageAbstract = page
        count += 1
    return abstract
```

Figure 66 Logic for Extracting Abstract

Extracting the Abstract is a straightforward method since papers contain a table of contents that has the word Abstract, the function will check if the word Abstract is in the string, and if found, it will check if the word table of contents is not in the string. If both conditions (Abstract in String and Table of Contents not in String) are True, then that specific string is an Abstract.

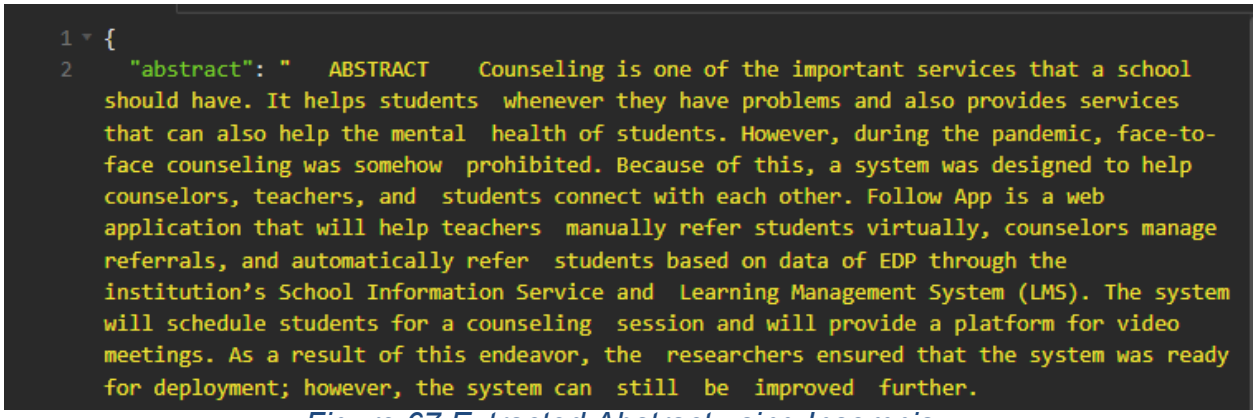


Figure 67 Extracted Abstract using Insomnia.

Some abstracts may contain noisy texts and may need to be cleaned but if the format is in-lined with the institution's standard there won't be any problem extracting.

```
def getFromPDFIntro(self, filename):
    count = 1
    finalText, final_intro = " ", " "
    limitPages, currentPage = 10, 0
    with pdfplumber.open('assets/upload/' + filename) as pdf:
        for page in pdf.pages:
            extractFromPDF = page.extract_text()
            finalText = finalText + extractFromPDF
            checkAbs = self.getIntroduction(finalText)
            if (checkAbs):
                final_intro = finalText
                final_intro = self.cleanString(final_intro)
                break
            if (currentPage == limitPages):
                break
            count += 1
            final_intro = " "
            currentPage += 1
            finalText = " "
    return final_intro
```

Figure 68 Handler for Extracting Introduction

The process of extracting the Introduction is like the Abstract. Limit the pages to scrape to 10.

```
def getIntroduction(self, processedText):
    count = 0
    introduction = False
    if (("INTRODUCTION" in processedText or "Introduction" in processedText)
        and ("TABLE OF CONTENTS" not in processedText and "Table of Contents" not in processedText)):
        if (count == 0):
            introduction = True
        count += 1
    return introduction
```

The logic for extracting the introduction is also like that of the abstract extraction.

```
  "introduction": "   CHAPTER I      INTRODUCTION    Rationale of the Study   The Guidance
Office plays a vital role in every school. The guidance  office  comprises guidance
counselors in charge of counseling students whenever the latter  encounter personal and
academic-related problems and conflicts. The function of school  counselors in ensuring
student achievement is fundamental. (Lapan, Gysbers, & Kayson,  2007; Stone & Dahir,
2006).  Most counseling in schools is done face-to-face through teachers' referrals or
individual appointments in which the students need to go to the office to have the
counseling session. At the University of San Jose-Recoletos (USJ-R), students are given
the appointment slip and are called to the guidance office for their session, and then
feedback is given to each teacher. But during the pandemic, the system of counseling  has
changed. Due to the fact that there were restrictions for physical encounters, the
implementation of online classes has been introduced. In lieu of this, counseling session
has been shifted to online as well to meet the student's need even in the online set-up.
Follow App is a web application for counseling that will help teachers, counselors,  and
students have the session in an online set-up. The application focuses on scheduling
students for the session, allowing counselors to give feedback online and refer students
based on academic performance and behavior.  10   ",
```


Figure 70 Extracted Introduction using Insomnia.

This is the structure of the introduction after extracting, it still contains noisy texts but will be removed later after text preprocessing.

```
def getFromPDFMethod(self, filename):
    count = 1
    finalText = " "
    final_method = " "
    limitPages = 10
    currentPage = 0
    with pdfplumber.open('assets/upload/' + filename) as pdf:
        for page in pdf.pages:
            extractFromPDF = page.extract_text()
            finalText = finalText + extractFromPDF
            checkAbs = self.getMethodology(finalText)
            if (checkAbs):
                final_method = finalText
                final_method = self.cleanString(final_method)
                break
            count += 1
            if (currentPage == limitPages):
                break
            currentPage += 1
            final_method = " "
            finalText = " "
    return final_method
```

The handler is the same as the previous two extractions.

```
def getMethodology(self, processedText):
    count = 0
    methodology = False
    if (("Research Methodology" in processedText or "RESEARCH METHODOLOGY" in processedText)
        and ("TABLE OF CONTENTS" not in processedText and "Table of Contents" not in processedText)):
        if (count == 0):
            methodology = True
        count += 1
    return methodology
```

Figure 72 Logic for Methodology Extraction

The process for extracting the methodology is like those previous two extractions.

There are instances where there will be no methodology extracted and the data appended data will only be composed of the Abstract and the Introduction. The methodology in this case is an optional requirement.

Extraction of these data is done once the user uploads their approved research paper, but before the user can view the extracted information, the system will first check if the paper uploaded is an authentic approved research paper.

```
def acceptanceChecker(self, filename):
    start_time = time.time()
    go = False
    endorsement = " "
    if (self.checkPages(filename) >= 5):
        endorsement = self.endorsementExtraction(filename)
        if ("PASSED" in endorsement):
            go = True
        else:
            os.remove("assets/upload/" + filename)
    else:
        os.remove("assets/upload/" + filename)
    end_time = time.time()
    execution_time = end_time - start_time
    print("Execution time:", execution_time, "seconds")
    return go
```

The system will perform checking of the paper and it will scrape the file until it finds the endorsement page, if it returns true, the paper uploaded is approved and if it's false, the paper is rejected, and removed from the local directory.

Published Date, Title, and Department

The system incorporates a feature that automatically extracts research information from the user once a file is uploaded to the system.

```
def extract_text_from_pdf(self):
    # Get the directory containing the script
    script_dir = os.path.dirname(os.path.abspath(__file__))
    # Get the parent directory of the script directory
    parent_dir = os.path.dirname(script_dir)
    # Construct the full path to the PDF file
    pdf_path = os.path.join(parent_dir, "assets",
                            "upload", self.document_path)
    # Open PDF file in binary mode
    with open(pdf_path, 'rb') as pdf_file:
        # Create a PDFPlumber object
        pdf_reader = pdfplumber.open(pdf_file)
        # Extract text from the first page of the PDF
        page = pdf_reader.pages[0]
        pdf_text = page.extract_text()
        # Convert PDF page to image
        x0, y0, x1, y1 = page.cropbox or (0, 0, page.width, page.height)
        image = page.to_image(resolution=300)
        image_file = 'temp_image.png'
        image.save(image_file, format='png')
        # Specify the path to Tesseract executable
        tesseract_path = os.path.join(os.getenv('PROGRAMFILES'), 'Tesseract-OCR', 'tesseract.exe')
        # Perform OCR using pytesseract
        ocr_text = pytesseract.image_to_string(image_file)
        # Extract paragraphs from extracted text
        paragraphs = self.extract_paragraphs_from_text(ocr_text)
        os.remove(image_file)

        return paragraphs
```

Figure 74 Converting Cover Page to Image File

The first step in this process is to convert the cover page into an image format. The goal here is to provide better extraction results and more code readability. After converting the cover page into an image, the system calls the pytesseract. image_to_string to perform OCR. The role of the OCR here is just simply to extract all strings. This removes the unnecessary spacing provided by the default extractor, pdfplumber. Although the system is still using the default extractor for other extraction functionalities.

```
def process_extracted_text(self, input_text, fromNode):
    information = {}
    information['title'] = self.extract_title(input_text)
    information['department'] = self.extract_department(input_text)
    information['authors'] = self.extract_names(input_text, fromNode)
    information['published_date'] = self.extract_published_date(input_text)
    return information
```

Figure 75 Handler for Information Extraction

```
def extract_title(self, input_text):
    title = ''
    if len(input_text) > 0:
        # Extract the first item as the title
        title = input_text[0].strip()
    return title
```

As per the pattern in the format, research titles are usually found at the beginning of the cover page.

```python
def extract_department(self, input_text):
    departments = [
        'School of Law', 'School of Business and Management',
        'School of Computer Studies', 'Senior High School',
        'School of Arts and Sciences', 'RITTC',
        'School of Allied Medical Sciences',
        'School of Engineering', 'School of Education', 'College of Information Computer and Communications Technology'
    ]
    extracted_department = ''
    for text in input_text:
        for department in departments:
            if department in text:
                extracted_department = department
                break
        if extracted_department:
            break
    return extracted_department
```

Figure 77 Department Extraction

Extracting the department is also a straightforward approach and mostly involves rule-based identification. It contains the names of all the departments in the university.

```
def extract_published_date(self, input_text):
    extracted_date = None
    current_date = datetime.datetime.now()  # Get current date and time
    for text in input_text:
        extracted_date_str = re.findall(r'\b\d{1,2}/\d{4}\b', text)
        if extracted_date_str:
            extracted_date = date_parser.parse(
                extracted_date_str[0], fuzzy=True)
        if not extracted_date:
            date_formats = [
                '%B %d, %Y',
                '%B %Y',              # Month Year (e.g. March 2020)
                '%m/%Y',              # Month/Year (e.g. 03/2020)
                '%b %Y',
                # Month Year without slash (e.g. 03 2020)
                '%m %Y',
                'date %Y',            # Custom date format (e.g. date 2023)
                # Custom date format (e.g. June 20, 2022)
                '%B %d, %Y',
                # Custom date format with time (e.g.June 20, 2022 12:34)
                '%B %d, %Y %H:%M',
            ]
            for line in text.split('\n'):
                try:
                    # Attempt to parse date from line using date_formats
                    for date_format in date_formats:
                        parsed_date = date_parser.parse(line, fuzzy=True, yearfirst=True,
                                                        default=current_date)
                        if parsed_date:
                            extracted_date = parsed_date
                            break
                    if extracted_date:
                        break
                except ValueError:
                    pass
```

Figure 78 Date Extraction

The function will check if the string given is of a date-time format. The function also utilizes rule-based classification to check if the token is a date or not.

SDG Classification

Applying TF-IDF on the New Document

The KNN algorithm involves the creation of training/test data, identifying which distance/similarity metric to use, and majority voting by its neighbors or the K. One of the common distance metrics used in KNN is the Euclidean, Manhattan, Minkowski, and Hamming distance. The study focuses on the use of cosine similarity as its main similarity metric since this technique focuses more on the semantic meaning between two documents rather than their geometric proximity.

```
def classifyResearch(self, data, testing):
    trainingDocs, newDocs = [], []
    goals = ['Goal 1', 'Goal 2', 'Goal 3', 'Goal 4', 'Goal 5',
             'Goal 6', 'Goal 7', 'Goal 8', 'Goal 9', 'Goal 10', 'Goal 11', 'Goal 12',
             'Goal 13',
             'Goal 14', 'Goal 15', 'Goal 16', 'Goal 17'
             ]
    if (self.checkDataSet() == False):
        for goal in goals:
            trainingData = self.extractAllPDF(goal)
            trainingDocs.append(trainingData)
    else:
        trainingDocs = self.extractTraining()
        newDocs.append(data)
        newData = self.preprocess_documents(newDocs)
        data = newData[0]

    trainingDocs.append(data)
    values = self.getTFIDF(trainingDocs, testing)
    count += 1
    if (self.checkLastData()):
        self.removeNewData()
    result = self.getCosine(values, count)
    return result
```

Figure 79 Handler for Classifier

To avoid retraining the model every time there's a new document to classify, the system provides some rules to handle that scenario.

First, the system will check if the TF-IDF.csv file exists in the directory. If it returns false, that's the time the retraining will happen, and if not, the system will retrieve all the preprocessed data into a list. These preprocessed data are stored beforehand during the training of the model.

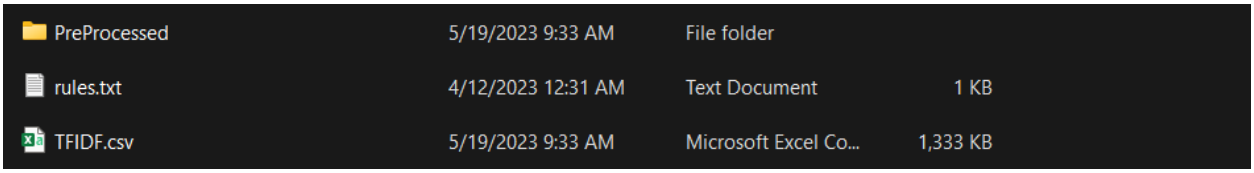


Figure 80 Directory of TFIDF.csv

```
def extractTraining(self):
    index = 17
    string = ""
    extractedTraining = []
    for i in range(index):
        with open(r"tfidf/Results/PreProcessed/PreProcessed " + str(i+1) + ".txt", 'r', encoding="utf8") as f:
            for line in f:
                string = line.split()
                extractedTraining.append(string)
                string = ""
    return extractedTraining
```

Figure 81 Retrieving Preprocessed Data

These data are stored separately based on their goals, in total, there are 17 text files in the directory that contains all these preprocessed data. This step plays a crucial role in decreasing the computational time of the classification as it skips the process of extracting raw data from all 17 goals and applying text processing.

```
else:
    trainingDocs = self.extractTraining()
    newDocs.append(data)
    newData = self.preprocess_documents(newDocs)
    data = newData[0]
```

Figure 82 New Document Preprocessing

The new document will be the only one that undergoes preprocessing. After preprocessing the new document. The newly uploaded document will be stored in the temporary 18th row of the SDG corpus.

```
def getTFIDF(self, documents, testing):
    tv, tf, final = [{}], [{}], [{}]
    index = 1
    if (self.checkDataSet() == False):
        preProcessedDocs = self.preprocess_documents(documents)
        unique = self.getUniqueWords(preProcessedDocs, False)
        for token in preProcessedDocs:
            self.writeListToTxt(' '.join(token), index)
            index += 1
        self.addChecker()
    else:
        preProcessedDocs = documents
        unique = self.getUniqueWords(preProcessedDocs, True)

    for listOfTokens in preProcessedDocs:
        tf.append(self.getTermFreq(
            unique, len(listOfTokens), listOfTokens))
        tv.append(self.getTerm(unique, len(listOfTokens), listOfTokens))
    tf.pop(0)
    tv.pop(0)
    idf = self.inverse(unique, preProcessedDocs, tv)
    final = self.calculateTFIDF(tf, idf, final)
    final.pop(0)
    values = []
    for doc in final:
        values.append(doc.values())
    self.test(values[0])

    if (testing == False):
        tf_idf = self.convertingToDP(final)
    return values
```

Figure 83 Apply TF-IDF to the Newly Uploaded Document

The process of applying TF-IDF is a similar process to that when training the model, the only difference is there's a new document appended.

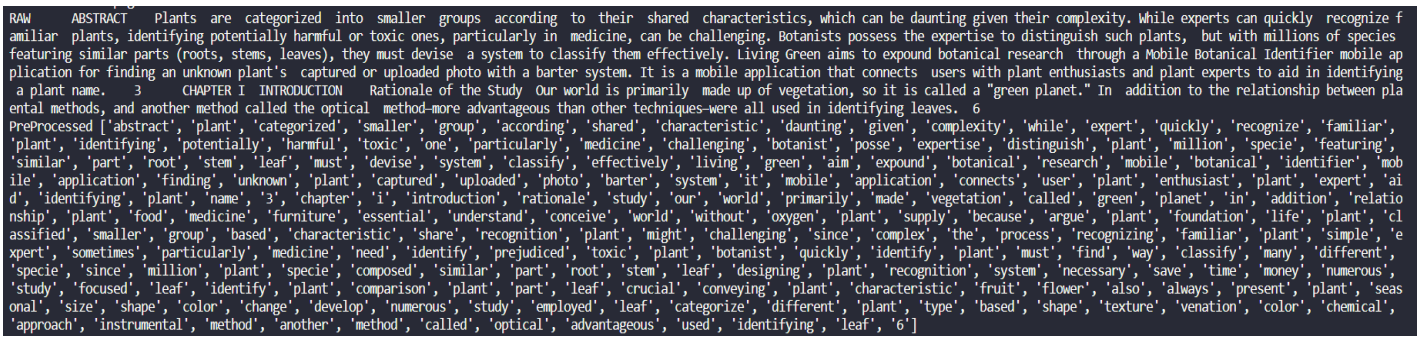


Figure 84 The Raw Data and Preprocessed Data of the New Document

This data represents the new document to be classified. The abstract and the Introduction are extracted and preprocessed.

overview	0
a	0
world	2
free	0
from	0
...	...
flower	1
color	2
categorize	1
venation	1
optical	1

Figure 86 Term Vectors of New Document

	Value
overview	0.556303
a	-1.054358
world	-1.139179
free	0.109144
from	-0.023481
...	...
flower	1.255273
color	0.954243
categorize	1.255273
venation	1.255273
optical	1.255273

Figure 85 Term Frequency of New Document

new	Value
overview	0.000000
a	0.000000
world	0.009174
free	0.000000
from	0.000000
...	...
flower	0.004587
color	0.009174
categorize	0.004587
venation	0.004587
optical	0.004587

Figure 87 Inverse Frequency of New Document

[10393 rows x 1 columns]

	Value
overview	0.000000
a	-0.000000
world	-0.010451
free	0.000000
from	-0.000000
...	...
flower	0.005758
color	0.008755
categorize	0.005758
venation	0.005758
optical	0.005758

Figure 88 TF-IDF of New Document

The process of applying TF-IDF on the newly uploaded document. Words below the three dots are all high values, this indicates that these words in the new document are unique with respect to the SDG corpus. After getting the TF-IDF of the new document

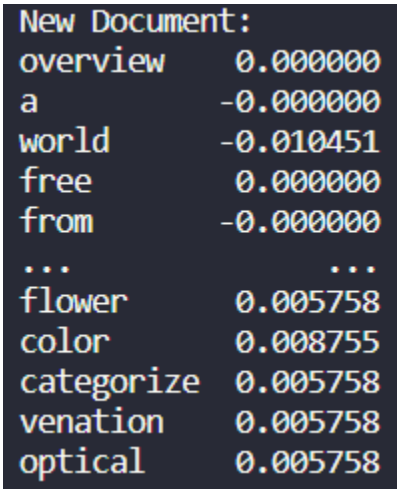
and appending it to the TF-IDF of the SDG corpus, the next step is cosine similarity scoring.

Cosine Similarity Scoring

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

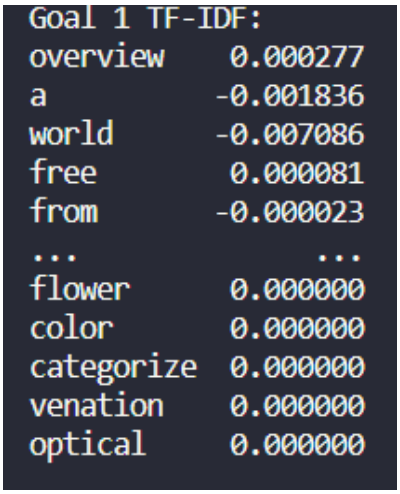
Figure 89 Cosine Similarity Formula

Mathematically, cosine similarity measures the cosine of the angle between two vectors in an inner product space or multi-dimensional space. In the context of this study, the vectors are arrays of TFIDF values of each document.



New Document:	
overview	0.000000
a	-0.000000
world	-0.010451
free	0.000000
from	-0.000000
...	...
flower	0.005758
color	0.008755
categorize	0.005758
venation	0.005758
optical	0.005758

Figure 90 TFIDF values of New Document



Goal 1 TF-IDF:	
overview	0.000277
a	-0.001836
world	-0.007086
free	0.000081
from	-0.000023
...	...
flower	0.000000
color	0.000000
categorize	0.000000
venation	0.000000
optical	0.000000

Figure 91 TFIDF values of Goal 1: No Poverty

Since we already have our arrays of TF-IDF values. The first step in getting the cosine similarity score is to apply the formula where A represents the New Document, and B the document in the TF-IDF. We first need to calculate the dot product of New

Document (A), with the first document in TF-IDF (B). The dot product measures the alignment of two vectors. If the result is positive, this means that the two vectors are pointing in a similar direction, and the opposite is negative. A zero value means that the two vectors are perpendicular to each other or in document classification's case, no similarity at all.

Dot Product of New Document and First Document in TF-IDF:

*Dot Product = (0.000000 x 0.000277) + (- 0.00000 * -0.001836) + (- 0.010451 * -0.007086) until the last value of both the Arrays.*

The next step is getting the product of the magnitudes of both vectors compared. This is to normalize the value of the dot product so we can have values from -1 and 1, representing the similarity/dissimilarity between the vectors.

||A|| x ||B||, where:

New Document (A) = [0.00000, -0.00000, -0.010451]

First Document in TFIDF (B) = [0.000277, -0.001836, -0.007086]

A = sqrt (0.00000^2 + -0.00000^2 + -0.010451^2)

B= sqrt (0.000277^2 + -0.001836^2 + -0.007086^2)

*Magnitude = A * B*

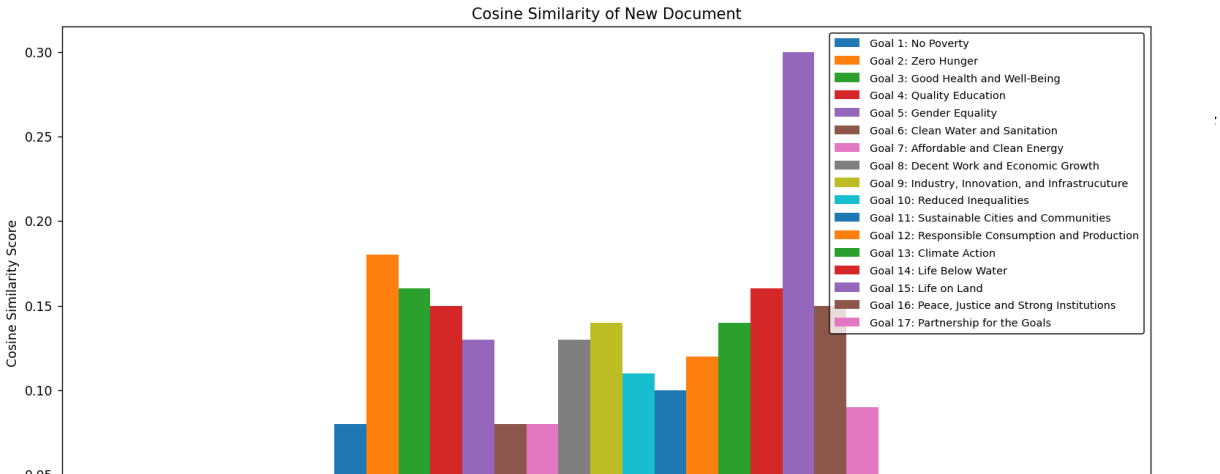
So, the result is Dot Product / Magnitude = Cosine Score

```
Dot Product:  0.0005002948157214309 Magnitude:  0.0031891493840120098
Dot Product:  0.0006354694623225191 Magnitude:  0.004224129279961806
Dot Product:  0.0006712936983007098 Magnitude:  0.005097953019234838
Dot Product:  0.0004649926236958813 5 Magnitude:  0.005640819885505017
Dot Product:  0.0005225613445220877 Magnitude:  0.006258342413813219
Dot Product:  0.0005404088936799382 Magnitude:  0.004154221245468342
Dot Product:  0.0004621314348072816 Magnitude:  0.0032975599194034295
Dot Product:  0.0003566790319039243 Magnitude:  0.00330276782326685
Dot Product:  0.0003773478244656774 7 Magnitude:  0.003663257124483432
Dot Product:  0.0005613008904062513 Magnitude:  0.004501854206457432
Dot Product:  0.0005504168288468866 Magnitude:  0.004048207131901482
Dot Product:  0.000580391169196893 Magnitude:  0.003524819031213985
Dot Product:  0.0010632756729797711 Magnitude:  0.0035150861364767897
```

Figure 92 Dot Product and Magnitude of New Document with respect to each Document

```
New Document x  Goal 1: No Poverty :  0.08088
New Document x  Goal 2: Zero Hunger :  0.18037
New Document x  Goal 3: Good Health and Well-Being :  0.15687
New Document x  Goal 4: Quality Education :  0.15044
New Document x  Goal 5: Gender Equality :  0.13168
New Document x  Goal 6: Clean Water and Sanitation :  0.08243
New Document x  Goal 7: Affordable and Clean Energy :  0.0835
New Document x  Goal 8: Decent Work and Economic Growth :  0.13009
New Document x  Goal 9: Industry, Innovation, and Infrastrucuture :  0.14014
New Document x  Goal 10: Reduced Inequalities :  0.10799
New Document x  Goal 11: Sustainable Cities and Communities :  0.10301
New Document x  Goal 12: Responsible Consumption and Production :  0.12468
New Document x  Goal 13: Climate Action :  0.13597
New Document x  Goal 14: Life Below Water :  0.16466
New Document x  Goal 15: Life on Land :  0.30249
New Document x  Goal 16: Peace, Justice and Strong Institutions :  0.14512
New Document x  Goal 17: Partnership for the Goals :  0.08801
```

Figure 93 Final Result of Cosine Similarity Scoring



words in the new document that exists in all the documents, the question is how relevant that word is to the entire SDG corpus, and that was the purpose of getting the TF-IDF. We can see that Goal 15: Life on Land outputs the highest score, 0.30249 or 30%. This is an accurate result since the new document we tested talks about plants and other species of them.

```
def getCosine(self, oldDoc, count):
    newVector, cosine = oldDoc[len(oldDoc)-1], []
    counter = 0
    classifier = {}
    del oldDoc[-1]
    goals = ["Goal 1: No Poverty", "Goal 2: Zero Hunger", "Goal 3: Good Health and Well-Being", "Goal 4: Quality Education"
             "Goal 9: Industry, Innovation, and Infrastructure", "Goal 10: Reduced Inequalities", "Goal 11: Sustainable Cit
             ]
    for val in oldDoc:
        val2 = val
        vector1, vector2 = [], []
        dotProduct, magnitude, magnitude1, magnitude2 = 0, 0, 0, 0
        for newvec in newVector:
            vector1.append(newvec)
        for oldVar in val2:
            vector2.append(oldVar)
        dotProduct = np.dot(vector1, vector2)
        magnitude1 = math.sqrt(
            sum(component ** 2 for component in newVector))
        magnitude2 = math.sqrt(sum(component ** 2 for component in val2))
        magnitude = magnitude1 * magnitude2
        cosine.append(round(dotProduct/magnitude, 5))
        percent = round(
            (dotProduct / magnitude) * 100, 2)
        classifier[goals[counter]] = percent
        counter += 1

    sorted_dict = dict(
        sorted(classifier.items(), key=lambda item: item[1], reverse=True))
    return sorted_dict
```

Figure 95 Code Snippet for Getting the Cosine Scores

Here we can see how getting the cosine scores is implemented. First, we have a variable oldDoc that is a list of values we got from our TF-IDF, under the hood, it contains 17 items. The variable newVector holds the TF-IDF values of the new document. Once we all have 2 variables, we then calculate the dot product of newVector to the first index of the oldDoc, once we get the dot product, we then proceed to get the magnitude. The variable magnitude1 stores the value of the newVector and the magnitude2 variable stores the value of the first element of oldDoc. After that, we simply multiply both and store it to variable magnitude. Finally, we then divide dotProduct and magnitude to get the cosine score of New Document and Goal 1. This process is repeated until the loop iterates over the entire oldDoc list.

At the end of the loop, the values are then stored in the dictionary which has the goal name as the key, and the cosine score as the value.


```
{
  "Goal 10: Reduced Inequalities": 17.32,
  "Goal 11: Sustainable Cities and Communities": 23.15,
  "Goal 12: Responsible Consumption and Production": 26.86,
  "Goal 13: Climate Action": 22.48,
  "Goal 14: Life Below Water": 25.71,
  "Goal 15: Life on Land": 26.21,
  "Goal 16: Peace, Justice and Strong Institutions": 32.32,
  "Goal 17: Partnership for the Goals": 22.26,
  "Goal 1: No Poverty": 20.25,
  "Goal 2: Zero Hunger": 22.78,
  "Goal 3: Good Health and Well-Being": 35.13,
  "Goal 4: Quality Education": 28.53,
  "Goal 5: Gender Equality": 25.57,
  "Goal 6: Clean Water and Sanitation": 14.38,
  "Goal 7: Affordable and Clean Energy": 13.79,
  "Goal 8: Decent Work and Economic Growth": 34.21,
```

Figure 98 shows all goals with their respective cosine similarity scores with the new document uploaded by the user. As evident by the results in the image, there are no goals that have a zero value, this is because, there are numerous words that can be found in the new document and in all the training data for each goal.

Since we need to have the top four goals, we're going to cut the result and only extract the top four goals in the entire list based on their cosine scores. So, the result would look like this,

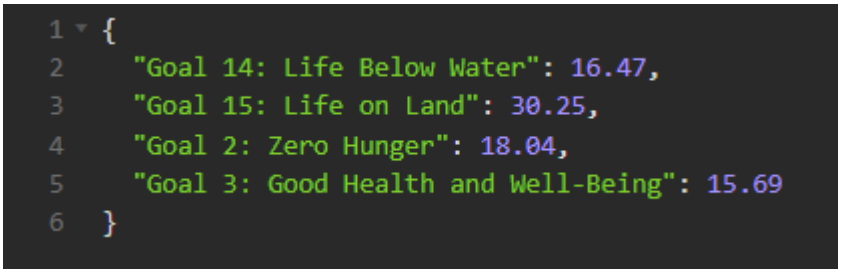


Figure 97: Final Classification Result

TFIDF Only

Another way to classify SDG papers is to use the TFIDF matrix. This approach involves a rule-based matchmaking with pre-defined set of rules for each goal.

The first step in this process is to create the rules needed for determining the classification label of the new document. To achieve this, the study determines what words are usually included in each goal.

```
rules = {}
rules["Goal 1"] = ['poverty', 'poor', 'inequality',
                   'income', 'disparity', 'reduction']
rules["Goal 2"] = ['hunger', 'malnutrition',
                   'famine', 'food', 'waste', 'farming']
rules["Goal 3"] = ['health', 'covid', 'mental',
                   'vaccination', 'disease', 'pandemic', 'healthcare']
rules["Goal 4"] = ['education', 'literacy',
                   'quality', 'learning', 'school', 'development']
rules["Goal 5"] = ['gender', 'equality', 'empowerment',
                   'women', 'rights', 'equal', 'protection', 'lgbtq']
rules["Goal 6"] = ['water', 'clean', 'sanitation',
                   'conservation', 'hygience', 'scarcity']
rules["Goal 7"] = ['energy', 'renewable',
                   'fuel', 'fossil', 'reduction', 'clean']
rules["Goal 8"] = ['work', 'growth', 'economic',
                   'labor', 'rights', 'productivity', 'job']
rules["Goal 9"] = ['innovation', 'artificial', 'intelligence',
                   'AI', 'technology', 'infrastructure', 'industrialization']
rules["Goal 10"] = ['inequality', 'racism', 'racist',
                    'empowerment', 'marginalized', 'equity']
rules["Goal 11"] = ['cities', 'sustainable',
                    'urbanization', 'green', 'spaces', 'resilient']
rules["Goal 12"] = ['consumption', 'sustainable',
                    'resource', 'management', 'waste']
rules["Goal 13"] = ['climate', 'change', 'carbon', 'emissions', 'energy']
rules["Goal 14"] = ['marine', 'conservation', 'coral',
                    'reefs', 'biodiversity', 'ocean', 'pollution']
rules["Goal 15"] = ['land', 'life', 'animals', 'wildlife',
                    'degradation', 'ecosystem', 'deforestation', 'landslide', 'plant', 'specie']
rules["Goal 16"] = ['peace', 'justice',
                    'institution', 'corruption', 'law', 'rule', 'society']
rules["Goal 17"] = ['partnerships', 'collaboration',
                    'cooperation', 'capacity', 'goals', 'sustainable']
```

Figure x: Set of Pre-defined Rules.

Figure x shows how the system creates the pre-defined rules. It starts with the creation of dictionary data structure, then populate each goal's pre-defined rules by manually adding keywords that are usually found in each goal. Determining these keywords are done with the help of the internet and clearly, general knowledge.

Once the rules are set, the next step is to create the TFIDF of the SDG corpus alongside the TFIDF of the new document uploaded.

```
def getTFIDF(self, data):
    newDocs = []
    trainingDocs = cons.extractTraining()
    newDocs.append(data)
    newData = cons.preprocess_documents(newDocs)
    data = newData[0]
    trainingDocs.append(data)
    listOfDict = cons.TFIDFForConfusion(trainingDocs, False)
    count = 1
    newDoc = listOfDict[len(listOfDict)-1]
    print(newDoc)
    del listOfDict[-1]
    result = self.compare(newDoc)
    return result
```

Figure x: Getting the TFIDF

Getting the TFIDF is straightforward and is also like the one used by TFIDF with Cosine. After getting the TFIDF of the SDG corpus, the next step is to remove the last row in the TFIDF, since it represents the new document. Once the TFIDF of the new document is extracted, the features or the keywords in the new document will then be used to compare with the pre-defined rules for each goal.

```
def compare(self, dic):
    total_goal, temp = [], []
    total = 1
    testing, final, super_final_dict = {}, {}, {}
    values = list(dic.keys())
    rules_values = list(rules.values())
    for rules1 in rules_values:
        for rules2 in rules1:
            for val in values:
                if (rules2 == val):
                    testing[dic[val]] = total
                    temp.append(dic[val])
        total += 1
        total_goal.append(temp)
```

Figure x: Comparison for TFIDF and Pre-defined Rules.

In this method, each keyword in the pre-defined rules is taken to match a word in the TFIDF of the new document. Once a keyword is found, the TFIDF score of that keyword is added to a list to be summed once the comparison function is done executing.

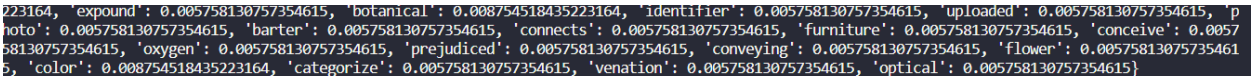


Figure x: TFIDF for New Document

Figure x shows the TFIDF of the new document with the values. This is only a snippet of a TFIDF using a dictionary to store its values.

```
for key, value in testing.items():
    str_value = str(value)
    if value != -0.0 and str_value != '-0.0':
        if value not in final:
            final[value] = key
        else:
            final[value] += key
```

Figure x: Filter Non-Zero Goals

After the comparison function stops executing, a filtering function will be called to remove those goals that have zero values in all its keywords. Those goals that remains are those that has one of its keywords matches in the TFIDF of the new document and with non-zero TFIDF score. These scores will then be added to determine which of the filtered goals the new document should be classified.

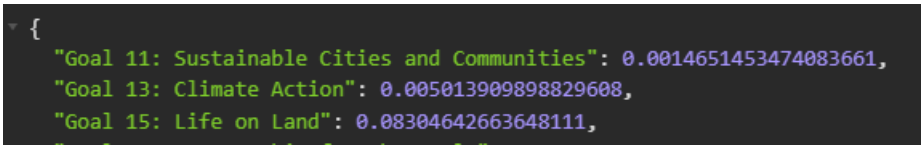


Figure x: Final Classification for TFIDF Only

Figure x shows the final classification result for TFIDF only and it only displays those goals that has non-zero TFIDF scores. In this case, the newly uploaded document is classified as Goal 15: Life on Land, which perfectly fits since the uploaded document talks about plants.

K-nearest Neighbor Algorithm

```
knn_model = KNeighborsClassifier(n_neighbors=n)
knn_model.fit(tfidf_matrix, target_labels)
preprocessed_new_document = self.preprocess_text(data)
new_tfidf = vectorizer.transform([preprocessed_new_document])
predicted_label = knn_model.predict(new_tfidf)
```

Figure x: KNN Implementation

The KNN algorithm also uses TFIDF as its input to data to classify SDG papers. This implementation uses different sets of parameters to classify papers, namely: 1,2,3,4,5,6, and 10 values. These values are called k-values. This k represents how

neighbors are eligible to vote once a paper is uploaded. The first step in this algorithm is to train the model using the fit function using the TFIDF of the entire corpus, then apply preprocess to the new document then call the predict function. The default distance metric for the KNN is the Euclidean Distance, this is very different from the ones used by TFIDF with Cosine since the latter uses a scoring metric instead of a distance metric. The model predicts the label by displaying the name of the goal in which the paper belongs.

TESTING PROCESS

This section contains the result of the testing process. The system has two testing processes namely, accuracy and performance testing. The former is used to test the validity of the results, and the latter determines the time it takes for the system to execute a feature from start to finish. The evaluation metric used for testing the performance of the SDG classifier is the accuracy testing since it is the most intuitive performance measure.

ACCURACY TESTING

Accuracy testing is used to validate the classified labels of the SDG classifier. Since the classifier outputs four top goals of the paper, the accuracy metric

There are 25 approved research papers in the test set. Each paper is labeled manually with its top four goals. An accuracy test was performed. Each paper in the data set is classified and the actual results are compared to the expected results.

Confusion Matrix

A confusion matrix is used to visualize and determine the performance of the SDG classifier using three different approaches, namely: TFIDF and Cosine Similarity, TFIDF only, and TFIDF, Cosine Similarity, and KNN with different hyperparameters or value of K.

Accuracy, Precision, Recall, and F1 score are the different performance metrics used to provide additional valuable information about the performance of a classification model.

Accuracy is the overall measure of correct predictions. It provides an overview of how well the model performs.

Precision measures the proportion of correctly predicted positive instances (TP) out of all instances predicted as positive (TP and FP). It focuses on the quality of positive predictions, indicating how reliable the model is when it predicts a positive outcome.

Recall measures the proportion of correctly predicted positive instances (TP) out of all actual positive instances (TP and FN). It focuses on capturing all positive instances and is particularly useful when the cost of false negatives (missing positive instances) is high.

The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, considering both precision and recall. The F1 score is useful when there is an imbalance between the positive and negative classes in the dataset.

Confusion Matrix (TFIDF and Cosine Similarity)

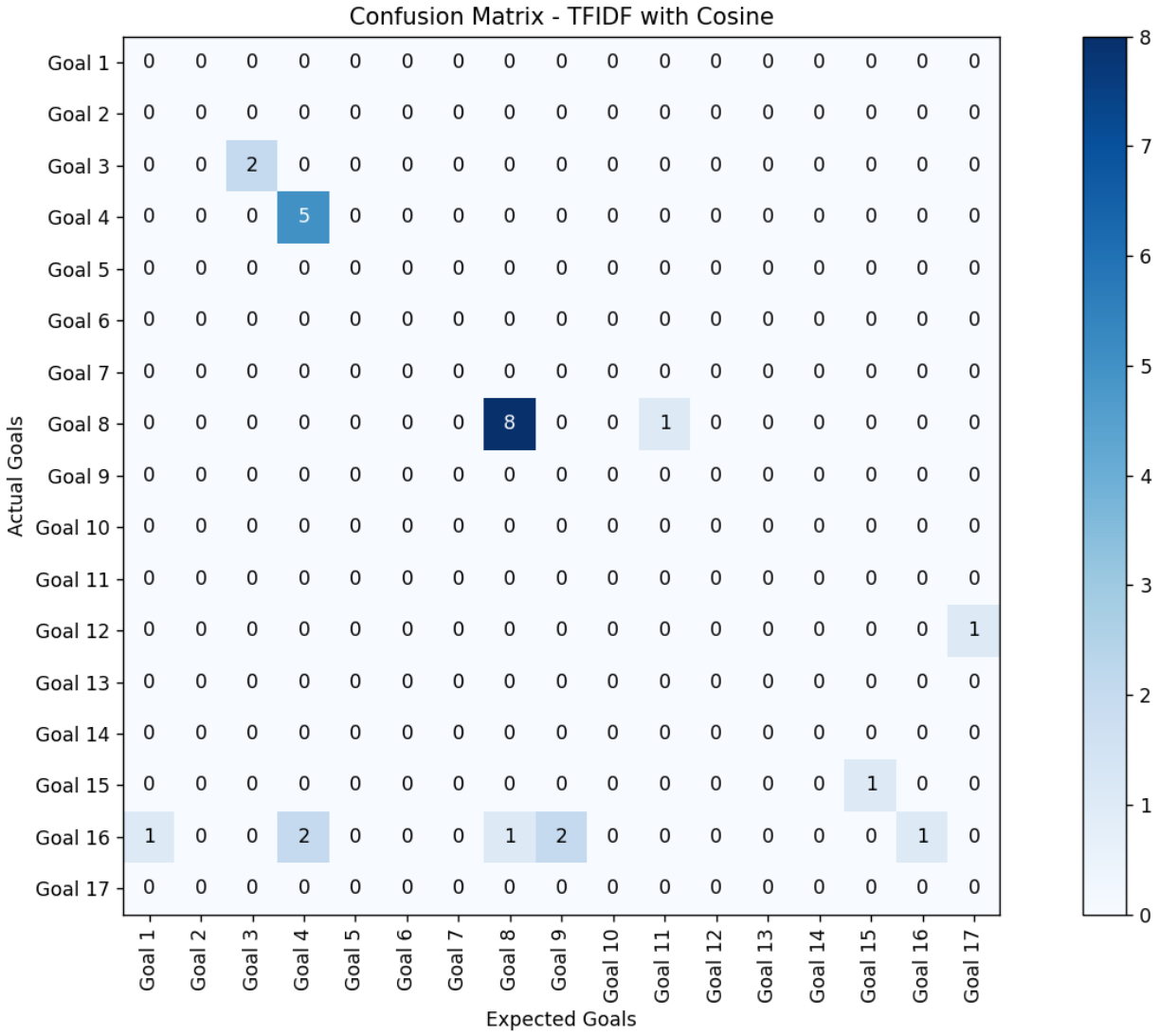


Figure 100 TFIDF and Cosine Similarity Confusion Matrix

Figure 100 shows the confusion matrix for TFIDF and Cosine Similarity with a total of 25 test sets. The column represents the expected goals of the classifier, and the rows represent the actual classifier result of the data set. The highest value in the matrix is 8, and it is Goal 8: Decent Work and Economic Growth, this shows the top goal for 8 papers is Goal 8. To properly analyze the result of the matrix, there are four basic characteristics used namely: True Positive, True Negative, False Positive, and False Negative.

True Positive or TP, shows the number of correctly classified papers in the data set. Since the classifier outputs the top four goals, only the goal which has the highest percentage is the actual value.

True Negative or TN, is the number of correctly identified papers that is not the top goal.

False Positive or FP, represents the number of misclassified papers that is not the top goal in the expected goals.

False Negative or FN, is an instance that is incorrectly classified as belonging to the negative class. This would be an instance where the SDG classifier does not identify the top 1 goal at all.

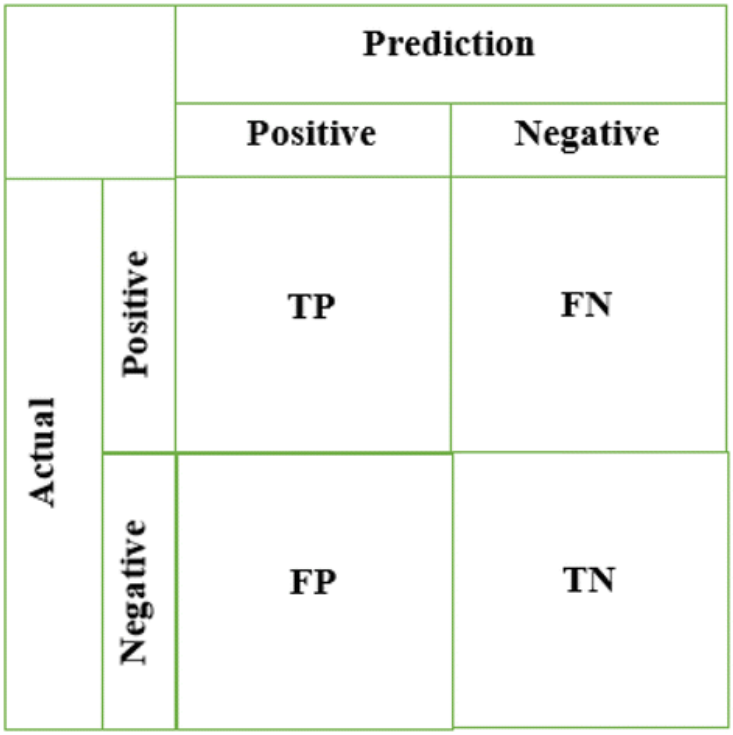


Figure 101 Confusion Matrix for Binary Classification

Figure 101 shows the confusion matrix for a binary classification. Since the study aims to classify research papers into 17 SDGs, it is called a multi-class classification. Getting the TP, TN, FP, and FN in a multi-class classification is different from a binary classification. To calculate it in a multi-class classification, we should calculate each process for each class. Goal 1 has its own TP, TN, FP, and FN, as well as its own Precision, Recall, and F1 score. This is called One-vs-Rest (OvR) approach.

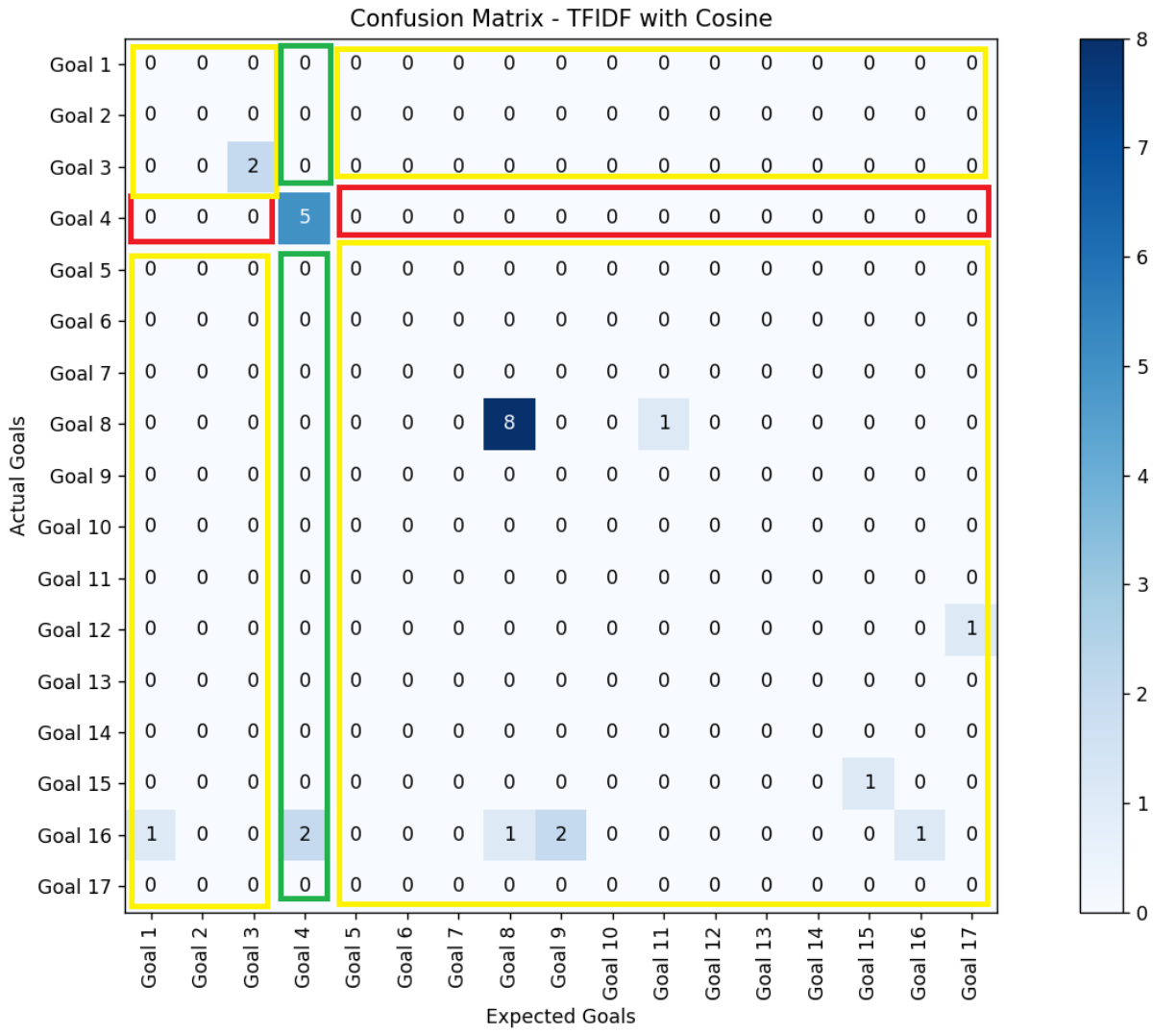
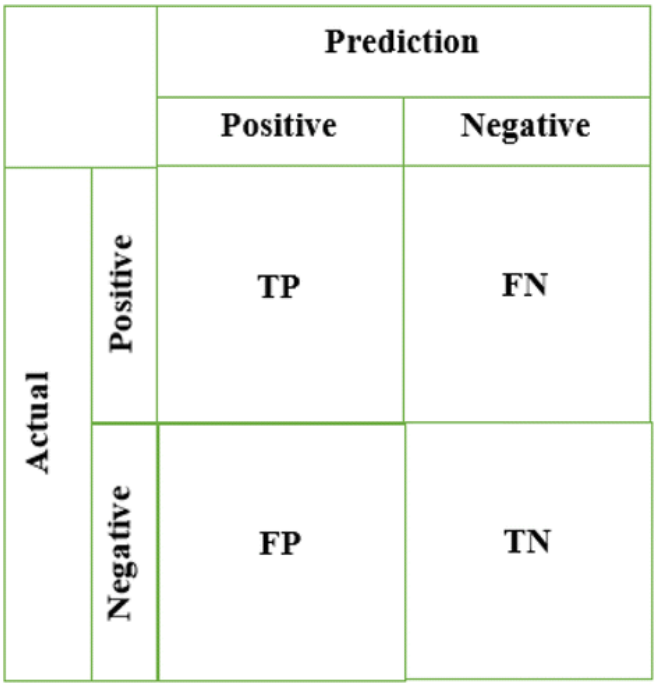


Figure 102 Sample for Multi-class Classification

Figure 102 visualizes how we can calculate the different metrics for each class.

Here we take Goal 4 as an example, by following the logic of binary classification we can get our desired values.



From Figure 101: Confusion Matrix for Binary Classification

We can pattern our logic by following the image above. In goal 4, all values from that row, colored in red, except the TP value or the diagonal value are added, the sum is 0 and that is our False Negative. To get the False Positive, we sum all values in Goal 4 column, colored in green, except for the diagonal value, we get 2. Finally, to get the True Negative, we sum all rows and columns except the values of the class we are calculating, colored in yellow, we get 18. We do this for each class, and we have our own values for TP, TN, FP, and FN for each goal in our matrix.

	TP	FP	TN	FN
Goal 1	0	1	24	0
Goal 2	0	0	25	0
Goal 3	2	0	23	0
Goal 4	5	2	18	0
Goal 5	0	0	25	0
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	8	1	15	1
Goal 9	0	2	23	0
Goal 10	0	0	25	0
Goal 11	0	1	24	0
Goal 12	0	0	24	1
Goal 13	0	0	25	0
Goal 14	0	0	25	0
Goal 15	1	0	24	0
Goal 16	1	0	18	6
Goal 17	0	1	24	0

Figure 103 Final Table for TP, FP, TN, and FN

Figure 103 shows the result of the calculation based on the steps mentioned above. The values of the four metrics for each goal will then be used to calculate the Precision, Recall, and F1 score for each goal. The total TP for TFIDF-Cosine is 17, FP is 8, TN is 398, and 8 for FN.

Recall metric represents the number of correctly classified positive papers out of the total number of actual positive papers.

$$Recall = \frac{TP}{TP + FN}$$

By using this formula and based on the values above we can get the recall for goal 4.

TP = 5, FN = 0, so 5 / 5 + 0 = 1.0 Recall for **Goal 4.**

Precision of an algorithm is represented as the ratio of correctly classified papers (TP) with that top 1 goal to the total papers predicted to have that top 1 goal. (TP+FP)

$$Precision = \frac{TP}{TP + FP}$$

5 / 5 + 2 = 0.71, Precision for Goal 4.

The F1 score states the equilibrium between the precision and the recall. F1 scores consider both false positives and false negatives.

$$F1Score = \frac{2 * precision * recall}{precision + recall}$$

2 * (0.71 * 1.0) / (0.71 +1.0) = 0.83, F1 score for Goal 4.

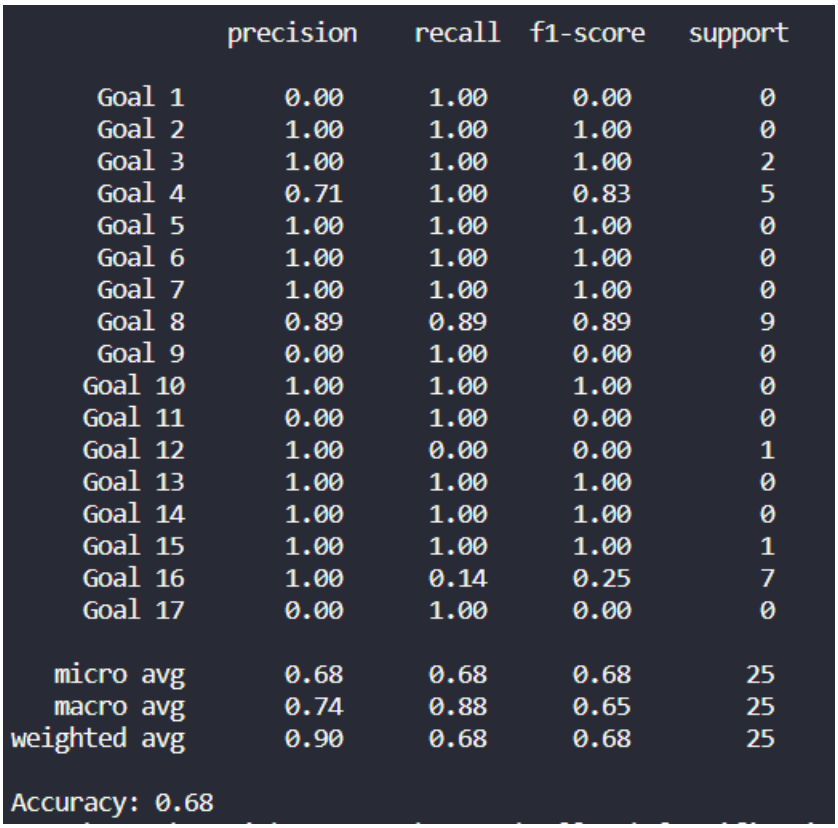
	Precision	Recall	F1 Score
Goal 1	0.0	0.0	0.0
Goal 2	0.0	0.0	0.0
Goal 3	1.0	1.0	1.0
Goal 4	0.7142857142857143	1.0	0.8333333333333333
Goal 5	0.0	0.0	0.0
Goal 6	0.0	0.0	0.0
Goal 7	0.0	0.0	0.0
Goal 8	0.8888888888888888	0.8888888888888888	0.8888888888888888
Goal 9	0.0	0.0	0.0
Goal 10	0.0	0.0	0.0
Goal 11	0.0	0.0	0.0
Goal 12	0.0	0.0	0.0
Goal 13	0.0	0.0	0.0
Goal 14	0.0	0.0	0.0
Goal 15	1.0	1.0	1.0
Goal 16	1.0	0.14285714285714285	0.25
Goal 17	0.0	0.0	0.0

Figure 104 Precision, Recall, and F1 Scores for each Goal

Figure 104 shows the result for each goal in calculating the Precision, Recall and F1 Scores.

These values alone are not enough to determine the performance of the model. An imbalance of the data set is very evident by observing the two figures (104 and 103), this situation is present in all the models as it is caused by the limited amount of data set. Imbalance refers to the uneven values of one class compared to another. Goal 1 has 0 values across all three metrics, and it also doesn't have any paper classified to it. This gives an inaccurate result when calculating the accuracy.

To solve this problem, we are implementing the micro, macro, and weighted averages for each metric.



	precision	recall	f1-score	support
Goal 1	0.00	1.00	0.00	0
Goal 2	1.00	1.00	1.00	0
Goal 3	1.00	1.00	1.00	2
Goal 4	0.71	1.00	0.83	5
Goal 5	1.00	1.00	1.00	0
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.89	0.89	0.89	9
Goal 9	0.00	1.00	0.00	0
Goal 10	1.00	1.00	1.00	0
Goal 11	0.00	1.00	0.00	0
Goal 12	1.00	0.00	0.00	1
Goal 13	1.00	1.00	1.00	0
Goal 14	1.00	1.00	1.00	0
Goal 15	1.00	1.00	1.00	1
Goal 16	1.00	0.14	0.25	7
Goal 17	0.00	1.00	0.00	0
micro avg	0.68	0.68	0.68	25
macro avg	0.74	0.88	0.65	25
weighted avg	0.90	0.68	0.68	25

Accuracy: 0.68

Figure 105 Final Classification Report for TFIDF + Cosine Similarity Score

Using sklearn's classification report package, we can generate a final tabulated information of all metrics. The first column shows the precision of each class, the second column is the recall, and f1-score on the third column. The support column indicates how many samples are in that goal. Goal 8 has 9 supports meaning there are 9 papers that are classified with Goal 8 as its top goal.

In the bottom of the report there are rows for micro, macro, and weighted averages. These three metrics provide different perspectives on how well a classifier performs across multiple classes.

Macro average is computed using the unweighted mean or the arithmetic mean of all the metric for each. This means that all classes or goals are treated equally regardless of their support values. In figure 105, let's take the macro average of the precision by adding all in the precision column and dividing it by 17. The result would be: 12.6 / 17 = 0.74 which matches the macro-averaged precision score in the classification report.

Weighted Average is calculated by getting the mean of all scores of a metric in each goal while considering each goal's support. Support refers to how many samples are in that goal. The weight here defines as the proportion of each class's support relative to the total of all support values. Using figure 105, we calculate the weighted average precision of the model: (1 x 2) + (0.71 x 5) + (0.89 x 9) + (1 x 1) + (1 x 1) + (1 x 7) / 25 = 0.9024, which correctly tallies on the report.

Micro average computes the global average of a metric by getting the sums of the TP, FN, and FP. In this case, our global TP is 17, global FN is 8, global FP is also 8. To

get the micro average for precision, TP / (TP + FP). 17 / (17 + 8) = 0.68, which is also shown in the report.

The accuracy in this case is computed the same way with the micro average, the proportion of the correctly classified papers out of all papers, thus the accuracy is 0.68 or 68%.

Confusion Matrix (TFIDF Only)

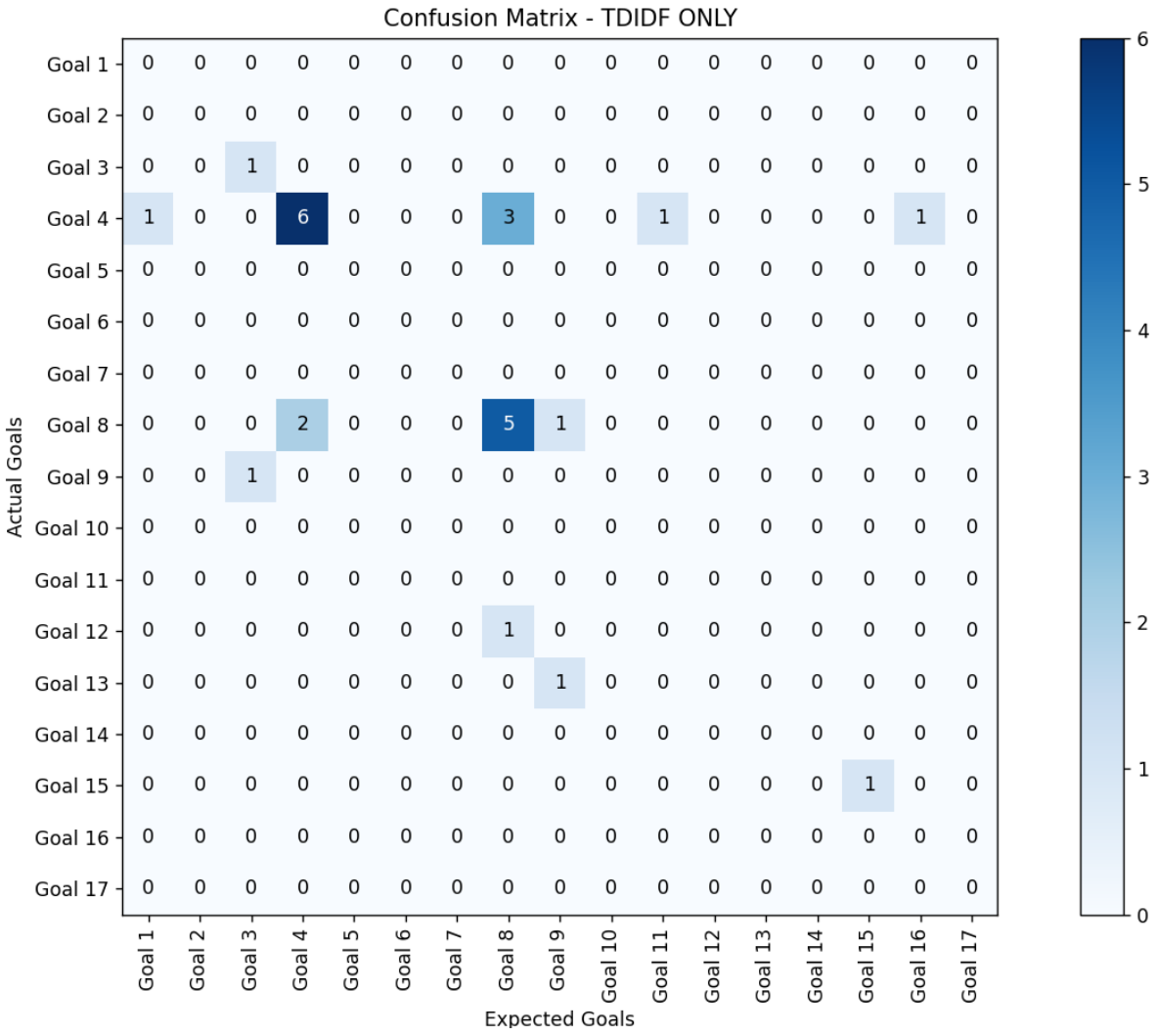


Figure 108 TFIDF Only Confusion Matrix

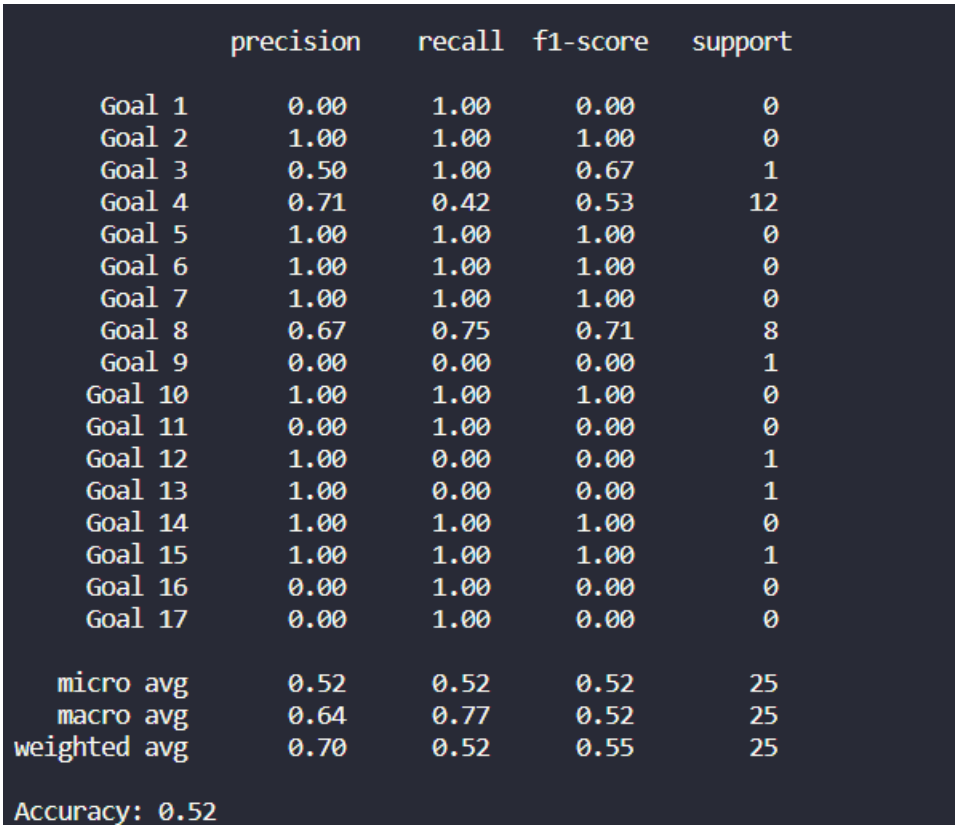
In this method, TFIDF is the only technique used to classify papers. As mentioned, this technique incorporates a rule-based approach instead of using a similarity scoring. In figure 108, it shows the result of a confusion matrix of a TFIDF only model. Also with 25 data sets, results are different compared to which with the Cosine Similarity. Goal 8 now only has a value of 5, Goal 4, a value of 6, other results are scattered unlike the other matrix.

Although the technique employed in classification is different, how we get the Accuracy, Precision, F1 score, and the recall are similar.

	TP	FP	TN	FN
Goal 1	0	1	24	0
Goal 2	0	0	25	0
Goal 3	1	1	23	0
Goal 4	5	2	11	7
Goal 5	0	0	25	0
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	6	3	14	2
Goal 9	0	2	22	1
Goal 10	0	0	25	0
Goal 11	0	1	24	0
Goal 12	0	0	24	1
Goal 13	0	0	24	1
Goal 14	0	0	25	0
Goal 15	1	0	24	0
Goal 16	0	1	24	0
Goal 17	0	1	24	0

Figure 109 TFIDF Only Final Table for TP, FP, TN, and FN

Calculating the TP, FP, TN, and FN in TFIDF only is also the same as how we calculate it with Cosine Similarity. Figure 109 shows the final values when getting the four metrics using the same formula used in Cosine Similarity.



	precision	recall	f1-score	support
Goal 1	0.00	1.00	0.00	0
Goal 2	1.00	1.00	1.00	0
Goal 3	0.50	1.00	0.67	1
Goal 4	0.71	0.42	0.53	12
Goal 5	1.00	1.00	1.00	0
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.67	0.75	0.71	8
Goal 9	0.00	0.00	0.00	1
Goal 10	1.00	1.00	1.00	0
Goal 11	0.00	1.00	0.00	0
Goal 12	1.00	0.00	0.00	1
Goal 13	1.00	0.00	0.00	1
Goal 14	1.00	1.00	1.00	0
Goal 15	1.00	1.00	1.00	1
Goal 16	0.00	1.00	0.00	0
Goal 17	0.00	1.00	0.00	0
micro avg	0.52	0.52	0.52	25
macro avg	0.64	0.77	0.52	25
weighted avg	0.70	0.52	0.55	25

Accuracy: 0.52

Figure 110 Final Classification Report for TFIDF Only

Figure 110 details the result of each goal in terms of the Recall, Precision, and F1-scores for the TFIDF model. Based on figure 110, in TFIDF only, values are more screwed towards goal 4.

Based on the report for TFIDF Only model, the classification only yields 0.52 accuracy rate. Also present in the report are the supports, in this model, most classified papers turned out to be in Goal 4: Quality Education.

Confusion Matrix (TFIDF with KNN k = 1)

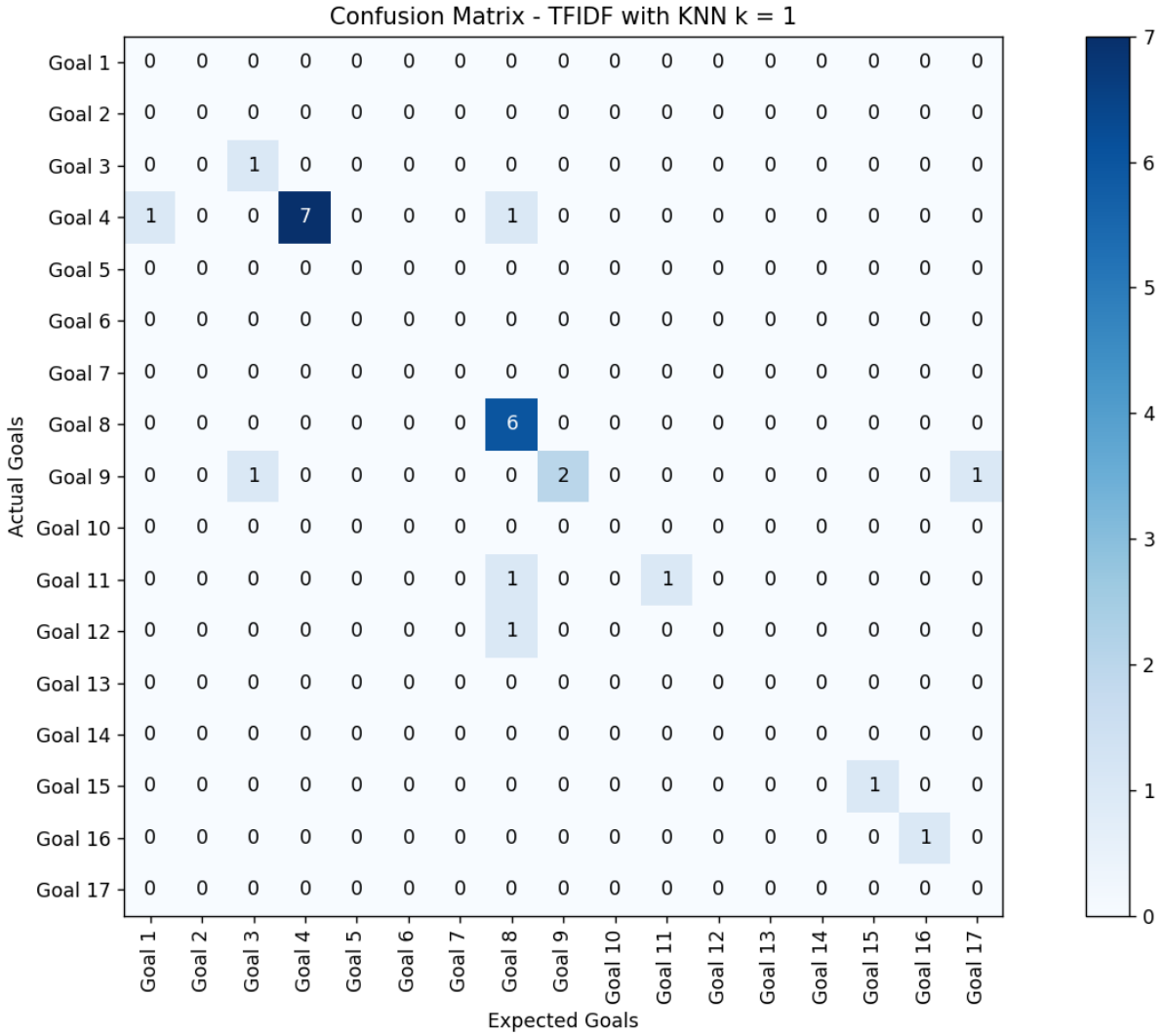


Figure 111 Confusion Matrix for KNN with k = 1

This model incorporates the KNN algorithm to classify papers with k = 1. The k in KNN refers to the number of neighbors that votes when a new document is classified. Figure 112 shows the confusion matrix of the model with 25 papers in the test set. Similarly, to the previous methods, we also need to evaluate the performance of this model using the same evaluation metric used in the earlier models.

	TP	FP	TN	FN
Goal 1	0	1	23	0
Goal 2	0	0	24	0
Goal 3	1	1	22	0
Goal 4	7	1	14	2
Goal 5	0	0	24	0
Goal 6	0	0	24	0
Goal 7	0	0	24	0
Goal 8	6	3	15	0
Goal 9	2	0	20	2
Goal 10	0	0	24	0
Goal 11	0	0	23	1
Goal 12	0	0	23	1
Goal 13	0	0	24	0
Goal 14	0	0	24	0
Goal 15	1	0	23	0
Goal 16	1	0	23	0
Goal 17	0	0	24	0

Figure 112 TFIDF with KNN k = 1 Final Table For TP, TN, FP, and FN

Figure 112 visualizes the final values for the 4 metrics, TP in goal 4 is seen here to be the highest. The four metrics here work the same with the other models.

	precision	recall	f1-score	support
Goal 1	0.00	1.00	0.00	0
Goal 2	1.00	1.00	1.00	0
Goal 3	0.50	1.00	0.67	1
Goal 4	1.00	0.78	0.88	9
Goal 5	1.00	1.00	1.00	0
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.67	1.00	0.80	6
Goal 9	1.00	0.50	0.67	4
Goal 10	1.00	1.00	1.00	0
Goal 11	1.00	0.50	0.67	2
Goal 12	1.00	0.00	0.00	1
Goal 13	1.00	1.00	1.00	0
Goal 14	1.00	1.00	1.00	0
Goal 15	1.00	1.00	1.00	1
Goal 16	1.00	1.00	1.00	1
Goal 17	0.00	1.00	0.00	0
micro avg	0.76	0.76	0.76	25
macro avg	0.83	0.87	0.75	25
weighted avg	0.90	0.76	0.77	25

Accuracy: 0.76

Figure 113 Final Classification Report for TFIDF with KNN k = 1

KNN with k = 1 shows a good result for goal 4 papers. This is shown clearly in the precision, recall, and F1 score of goal 4. This means that if the classified result is goal 4, then that result, based on the percentage of the three metrics, will be most likely true positive.

Final report shows that the model is performing well in classifying papers that are in either goal 4 or goal 8. The weighted average for precision is at .90 but its recall is .76 and the final f1-score is .77. This shows how important the role of the f1 score in giving balance to the recall and precision. The final accuracy for k = 1 is .76 which is already better than the previous models.

TFIDF-KNN with k = 2

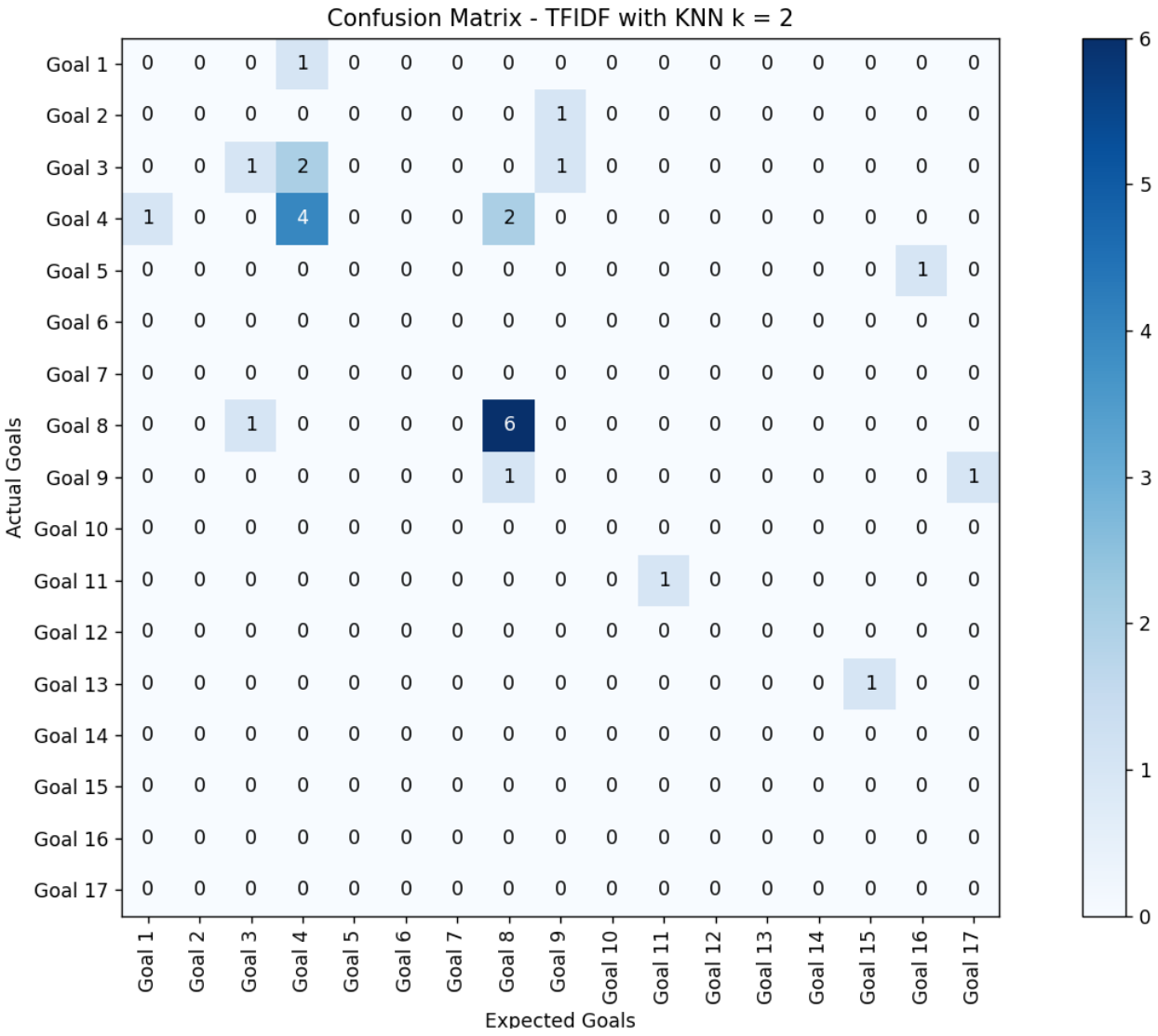


Figure 114 Confusion Matrix for TFIDF with KNN k = 2

Another parameter to test for KNN is k = 2, figure 116 shows the final confusion matrix of the parameter.

	TP	FP	TN	FN
Goal 1	0	0	24	1
Goal 2	0	0	23	2
Goal 3	1	1	20	3
Goal 4	5	4	14	2
Goal 5	0	0	24	1
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	6	3	15	1
Goal 9	0	2	21	2
Goal 10	0	0	25	0
Goal 11	0	1	24	0
Goal 12	0	0	25	0
Goal 13	0	0	24	1
Goal 14	0	0	25	0
Goal 15	0	1	24	0
Goal 16	0	1	24	0
Goal 17	0	0	25	0

Figure 115 Values for TFIDF with KNN k = 2

True Positive values for k = 2 are lower compared to the previous parameter. Goals 8 and 4 still hold the greatest number of correctly classified papers.

	precision	recall	f1-score	support
Goal 1	0.00	0.00	0.00	1
Goal 2	1.00	0.00	0.00	1
Goal 3	0.50	0.25	0.33	4
Goal 4	0.57	0.57	0.57	7
Goal 5	1.00	0.00	0.00	1
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.67	0.86	0.75	7
Goal 9	0.00	0.00	0.00	2
Goal 10	1.00	1.00	1.00	0
Goal 11	1.00	1.00	1.00	1
Goal 12	1.00	1.00	1.00	0
Goal 13	1.00	0.00	0.00	1
Goal 14	1.00	1.00	1.00	0
Goal 15	0.00	1.00	0.00	0
Goal 16	0.00	1.00	0.00	0
Goal 17	0.00	1.00	0.00	0
micro avg	0.48	0.48	0.48	25
macro avg	0.63	0.63	0.45	25
weighted avg	0.59	0.48	0.46	25

Accuracy: 0.48

Figure 116 Final Classification Report for TFIDF with KNN k = 2

Final metric values for k = 2, figure 116 clearly shows, excluding the zero values, goal 3 has the highest negativity rate, meaning papers classified to this goal are in most cases false positives. Overall accuracy of k = 2, is 0.48 which clearly is lower than the k = 1.

TFIDF-KNN with k = 3

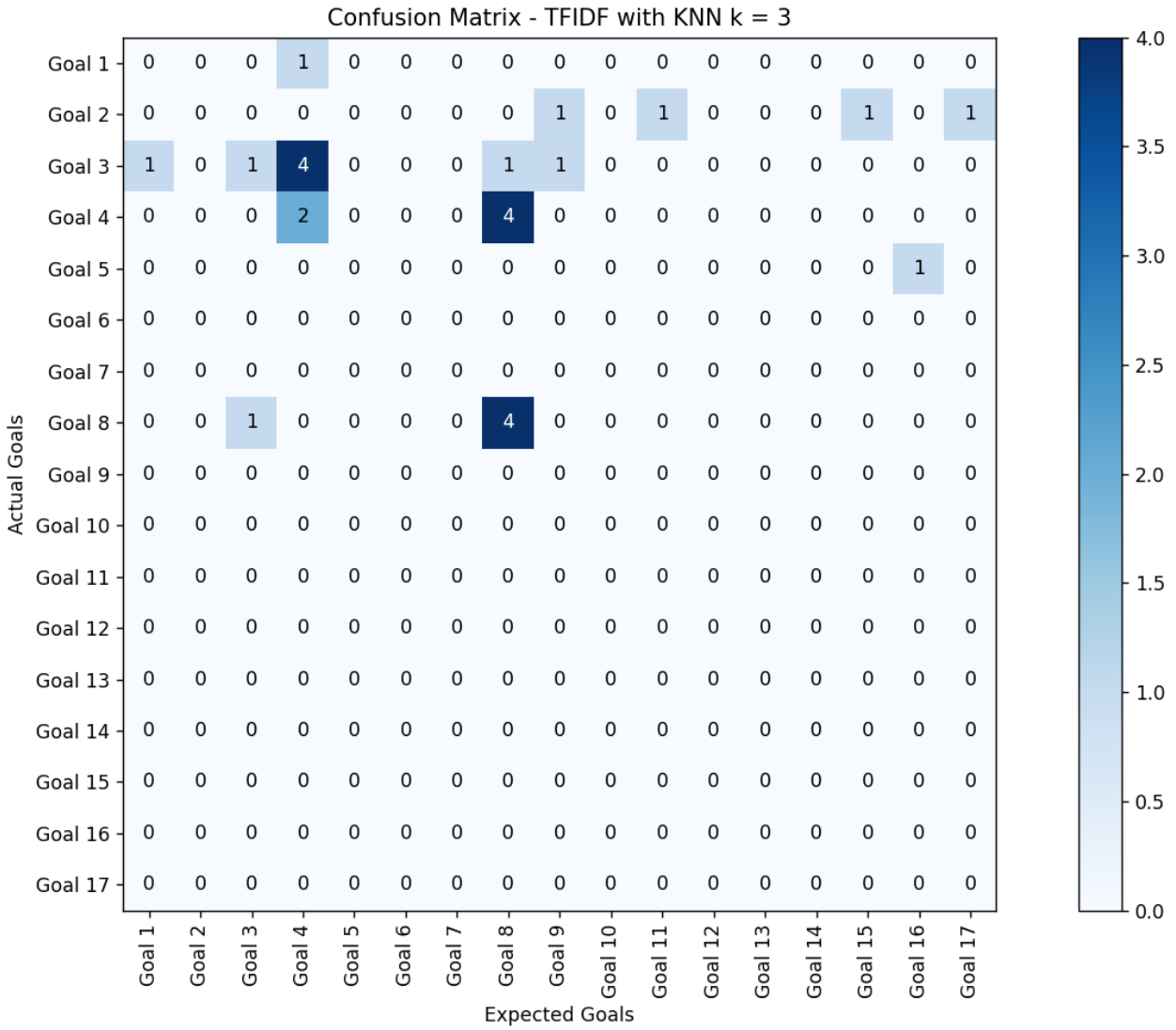


Figure 117 Confusion Matrix for TFIDF with KNN k = 3

Figure 117 shows how values are now distributed between goal 3 and goal 4. The values are getting closer between goal 1 and goal 4.

	TP	FP	TN	FN
Goal 1	0	2	22	1
Goal 2	0	0	21	4
Goal 3	2	1	16	6
Goal 4	3	3	15	4
Goal 5	0	0	21	4
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	0	8	16	1
Goal 9	0	2	23	0
Goal 10	0	0	25	0
Goal 11	0	2	23	0
Goal 12	0	0	25	0
Goal 13	0	0	25	0
Goal 14	0	0	25	0
Goal 15	0	1	24	0
Goal 16	0	1	24	0
Goal 17	0	0	25	0

Figure 118: Values TFIDF with KNN k = 3

As the confusion matrix shown in figure 117, goal 3 has the highest classified papers in all the goals and has the lesser true positive score of 2.

	precision	recall	f1-score	support
Goal 1	0.00	0.00	0.00	1
Goal 2	1.00	0.00	0.00	4
Goal 3	0.50	0.12	0.20	8
Goal 4	0.29	0.33	0.31	6
Goal 5	1.00	0.00	0.00	1
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.44	0.80	0.57	5
Goal 9	0.00	1.00	0.00	0
Goal 10	1.00	1.00	1.00	0
Goal 11	0.00	1.00	0.00	0
Goal 12	1.00	1.00	1.00	0
Goal 13	1.00	1.00	1.00	0
Goal 14	1.00	1.00	1.00	0
Goal 15	0.00	1.00	0.00	0
Goal 16	0.00	1.00	0.00	0
Goal 17	0.00	1.00	0.00	0
micro avg	0.28	0.28	0.28	25
macro avg	0.54	0.72	0.42	25
weighted avg	0.52	0.28	0.25	25

Accuracy: 0.28

Figure 119 Final Classification Report for TFIDF with KNN k = 3

KNN with k = 3 shows a decrease of accuracy, goals are seen to be classified into goal 3, goal 4, or goal 8. Although 4 goals are seen to have more samples, the model yielded 0.28 as its accuracy score which is not ideal for a classifier.

TFIDF-KNN with k = 4

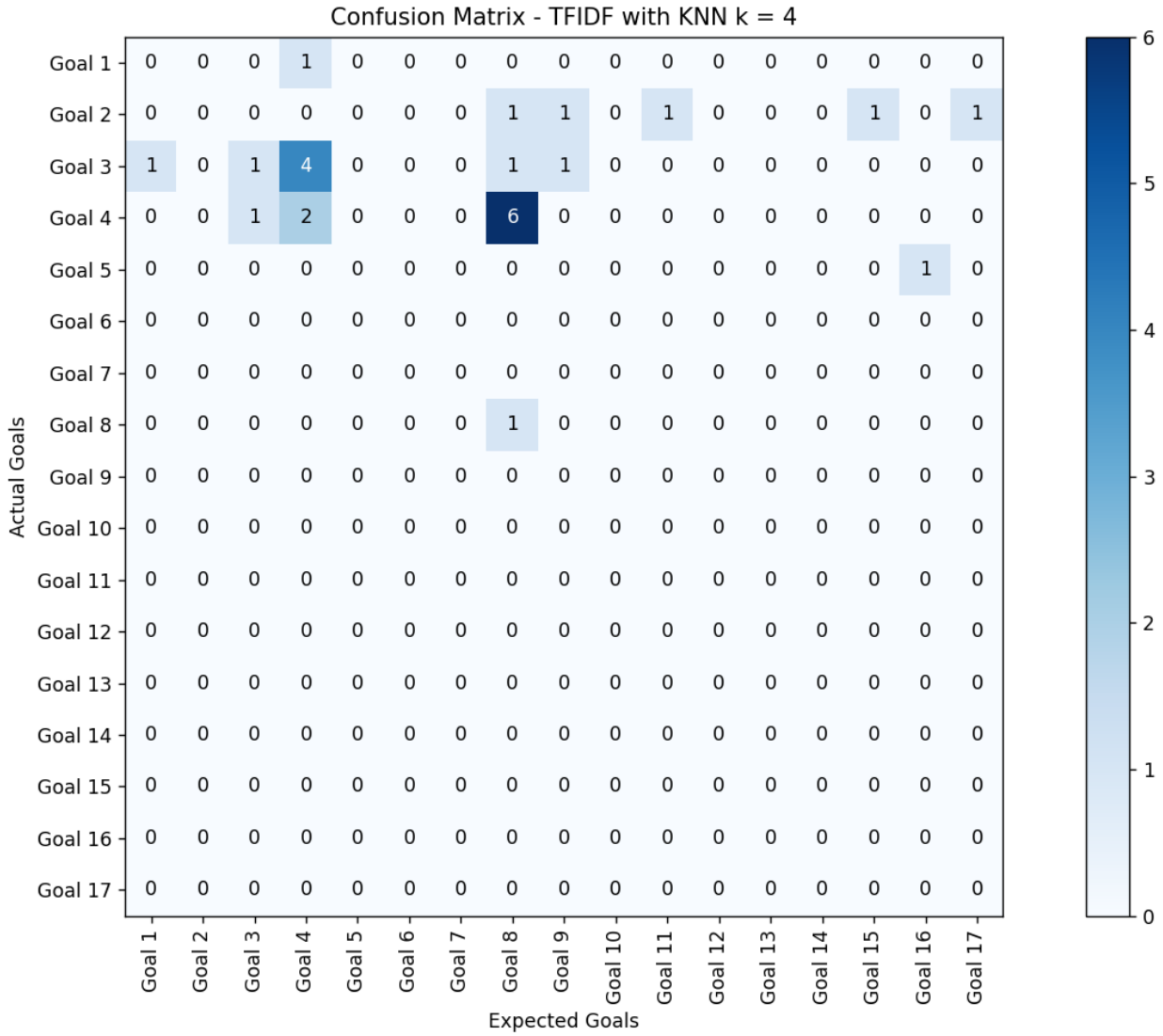


Figure 120 Confusion Matrix for TFIDF with KNN k = 4

Figure 120 visualizes the matrix for k = 4, as observed in the matrix, there are now less values in goal 8 which previously in all other models is consistent to have good sample size.

	TP	FP	TN	FN
Goal 1	0	0	21	4
Goal 2	0	0	21	4
Goal 3	1	1	13	10
Goal 4	2	7	14	2
Goal 5	0	0	24	1
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	1	7	17	0
Goal 9	0	2	23	0
Goal 10	0	0	25	0
Goal 11	0	2	23	0
Goal 12	0	0	25	0
Goal 13	0	0	25	0
Goal 14	0	0	25	0
Goal 15	0	1	24	0
Goal 16	0	1	24	0
Goal 17	0	0	25	0

Figure 121: Values TFIDF with KNN k = 4

There are now less true positive values in k = 4. More papers are labeled as False Negative or False Positive. This just means that there are now more papers that are wrongly classified.

	precision	recall	f1-score	support
Goal 1	0.00	0.00	0.00	1
Goal 2	1.00	0.00	0.00	5
Goal 3	0.50	0.12	0.20	8
Goal 4	0.29	0.22	0.25	9
Goal 5	1.00	0.00	0.00	1
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.11	1.00	0.20	1
Goal 9	0.00	1.00	0.00	0
Goal 10	1.00	1.00	1.00	0
Goal 11	0.00	1.00	0.00	0
Goal 12	1.00	1.00	1.00	0
Goal 13	1.00	1.00	1.00	0
Goal 14	1.00	1.00	1.00	0
Goal 15	0.00	1.00	0.00	0
Goal 16	0.00	1.00	0.00	0
Goal 17	0.00	1.00	0.00	0
micro avg	0.16	0.16	0.16	25
macro avg	0.52	0.73	0.39	25
weighted avg	0.51	0.16	0.16	25

Accuracy: 0.16

Figure 122 Final Classification Report for TFIDF with KNN k = 4

As evident in the previous figures, the accuracy of k = 4 is 0.16, which is significantly lower than its previous parameters. We are now seeing a trend here every time a value of k is incremented.

TFIDF-KNN with k = 5

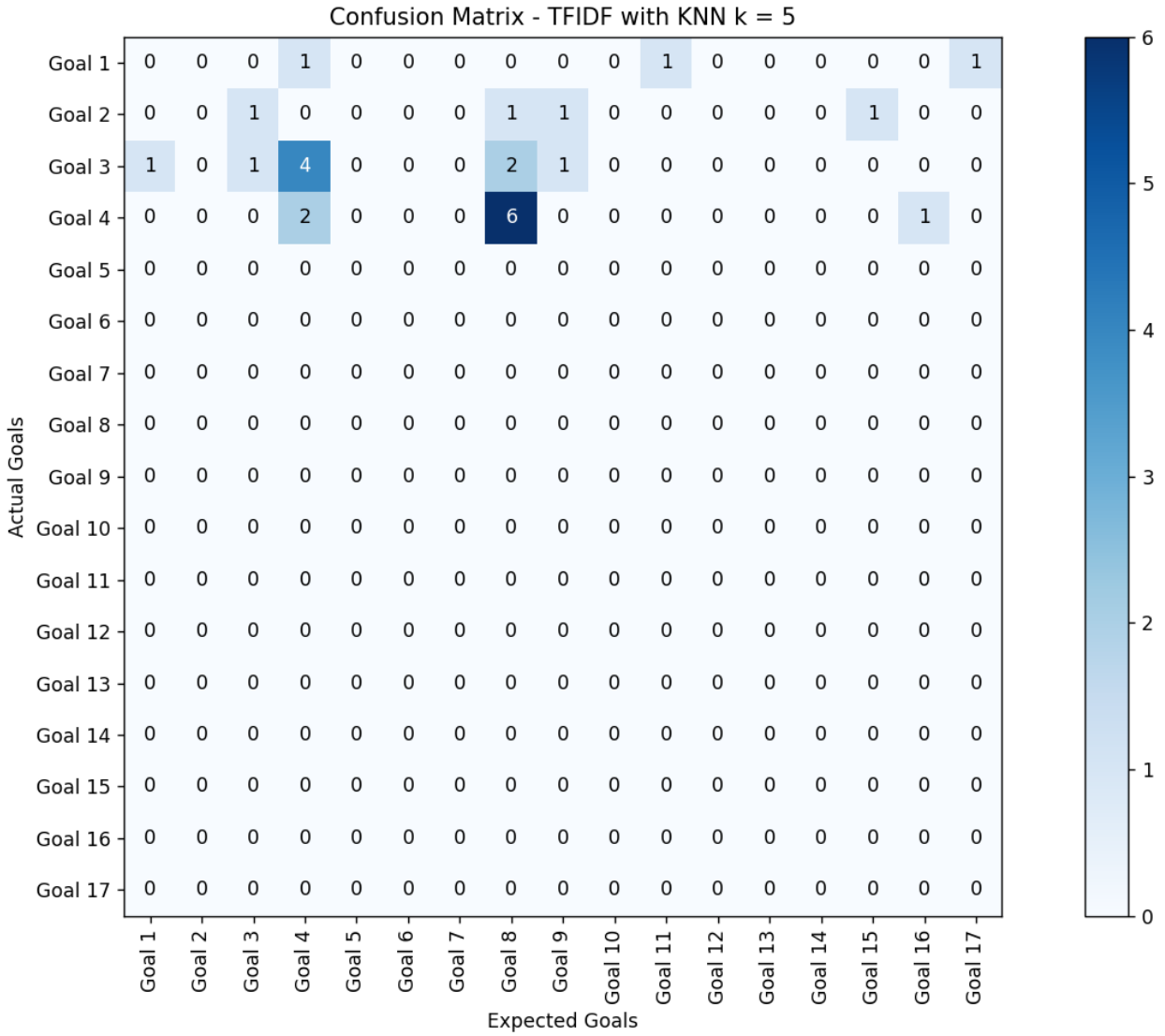


Figure 123 Confusion Matrix for TFIDF with KNN k = 5

Values in k = 5 are now seen to be only in between goal 1 to goal 4. Goal 4 still holds the highest number of papers classified to it.

	TP	FP	TN	FN
Goal 1	0	1	21	3
Goal 2	0	0	21	4
Goal 3	1	1	15	8
Goal 4	2	5	11	7
Goal 5	0	0	25	0
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	0	9	16	0
Goal 9	0	2	23	0
Goal 10	0	0	25	0
Goal 11	0	1	24	0
Goal 12	0	0	25	0
Goal 13	0	0	25	0
Goal 14	0	0	25	0
Goal 15	0	1	24	0
Goal 16	0	1	24	0
Goal 17	0	1	24	0

Figure 124: Values TFIDF with KNN k = 5

The true positive values for k = 5 are also less like k = 4. Both also show similarity in correctly classified papers which is in goal 3 and goal 4.

	precision	recall	f1-score	support
Goal 1	0.00	0.00	0.00	3
Goal 2	1.00	0.00	0.00	4
Goal 3	0.50	0.11	0.18	9
Goal 4	0.29	0.22	0.25	9
Goal 5	1.00	1.00	1.00	0
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.00	1.00	0.00	0
Goal 9	0.00	1.00	0.00	0
Goal 10	1.00	1.00	1.00	0
Goal 11	0.00	1.00	0.00	0
Goal 12	1.00	1.00	1.00	0
Goal 13	1.00	1.00	1.00	0
Goal 14	1.00	1.00	1.00	0
Goal 15	0.00	1.00	0.00	0
Goal 16	0.00	1.00	0.00	0
Goal 17	0.00	1.00	0.00	0
micro avg	0.12	0.12	0.12	25
macro avg	0.52	0.78	0.44	25
weighted avg	0.44	0.12	0.16	25

Accuracy: 0.12

Figure 125 Final Classification Report for TFIDF with KNN k = 5

K = 5 shows a decrease in accuracy of 0.12. This shows that k = 5 is not the right parameter value for this study.

TFIDF-KNN with k = 6

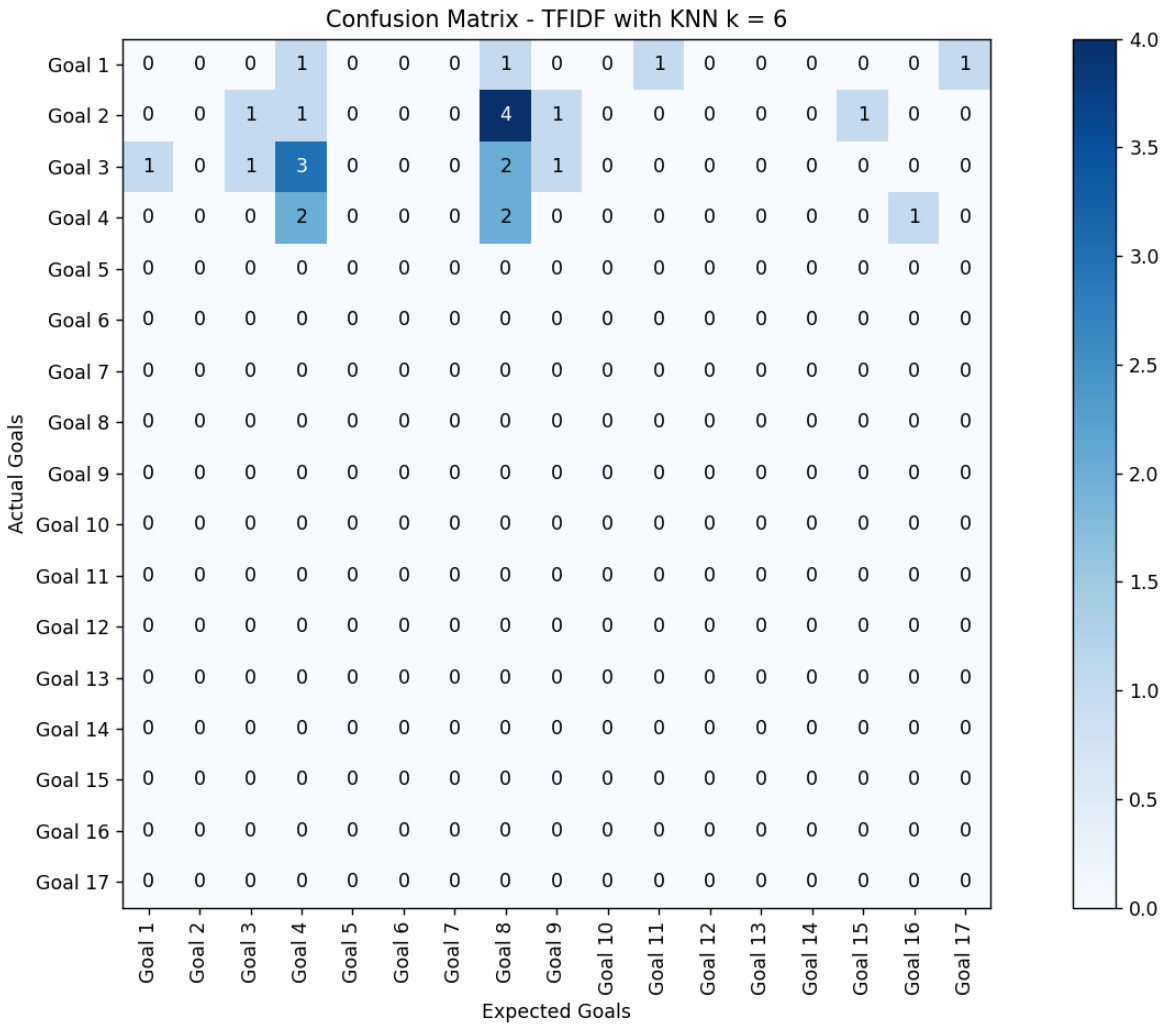


Figure 126 Confusion Matrix for TFIDF with KNN k = 6

Figure 126 shows the confusion matrix for KNN k = 6, values are scattered similarly with k = 5. This is expected as both values are closer to each other.

	TP	FP	TN	FN
Goal 1	0	1	20	4
Goal 2	0	0	17	8
Goal 3	1	1	16	7
Goal 4	2	5	15	3
Goal 5	0	0	25	0
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	0	9	16	0
Goal 9	0	2	23	0
Goal 10	0	0	25	0
Goal 11	0	1	24	0
Goal 12	0	0	25	0
Goal 13	0	0	25	0
Goal 14	0	0	25	0
Goal 15	0	1	24	0
Goal 16	0	1	24	0
Goal 17	0	1	24	0

Figure 127: Values for TFIDF with KNN k = 5

As with the confusion matrix, the value in the table above shows similarity with that of k = 5 table, particularly in the column of true positive and false positive.

	precision	recall	f1-score	support
Goal 1	0.00	0.00	0.00	4
Goal 2	1.00	0.00	0.00	8
Goal 3	0.50	0.12	0.20	8
Goal 4	0.29	0.40	0.33	5
Goal 5	1.00	1.00	1.00	0
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.00	1.00	0.00	0
Goal 9	0.00	1.00	0.00	0
Goal 10	1.00	1.00	1.00	0
Goal 11	0.00	1.00	0.00	0
Goal 12	1.00	1.00	1.00	0
Goal 13	1.00	1.00	1.00	0
Goal 14	1.00	1.00	1.00	0
Goal 15	0.00	1.00	0.00	0
Goal 16	0.00	1.00	0.00	0
Goal 17	0.00	1.00	0.00	0
micro avg	0.12	0.12	0.12	25
macro avg	0.52	0.80	0.44	25
weighted avg	0.54	0.12	0.13	25

Accuracy: 0.12

Figure 128: Final Classification Report for TFIDF with KNN k = 5

Final report for k = 6 shows similar accuracy score for k = 5, this clearly means that k = 5, and k = 6 doesn't show any form of improvement when it comes to classification accuracy.

TFIDF-KNN with k = 10

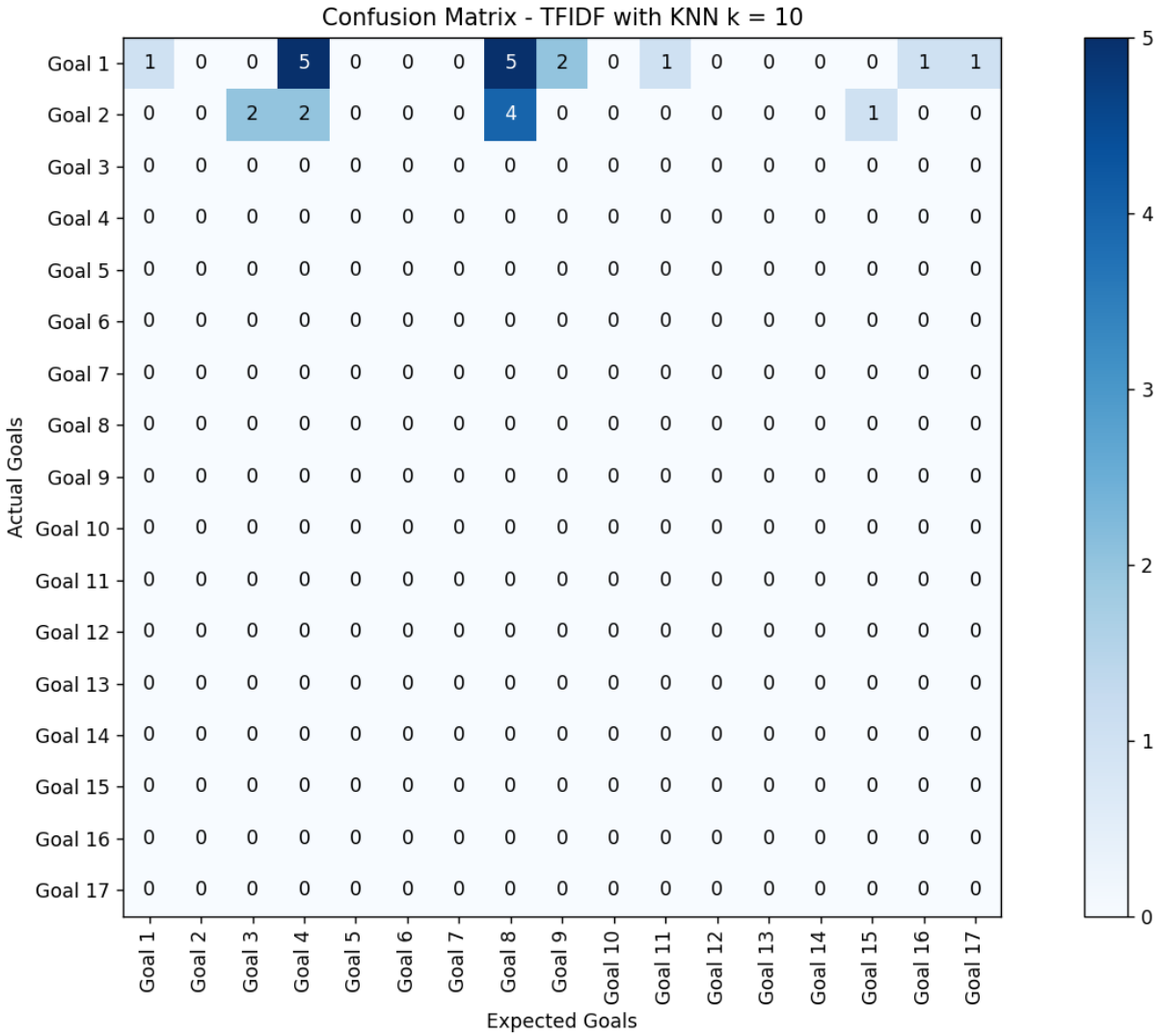


Figure 129 Confusion Matrix for TFIDF with KNN k = 10

A jump in parameter from k = 5 to k = 10 shows how values are now in between goal 1 and goal 2.

	TP	FP	TN	FN
Goal 1	1	0	9	15
Goal 2	0	0	16	9
Goal 3	0	2	23	0
Goal 4	0	7	18	0
Goal 5	0	0	25	0
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	0	9	16	0
Goal 9	0	2	23	0
Goal 10	0	0	25	0
Goal 11	0	1	24	0
Goal 12	0	0	25	0
Goal 13	0	0	25	0
Goal 14	0	0	25	0
Goal 15	0	1	24	0
Goal 16	0	1	24	0
Goal 17	0	1	24	0

Figure 130: Values for TFIDF with KNN k = 10

Figure 127 shows clearly that the model only got 1 paper correctly classified. This tells us that k = 10 should not be used as a parameter for KNN in this study.

	precision	recall	f1-score	support
Goal 1	1.00	0.06	0.12	16
Goal 2	1.00	0.00	0.00	9
Goal 3	0.00	1.00	0.00	0
Goal 4	0.00	1.00	0.00	0
Goal 5	1.00	1.00	1.00	0
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.00	1.00	0.00	0
Goal 9	0.00	1.00	0.00	0
Goal 10	1.00	1.00	1.00	0
Goal 11	0.00	1.00	0.00	0
Goal 12	1.00	1.00	1.00	0
Goal 13	1.00	1.00	1.00	0
Goal 14	1.00	1.00	1.00	0
Goal 15	0.00	1.00	0.00	0
Goal 16	0.00	1.00	0.00	0
Goal 17	0.00	1.00	0.00	0
micro avg	0.04	0.04	0.04	25
macro avg	0.53	0.89	0.42	25
weighted avg	1.00	0.04	0.08	25

Accuracy: 0.04

Figure 131: Final Classification Report for TFIDF with KNN k = 10

An accuracy of 0.04 is straight out bad. Values are more focused on goal 1 and goal 2 with very low f1-score and recall.

TFIDF-KNN with k = 17

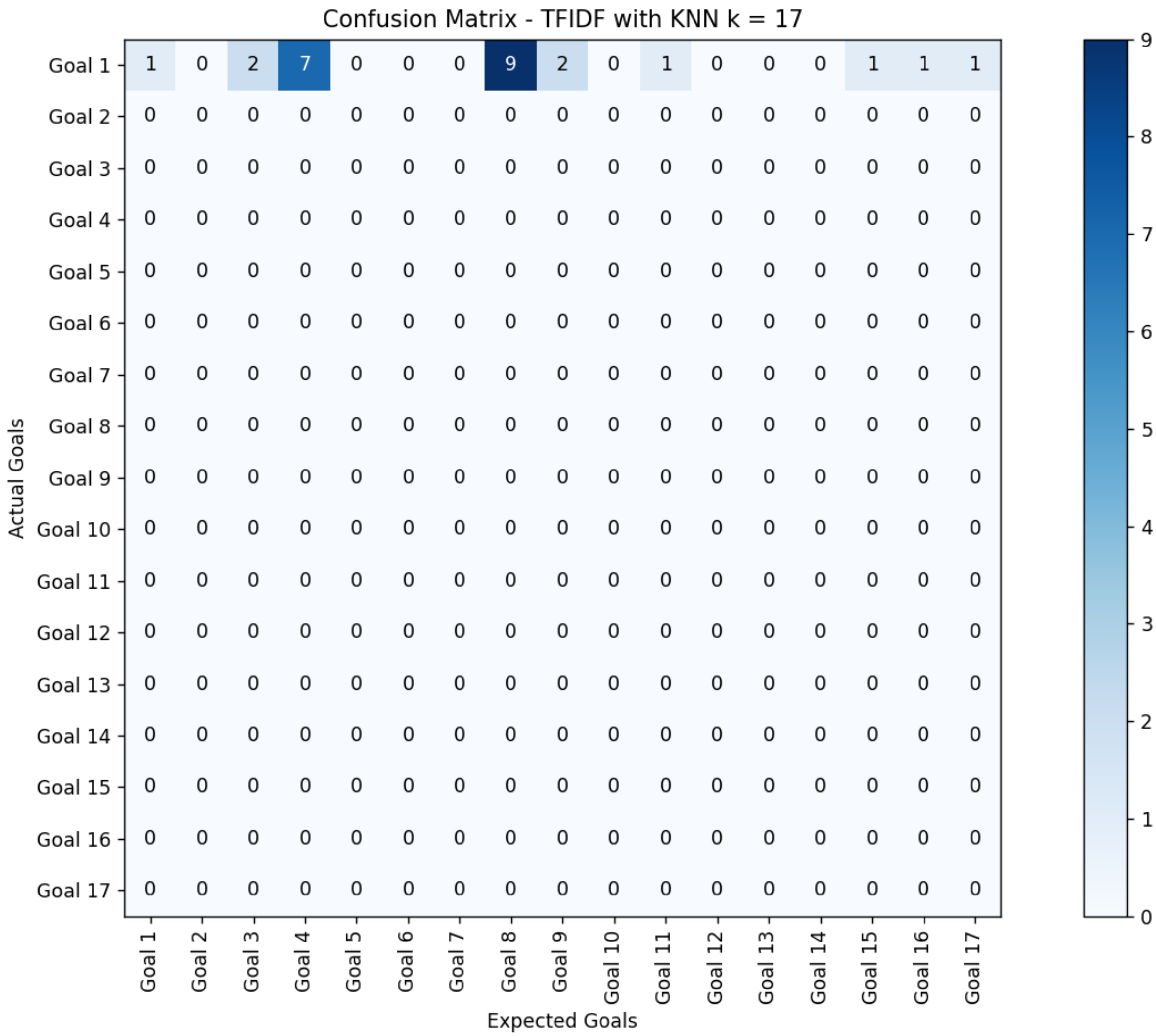


Figure 132: Confusion Matrix for TFIDF with KNN k = 17

The final value of k to be tested is 17, this is the final value since there are only 17 documents in the SDG corpus. Figure 129 now shows that all classified papers are in goal 1. This already shows how inaccurate this parameter is.

	TP	FP	TN	FN
Goal 1	1	0	0	24
Goal 2	0	0	25	0
Goal 3	0	2	23	0
Goal 4	0	7	18	0
Goal 5	0	0	25	0
Goal 6	0	0	25	0
Goal 7	0	0	25	0
Goal 8	0	9	16	0
Goal 9	0	2	23	0
Goal 10	0	0	25	0
Goal 11	0	1	24	0
Goal 12	0	0	25	0
Goal 13	0	0	25	0
Goal 14	0	0	25	0
Goal 15	0	1	24	0
Goal 16	0	1	24	0
Goal 17	0	1	24	0

Figure 133: Values for TFIDF with KNN k = 17

Although all values are in goal 1, there's one paper that is correctly classified, but all other values are all wrongly classified as shown in the FN column for goal 1.

	precision	recall	f1-score	support
Goal 1	1.00	0.04	0.08	25
Goal 2	1.00	1.00	1.00	0
Goal 3	0.00	1.00	0.00	0
Goal 4	0.00	1.00	0.00	0
Goal 5	1.00	1.00	1.00	0
Goal 6	1.00	1.00	1.00	0
Goal 7	1.00	1.00	1.00	0
Goal 8	0.00	1.00	0.00	0
Goal 9	0.00	1.00	0.00	0
Goal 10	1.00	1.00	1.00	0
Goal 11	0.00	1.00	0.00	0
Goal 12	1.00	1.00	1.00	0
Goal 13	1.00	1.00	1.00	0
Goal 14	1.00	1.00	1.00	0
Goal 15	0.00	1.00	0.00	0
Goal 16	0.00	1.00	0.00	0
Goal 17	0.00	1.00	0.00	0
micro avg	0.04	0.04	0.04	25
macro avg	0.53	0.94	0.48	25
weighted avg	1.00	0.04	0.08	25

Accuracy: 0.04

Figure 134: Final Classification Report for TFIDF with KNN k = 17

The final parameter k = 17 shows 0.04 accuracy rate for the SDG classifier. This tells us that k = 10 and k = 17 will always yield a very low accuracy score.

```
ValueError: Expected n_neighbors <= n_samples,  but n_samples = 17, n_neighbors = 18
```

Figure 135: Value Error thrown when k >= 18

Figure 132 shows the error thrown when assigning 18 as the value for k. This is because the number of labels is only 17 to represent the UN SDGs, any values more than that is not accepted.

PERFORMANCE ANALYSIS

This section tests the performance speeds of the different processes in the system. The first test is the training. The execution time of the training is expected to increase every time the number of PDF files in the directory increases. The performance is measured through the start of the execution of the feature and until the execution stops. The library time provided by python was used to determine the start and end time of the feature.

Test Bed Configuration:

- *CPU – AMD Ryzen 5 4600H 3.0 GHz (12 CPUs)*
- *RAM – 16 GB 3.2 MHz*
- *Internet Connection – Converge Fiber 300 mbps*

Training Model

There are exactly 85 papers in the training set, each goal consists of 5 papers. The execution time of processing it until the creation of the TF-IDF takes around this time.

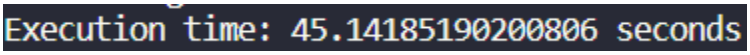


Figure 136: Execution Time for Model Training

SDG Classifier (TFIDF with Cosine Similarity Scoring)

The SDG classifier that incorporates TFIDF with Cosine is tested with 25 different research papers as shown in the testing process. The performance when it comes to classification speed is measured by getting the sum of all classification time of each paper and divides it by the total number of papers in the test set.

Figure 137: TFIDF with Cosine Classification Time

Figure 134 shows the speed of the classification for each paper in the test set using the TFIDF with Cosine model. The total time is 1011.3191576004028 seconds or 16.9 minutes. 16.9 minutes is the speed of the testing process in which there are 25 papers involved. The classification time for each paper takes around 31-36 seconds which is already reasonable considering the model is implemented without the use of any third-party packages.

SDG Classifier (TFIDF Only)

The same metric is used to measure the performance of the SDG Classifier with TFIDF only. The number of research papers in the test set is also the same number, which is 25.

Figure 138: TFIDF Only Classification Speed

TFIDF only model shows a slight boost when it comes to the testing process compared to the TFIDF with Cosine from 16.9 minutes to 16.5 minutes. For each paper classified using TFIDF only, it will take around 31-32 seconds, slightly faster compared to TFIDF with Cosine. This speed is expected since TFIDF with Cosine and TFIDF only incorporates the same method in creating the TFIDF, the only difference is that TFIDF only model uses a rule-based approach in classifying papers.

SDG Classifier (TFIDF with KNN k = 1)

As with the other models, KNN k = 1 is also measured using the same metrics and with the same amount of research papers in the test set.

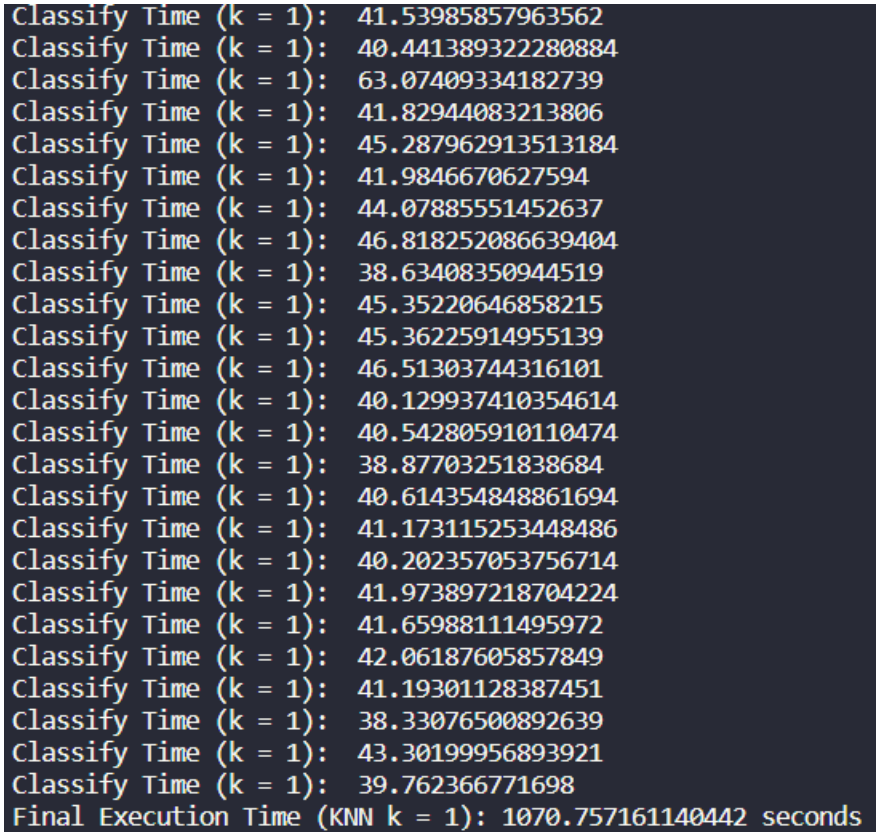


Figure 138: TFIDF with KNN k = 1 Classification Speed

Figure 136 shows the test log of the testing process using KNN with k = 1, in the image shown, classification speed for each paper will take around 38-63 seconds. In total, KNN with k = 1 with 25 research papers will take around 17.8 minutes to finish.

SDG Classifier (TFIDF with KNN k = 2)

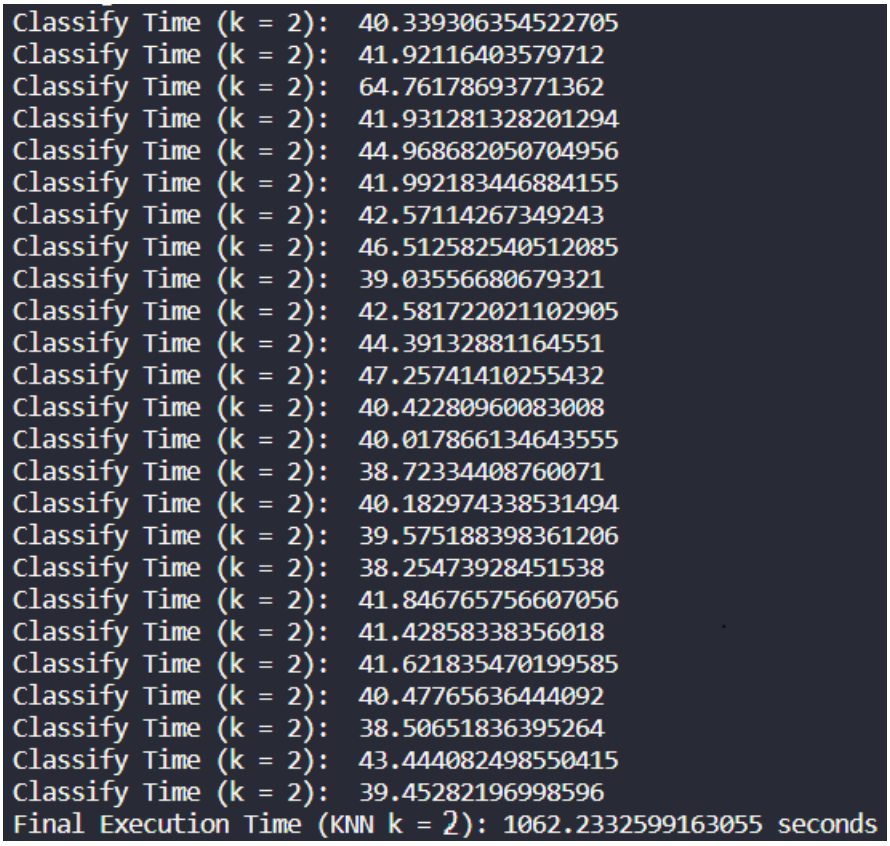


Figure 139: TFIDF with KNN k = 2 Classification Speed

KNN with k = 2 is measured also using the same metric with k = 1. In k = 2, classification speed for each paper will take around 38-64 seconds depending on how much raw data are extracted for each paper. In the 25 papers as the sample size, the testing process will take around 1062.2332599163055 or 17.7 minutes.

SDG Classifier (TFIDF with KNN k = 3)

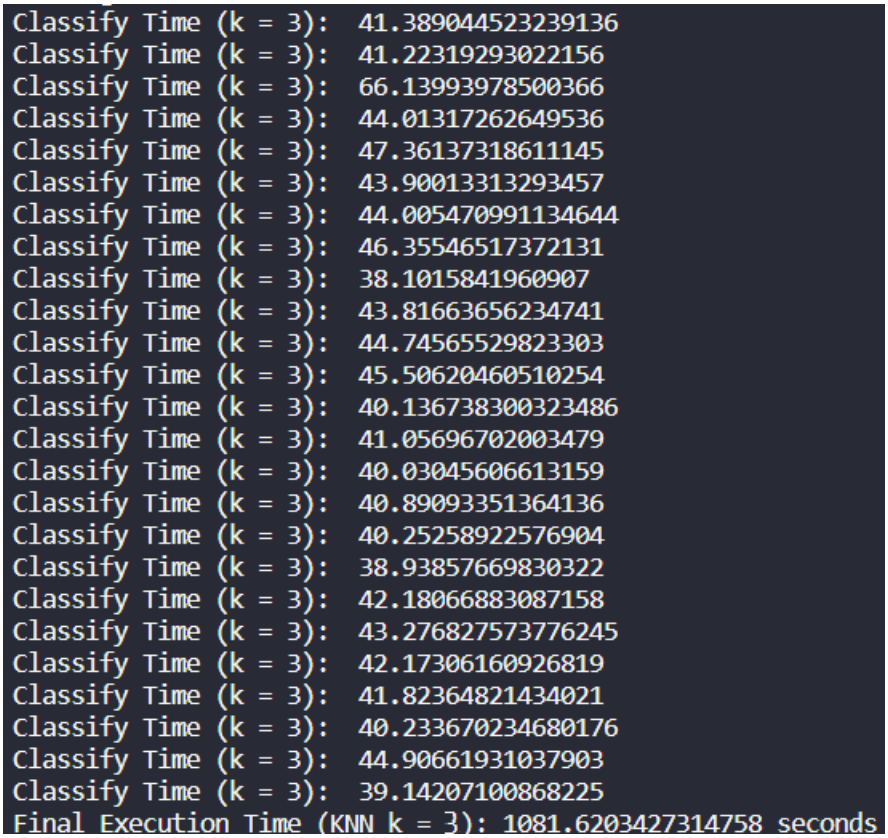


Figure 140: TFIDF with KNN k = 3 Classification Speed

In k = 3, research papers are classified in around 38-66 seconds with the testing process that takes around 1081.6203427314758 or 18.027 mins. This is a bit higher compared to the previous values of k.

SDG Classifier (TFIDF with KNN k = 4)

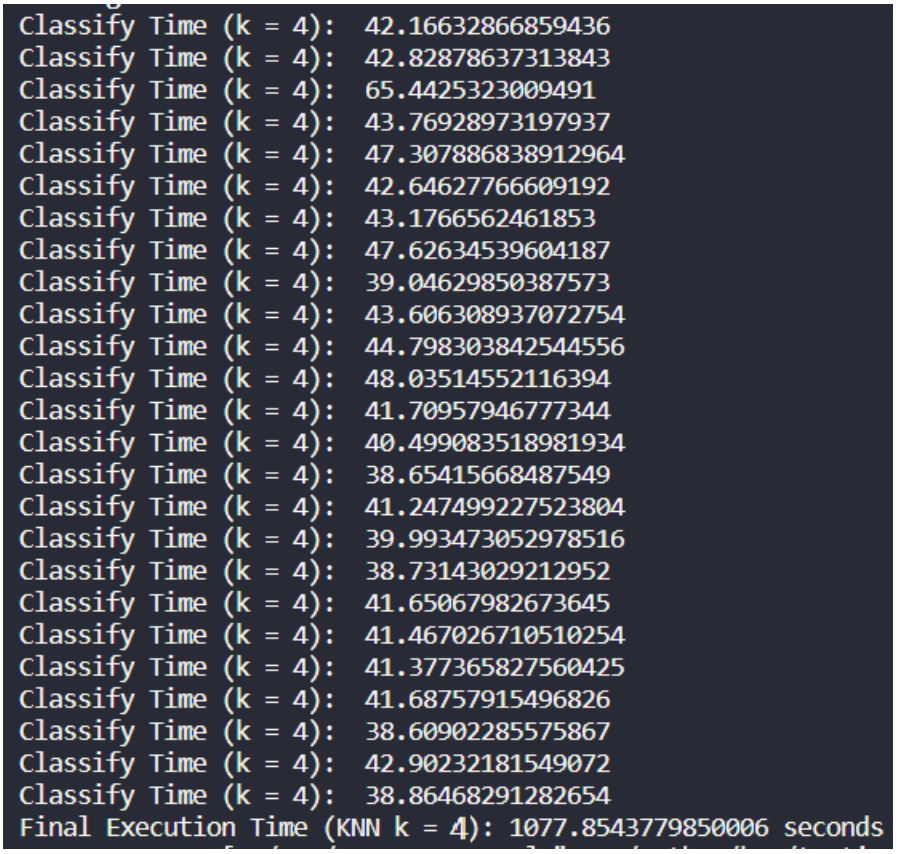


Figure 141: TFIDF with KNN k = 4 Classification Speed

KNN with k = 4 shows that for each paper classified it will take around 38-65 seconds. For testing with 25 papers in the test set, it will take around 1077.8543779850006 or 17.96 mins.

SDG Classifier (TFIDF with KNN k = 5)

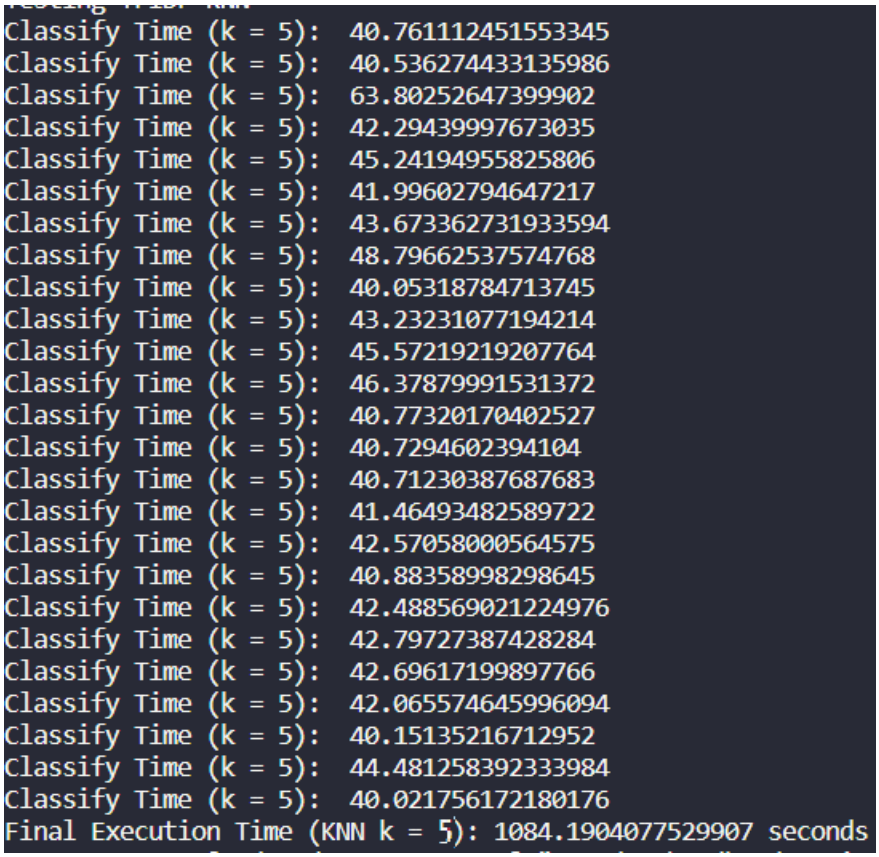


Figure 142: TFIDF with KNN k = 5 Classification Speed

KNN with k = 5 shows yields a performance speed of 40-63 seconds for each paper. The testing process with 25 papers in the data set takes around 1084.19040077529907 seconds or 18.1 minutes.

SDG Classifier (TFIDF with KNN k = 6, 10, and 17)

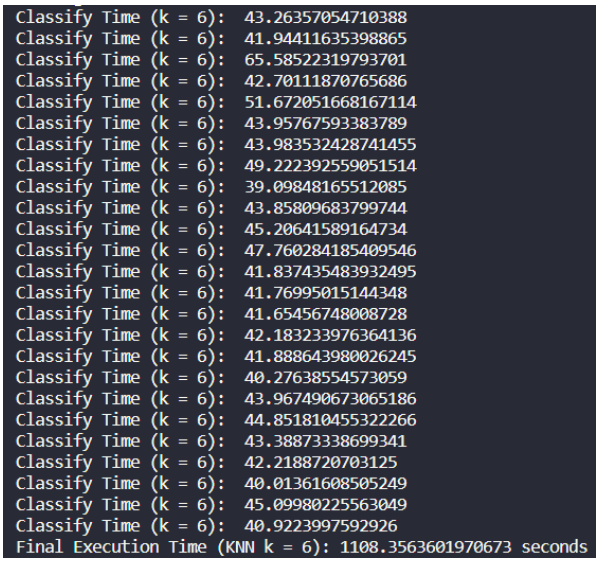


Figure 143: TFIDF with KNN k = 6 Classification Speed

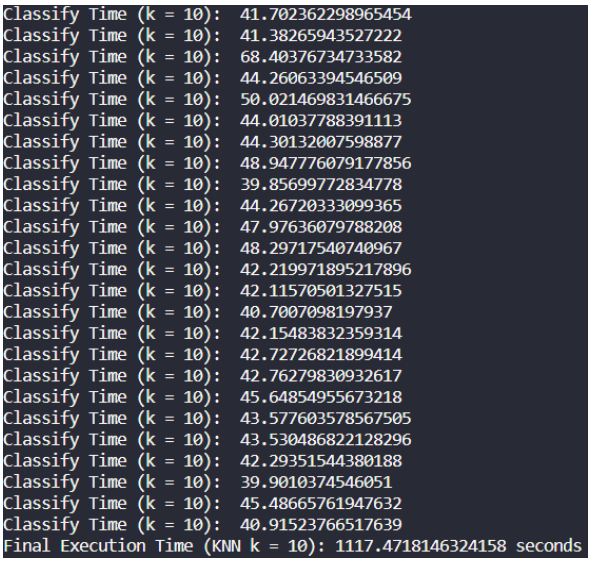


Figure 144: TFIDF with KNN k = 10 Classification Speed

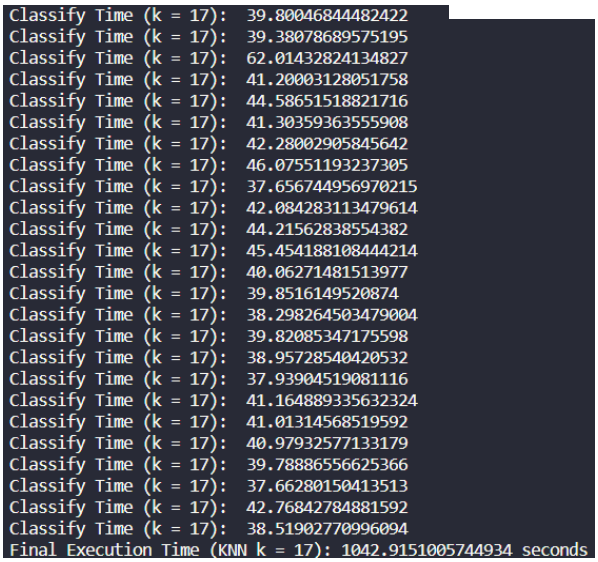


Figure 145: TFIDF with KNN k = 17 Classification Speed

Above shows the results of for k = 6, 10, and 17 in terms of classification speed. k = 6 classifies paper around 39-65 seconds, k = 10 around 39-68 seconds, and k = 17, 37-62 seconds. In terms of the testing process with 25 papers in the test set, k = 6 takes around 18.47 minutes, k = 10 is 18.62 minutes, and lastly, k = 17 will take around 17.38 minutes.

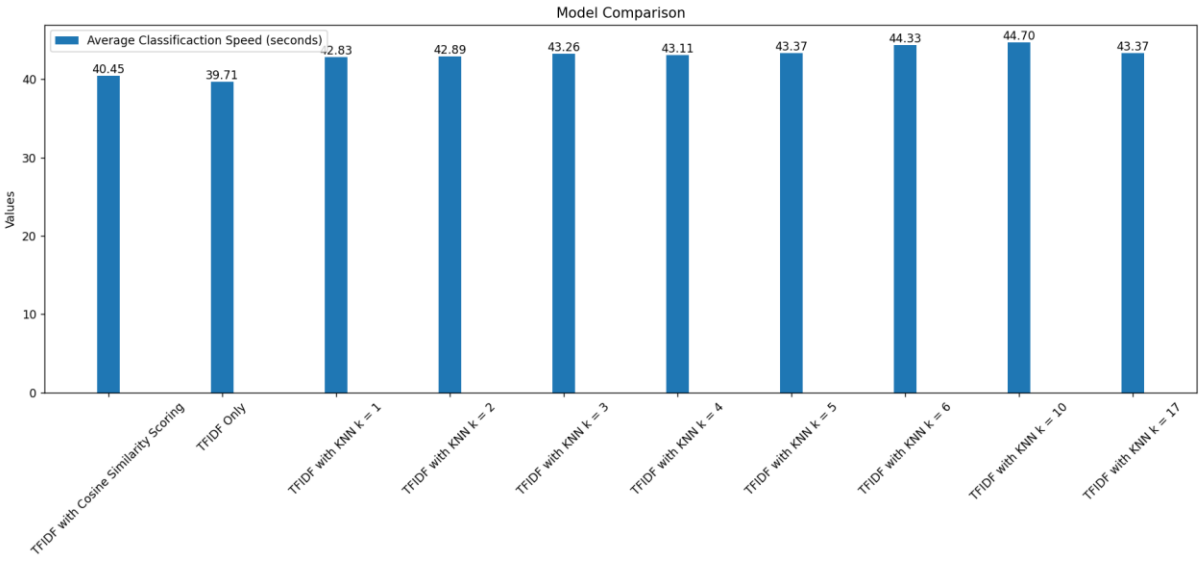


Figure 146: Graph for Average Classification Speed Measured in Seconds

The image above shows the graph of the different models with their respective average classification speed in seconds tested in 25 papers. Here we can see that the TFIDF Only model shows the fastest average classification speed of 39.71 seconds followed by TFIDF with Cosine that is around 40.45 seconds. KNN parameters on the other hand show a significant increase of its average classification speed starting from k = 1 with 42.83 seconds until k = 17 with 43.37 seconds. This gives us a clear view on how each model performs when it comes to its performance given that all models are tested using the same test set of 25 research papers.

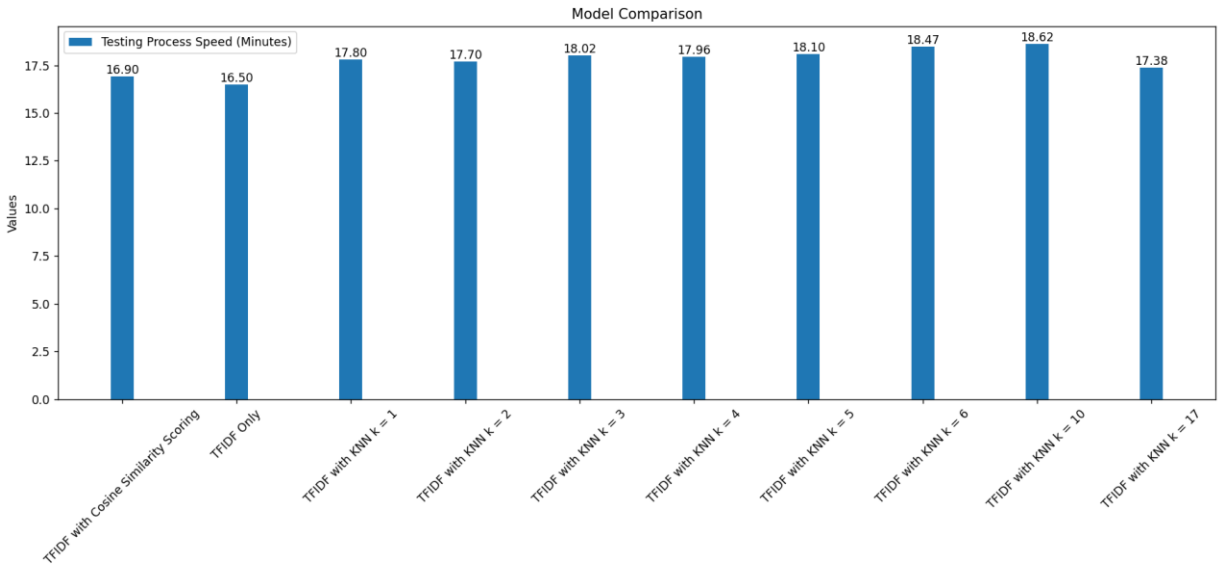


Figure 147: Graph for Testing Process Speed Measured in Minutes

Figure 145 shows the graph for the testing process measured in minutes. As evident in the previous graph, TFIDF only model yields the fastest testing process time in only 16.5 minutes, followed by TFIDF with Cosine and k = 17.

Both graphs show a trend in the KNN with different parameter values. Starting from k =1, the values are consistent to be balanced, until it gets to k = 10 which displays the highest value for both graphs. Once the value of k hits 10, from there it starts to slowly decrease until it hits k = 17 which is the last parameter value for k in this study.

Information Extraction

```
Execution Time Information Extraction:  2.364279270172119
```


Figure 148: Execution Time for Extracting Information

```
{
  "authors": [
    "Rasmil Lloyd P. Augusto",
    "Ronel John S. Tano",
    "Mervyn B. Morales",
    "Christian D. Lastimosa"
  ],
  "department": "School of Computer Studies",
  "published_date": null,
  "title": "LivingGreen: A Botanical Identification Mobile Application with Barter System"
}
```


Figure 149: Extracted Result

The time it takes to extract the information above is 2.37 seconds. This is a reasonable speed since the system utilizes a rule-based approach in extracting this information.


```
{
  "abstract": "  ABSTRACT  Counseling is one of the important services that a school should have. It helps students whenever they have problems and also provides services that can also help the mental health of students. However, during the pandemic, face-to-face counseling was somehow prohibited. Because of this, a system was designed to help counselors, teachers, and students connect with each other. Follow App is a web application that will help teachers manually refer students virtually, counselors manage referrals, and automatically refer students based on data of EDP through the institution's School Information Service and Learning Management System (LMS). The system will schedule students for a counseling session and will provide a platform for video meetings. As a result of this endeavor, the researchers ensured that the system was ready for deployment; however, the system can still be improved further.
                                4  ",
  "appendedData": "  ABSTRACT  Counseling is one of the important services that a school should have. It helps students whenever they have problems and also provides services that can also help the mental health of students. However, during the pandemic, face-to-face counseling was somehow prohibited. Because of this, a system was designed to help counselors, teachers, and students connect with each other. Follow App is a web application that will help teachers manually refer students virtually, counselors manage referrals, and automatically refer students based on data of EDP through the institution's School Information Service and Learning Management System (LMS). The system will schedule students for a counseling session and will provide a platform for video meetings. As a result of this endeavor, the researchers ensured that the system was ready for deployment; however, the system can still be improved further.
                                4  CHAPTER I  INTRODUCTION  Rationale of the Study  The Guidance Office plays a vital role in every school. The guidance office comprises guidance counselors in charge of counseling students whenever the latter encounter personal and academic-related problems and conflicts. The function of school counselors in ensuring student achievement is fundamental. (Lapan, Gysbers, & Kayson, 2007; Stone & Dahir, 2006). Most counseling in schools is done face-to-face through teachers' referrals or individual appointments in which the students need to go to the office to have the counseling session. At the University of San Jose-Recoletos (USJ-R), students are given the appointment slip and are called to the guidance office for their session, and then feedback is given to each teacher. But during the pandemic, the system of counseling has changed. Due to the fact that there were restrictions for physical encounters, the implementation of online classes has been introduced. In lieu of this, counseling session has been shifted to online as well to meet the student's need even in the online set-up. Follow App is a web application for counseling that will help teachers, counselors, and students have the session in an online set-up. The application focuses on scheduling students for the session, allowing counselors to give feedback online and refer students based on academic performance and behavior.  10  ",
  "introduction": "  CHAPTER I  INTRODUCTION  Rationale of the Study  The Guidance Office plays a vital role in every school. The guidance office comprises guidance counselors in charge of counseling students whenever the latter encounter personal and academic-related problems and conflicts. The function of school counselors in ensuring student achievement is fundamental. (Lapan, Gysbers, & Kayson, 2007; Stone & Dahir, 2006). Most counseling in schools is done face-to-face through teachers' referrals or individual appointments in which the students need to go to the office to have the counseling session. At the University of San Jose-Recoletos (USJ-R), students are given the appointment slip and are called to the guidance office for their session, and then feedback is given to each teacher. But during the pandemic, the system of counseling has changed. Due to the fact that there were restrictions for physical encounters, the implementation of online classes has been introduced. In lieu of this, counseling session has been shifted to online as well to meet the student's need even in the online set-up. Follow App is a web application for counseling that will help teachers, counselors, and students have the session in an online set-up. The application focuses on scheduling students for the session, allowing counselors to give feedback online and refer students based on academic performance and behavior.  10  ",
  "method": " "
```


Figure 150: Extracted Result

```
Execution time: 1.9253780841827393 seconds
```

Figure 151: Execution Time

It takes around 1.9 seconds to extract the Abstract, Introduction, and the Research Methodology of the paper. This is expected since the system will only scrape until the 10th page to increase computational efficiency.

CHAPTER IV

SUMMARY, FINDINGS, CONCLUSION, AND RECOMMENDATIONS

SUMMARY OF FINDINGS

EUL is a unified digital repository system that is intended to incorporate an automatic SDG classification of approved USJ-R research papers. This falls under the field of Natural Language Processing, specifically Information Extraction. Extracting Information using a third-party package poses problems that should not occur, some words are cut off, and there are some unnecessary spaces when extracting words. However, if the paper follows the institution's format, there should be no problem in extracting the necessary information. The performance of the classifier revolves around the data used for training, thus having 85 papers for the training set is also ideal considering that this will affect the computational performance of the classifier. This gives a question to the study, should the system retrain the model every time it is finished classifying? Or rely heavily on the 85 papers for training set. This entirely depends on the nature of the training set and as for now, the 85 papers for the training set are sufficient to support the accuracy of the classifier used in this study.

Results for TFIDF with Cosine, TFIDF Only, and TFIDF with KNN

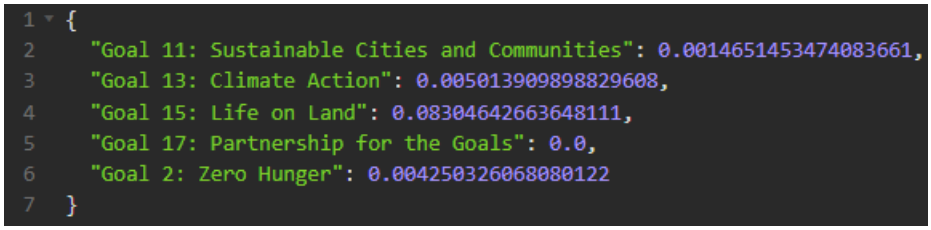


Figure x: Final Classification Result for TFIDF Only

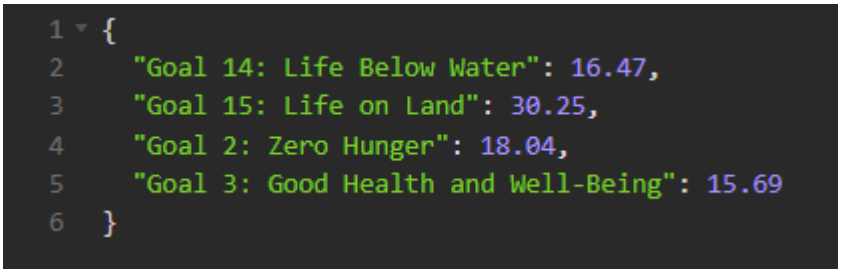


Figure 98: Final Classification Result for TFIDF with Cosine

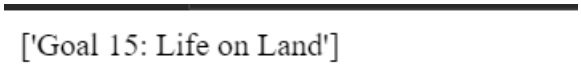


Figure x: Final Classification Result for KNN Across All Parameters

The three figures illustrates how each model displays the final classification results. TFIDF only shows the TFIDF scores for each goal that contains and matches the specific keywords in the predefined rules. KNN model only displays the final classification label. TFIDF with Cosine Scores displays the similarity scores of the new document with respect to each goal in the corpus. These results were used in the consideration of the final model that the study chose to implement.

Accuracy and Performance Discussion

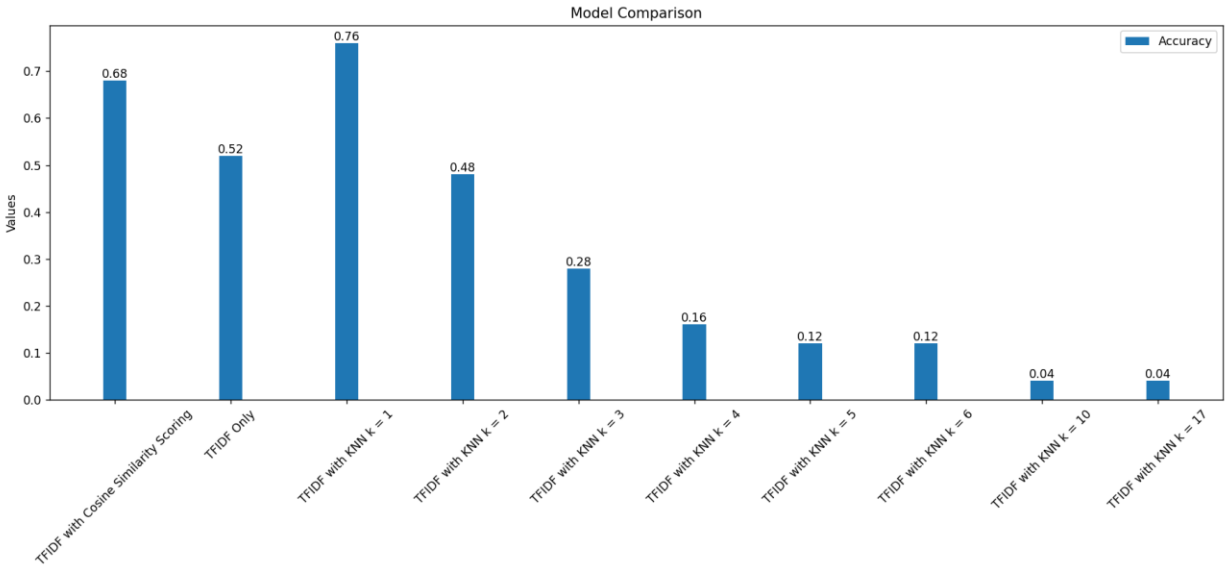


Figure 152: Accuracy Model Comparison Graph

The graph in figure above illustrates the gathered accuracy scores for each model compared in the study. The highest accuracy score, which is .76 was garnered by the KNN model with k = 1, followed by TFIDF with Cosine with .68, and then the TFIDF only with .52. As clearly shown in the graph there's a downward trend of accuracy starting from k = 2 up until k = 17. This gives us a very clear observation that the optimal value for k in this study is 1. The downward trend is caused by the model's overfitting, this shows that the model can't accurately predict papers if the value of k is greater than or equal to 2. This is evident in the testing process that shows k = 17, the confusion matrix for k = 17 showed that all papers are classified into only one goal which is Goal 1: No Poverty. Though it is only shown to be the k = 17 that represents the overfitting of the model, the trend starts when the k = 10 as shown in the accuracy graph above and the confusion matrix for k = 10.

Model	Accuracy Score	Average Classification Speed (seconds)
TFIDF Only	0.52	39.71
TFIDF with Cosine	0.68	40.45
KNN with K = 1	0.76	42.83
KNN with K = 2	0.48	42.89
KNN with K = 4	0.16	43.11
KNN with K = 3	0.28	43.26
KNN with K = 5	0.12	43.37
KNN with K = 17	0.04	43.37
KNN with K = 6	0.12	44.33
KNN with K = 10	0.04	44.70

Figure 153: Sorted Summarized Information for all Models.

Figure 153 shows the sorted information in ascending order with respect to the average classification speed. Here we can see that the top model for average classification speed is the TFIDF only but has a lesser accuracy score. If we sort the table based on the accuracy score, the expected top model would be the KNN with k = 1 with .76. Now the sweet spot for both accuracy and performance is the model that uses TFIDF with Cosine Similarity Scoring. Although the difference with k =1 in terms of classification speed is 2 seconds, this is still crucial when dealing with a lot of research papers.

CONCLUSION

EUL has achieved its purpose of creating a unified digital research repository system and an SDG classification framework. It can give accurate results in the form of SDG goals which are also labeled with their respective scores to give the users a better idea on the relevance of their paper with respect to the UN SDGs. In addition, EUL also successfully created a comparison framework for different working models that classifies papers. These models are the following:

- TFIDF with Cosine Similarity Scores
- TFIDF Only
- KNN with k = 1,2,3,4,5,6,10, and 17

The use of TF-IDF with Cosine Similarity Scoring, and the KNN Algorithm with k = 1, prove to perform well in supervised document classification. TF-IDF of the SDG corpus demonstrated a key role in the classification process.

To align with the study's objective, the study focuses on the use of TFIDF with Cosine Similarity Scoring instead of the KNN with k = 1. There are several factors for this conclusion, one is the semantic similarity. The study aims to give users a better visualization on relationship of their paper with respect to the UN SDGs. Using the cosine similarity metric achieved this objective, this is clearly shown in the study that instead of just giving the actual classified goal, which is the result of the KNN model, the TFIDF with Cosine Similarity Scoring also calculates and displays to the user the percentage on how much the paper belongs to a specific goal.

Last factor is the balance between accuracy and performance. As mentioned in the previous chapter, TFIDF with Cosine Similarity Scoring is the model that captures a

good accuracy with fair result in the performance, compared to the KNN with k = 1, though better in accuracy by .8 but offers much slower performance compared to TFIDF with Cosine.

TFIDF with Cosine Scores offers a more informative approach to SDG document classification and leveraging on the balance between the accuracy and its performance compared to using KNN.

RECOMMENDATIONS

To alleviate the problem when extracting information, the usage of OCR is recommended. In improving the computational performance of the classifier, various optimization techniques may be used just like multi-threading. Adding more PDF files to each goal in the training data will greatly improve the accuracy of the classifier because it will improve reliability.

To give more definitive meaning on the comparison of the models, the amount of sample research papers on the test set may be increase from 25 to around 50 above papers.

To improve the performance of the system when it comes to classification, other sophisticated classification algorithms like SVM or Random Forest may be included.

BIBLIOGRAPHY

[1] History of artificial intelligence - javatpoint. www.javatpoint.com. (n.d.). Retrieved July 25, 2022, from https://www.javatpoint.com/history-of-artificial-intelligence.

[2] Editor. (2021, November 17). *"Document classification with machine learning: Computer vision, OCR, NLP, and other techniques*." ("Document Classification With Machine Learning: Computer ... ") AltexSoft. Retrieved August 4, 2022, from https://www.altexsoft.com/blog/document-classification/.

[3] Burns, E. (2021, March 30). *What is machine learning and why is it important?* SearchEnterpriseAI. Retrieved July 25, 2022, from https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

[4] Petersson, D. (2021, March 26). *What is supervised learning?* SearchEnterpriseAI. Retrieved August 4, 2022, from https://www.techtarget.com/searchenterpriseai/definition/supervised-learning

[5] *Classification algorithm in Machine Learning - Javatpoint*. (n.d.). Retrieved August 4, 2022, from https://www.javatpoint.com/classification-algorithm-in-machine-learning

[6] Pratt, M. K. (2020, July 8). *What is unsupervised learning?* SearchEnterpriseAI. Retrieved July 28, 2022, from https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning

[7] Webb, G. I. (1970, January 1). *Naïve bayes*. SpringerLink. Retrieved July 27, 2022, from https://link.springer.com/10.1007%2F978-0-387-30164-8_576

[8] (LEDU), E. E. (2018, September 12). *Understanding K-means clustering in machine learning*. Medium. Retrieved August 4, 2022, from https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[9] Chen, S. (2020, May 26). *Getting started with text vectorization*. Medium. Retrieved August 1, 2022, from https://towardsdatascience.com/getting-started-with-text-vectorization-2f2efbec6685#:~:text=Text%20Vectorization%20is%20the%20process,(L1)%20Normalized%20Term%20Frequency

[10] *Understanding TF-ID: A simple introduction*. MonkeyLearn Blog. (2019, May 10). Retrieved August 3, 2022, from https://monkeylearn.com/blog/what-is-tf-idf/#:~:text=TF%2DIDF%20

[11] By: IBM Cloud Education. (n.d.). *What is natural language processing?* IBM. Retrieved August 4, 2022, from https://www.ibm.com/cloud/learn/natural-language-processing

[12] *What is optical character recognition (OCR)?* IBM. (n.d.). Retrieved July 28, 2022, from https://www.ibm.com/cloud/blog/optical-character-recognition

[13] Kim, S.-W., & Gil, J.-M. (2019, August 26). *Research paper classification systems based on TF-IDF and LDA Schemes - human-centric computing and Information Sciences*. SpringerOpen. Retrieved August 4, 2022, from https://hcis-journal.springeropen.com/articles/10.1186/s13673-019-0192-7

[14] Mohsen Taheriyan University of Southern California, Taheriyan, M., California, U. of S., Nebraska, U. of, Army, U. S., Massachusetts, U. of, Oklahoma, U. of, Saic, Maryland, U. of, Nasa, & Metrics, O. M. V. A. (2011, August 1). *Subject classification of research papers based on Interrelationships Analysis: Proceedings of the 2011 Workshop on Knowledge Discovery, modeling and Simulation*. ("Subject classification of research papers based on interrelationships ...") ACM Conferences. Retrieved July 30, 2022, from https://dl.acm.org/doi/10.1145/2023568.2023579

[15] Maheshwari, A. (2018, July 17). *Report on text classification using CNN, RNN & Han*. Medium. Retrieved August 28, 2022, from https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f#:~:text=Text%20Classification%20Using%20Convolutional%20Neural,inspired%20by%20animal%20visual%20cortex.