***Umanga Niroula***

**University ID: 14825284**

**Email: 230272@softwarica.edu.np**

Video Presentation Link:

Github Link :

*STW7071CEM - information Retrieval*

Table of Contents

[1. Introduction 2](#_Toc158684425)

[2. Part 1: Search Engine 3](#_Toc158684426)

[2.1. Vertical Search Engine 3](#_Toc158684427)

[2.2. Web Crawling 3](#_Toc158684428)

[2.3. Data Extraction 4](#_Toc158684429)

[2.4. Implementation 5](#_Toc158684430)

[2.4.1. Polite Crawling 5](#_Toc158684431)

[2.4.2. Scheduled Task 6](#_Toc158684432)

[2.4.3. Pre-processing 7](#_Toc158684433)

[2.4.4. User Interface 9](#_Toc158684434)

[2.5. Results and Discussion 10](#_Toc158684435)

[3. Part 2: Document Clustering 11](#_Toc158684436)

[3.1. Source Selection 11](#_Toc158684437)

[3.2. User Interface 12](#_Toc158684438)

[3.3. Assignment Process 14](#_Toc158684439)

[3.4. Robustness Analysis 16](#_Toc158684440)

[4. Conclusion 17](#_Toc158684441)

**Table of Figures**

[Figure 1: Coventry University's Centre for Health and Life Sciences Website. 4](#_Toc158721537)

[Figure 2: Publication Details in the Target Website. 5](#_Toc158721538)

[Figure 3: Fetching the Web Page Containing the List of Publications. 6](#_Toc158721539)

[Figure 4: Target URL for Crawling. 6](#_Toc158721540)

[Figure 5: Creating the Index Folder if it Doesn't Already Exist. 7](#_Toc158721541)

[Figure 6: Try-Except Block to Handle Potential Errors. 7](#_Toc158721542)

[Figure 7: Scheduling Function to Run Once per Week. 8](#_Toc158721543)

[Figure 8: Pre-Processing for Crawled Data. 9](#_Toc158721544)

[Figure 9: Pre-Processing for User Queries. 10](#_Toc158721545)

[Figure 10: User Interface. 11](#_Toc158721546)

[Figure 11: App.py File for Utilizing Flask Web Application. 12](#_Toc158721547)

[Figure 12: Sample of Collected Dataset. 13](#_Toc158721548)

[Figure 13: Tkinter GUI. 14](#_Toc158721549)

[Figure 14: GUI Showing Error Message when No Input Provided. 15](#_Toc158721550)

[Figure 15: Successful Working of System. 16](#_Toc158721551)

[Figure 16: Preprocessing Dataset. 17](#_Toc158721552)

[Figure 17: TF-IDF Vectorization. 17](#_Toc158721553)

[Figure 18: K-means Clustering. 18](#_Toc158721554)

[Figure 19: Cluster Prediction for New Documents. 18](#_Toc158721555)

[Figure 20: Short Input Test. 19](#_Toc158721556)

[Figure 21: Long Format Input. 20](#_Toc158721557)

[Figure 22: Inputs without Stop Words. 21](#_Toc158721558)

[Figure 23: Inputs with Stop Words. 22](#_Toc158721559)

[Figure 24: Health Realted Topic Input. 23](#_Toc158721560)

# Introduction

The prime objective of this report is to put forward the development and assessment of two different systems: a vertical search engine design to crawl and retrieve publication data form the Research Centre for Health and Life Sciences (RCHL) of Coventry University, and a document clustering system that classifies the clusters based on user input through the help of machine learning algorithms like K-means clustering. Further, both the system is designed to meet certain criteria in the field of information retrieval.

The first task, i.e., to create a vertical search engine quite similar to Google Scholar is developed, just how the Google Scholar finds publication done form various authors, so does the vertical search engine created for task one. The only difference is that it utilizes crawling mechanism to exclusively focus on publications made in the Coventry University’s RCHL. The system crawls various data on publications such as the title, authors name(s), publication year, URL to the publication link, and author’s profile URL. Furthermore, this system uses a polite crawling approach where the robots.txt page is in consideration while respecting the limitation it possesses and making sure the crawling time period is also elongated such that no disturbance in the system.

In the second task, a document clustering system is created to classify different documents formats into namely three categories Sport, Health, and Business. The clustering system utilize a standard clustering method which is the K-means Clustering. First, a dataset is formed through gathering various sports, health, and business-related news, then with the help of K-means clustering each of the data gathered is classified into one of three categories, this is training the system, then a GUI is developed which further classifies users input into one of three categories.

# Part 1: Search Engine

## Vertical Search Engine

A vertical search engine concentrates on a certain industry, topic, or kind of material, resulting in more focused results than broad search engines such as Google. Vertical search engines are widely used in specialized industries such as academics, healthcare, and e-commerce to obtain information that is suited to specific requirements (Long & Chang, 2014).

## Web Crawling

online crawling, often known as scraping or harvesting, is the automated extraction of information from online sites. Web crawling is required to collect data from numerous online sources, allowing for activities such as content indexing, data analysis, and website update tracking (Khder, 2021).

In the context of this report, the website in which the crawling action is going to be performed is the Research Centre for Health and Life Sciences (RCHL) of Coventry University website, shown in the following figure 1.

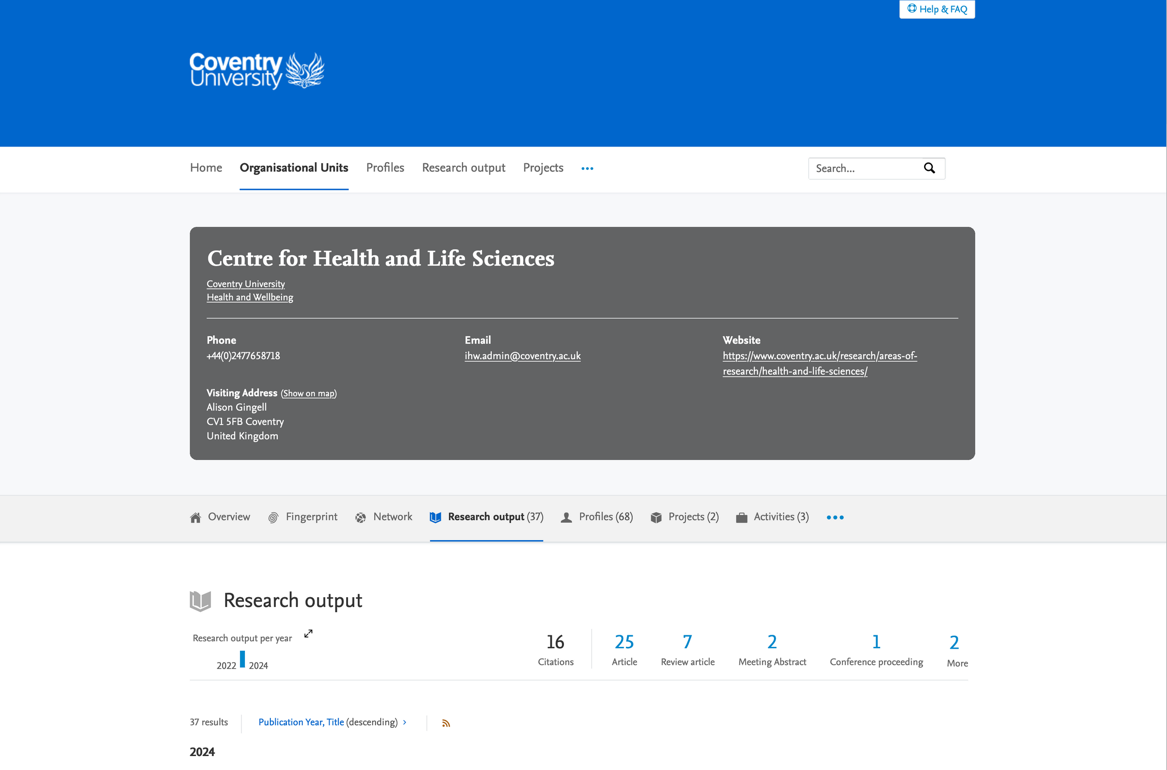


Figure 1: Coventry University's Centre for Health and Life Sciences Website.

## Data Extraction

Data extraction is the process of collecting particular information from a dataset or source, which is commonly done automatically using techniques such as web scraping or database querying. Data extraction from web pages uses a variety of approaches, including text parsing, regular expressions, and HTML parsing tools such as BeautifulSoup.

In the context of this report, title, authors name(s), publication year, URL to the publication, and author’s profile URL are the data that are extracted for the website shown in figure 1. These data are present in the following format in the website, that is illustrated by the following figure 2. The red box represents publication details provided in the website.



Figure 2: Publication Details in the Target Website.

## Implementation

### Polite Crawling

This search engine's crawling method follows the rules of civility. This means it follows the restrictions outlined in the target website's robots.txt file and avoids overloading the servers with queries. This is done by establishing a suitable delay between queries and ensuring that our crawler adheres to the website's standards.

In the code, the problem is tackled by utilizing the requests library appropriately, avoiding sending requests too frequently or in a way that the server may view as pushy. Also using error handling tools to address situations in which the server rejects our requests owing to high traffic.

The following figure 3, with the highlighted portion shows the utilization of ‘***request.get()***’ that helps to fetch all the information related to publication firm the URL provided.



Figure 3: Fetching the Web Page Containing the List of Publications.

The URL is also specified in the following figure 4 with the red box, it is the URL which contains all the publications record in the Coventry University.



Figure 4: Target URL for Crawling.

Further, the following figure 5 represents the code snippet for ensuring the creation of index folder, if it already doesn’t exist. This step is important as checking for directories is important before creating one to avoid any upcoming errors relating to the folder already existing.

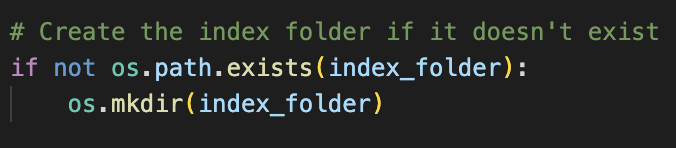


Figure 5: Creating the Index Folder if it Doesn't Already Exist.

In the following section of code demonstrated by figure 6, the utilization of try-except block to handle any failures, notably the ‘***LockError***’ that might occur while trying to add pages to the Whoosh index is done. This issue might be caused by numerous processes attempting to access the index at the same time, causing a problem to the server. If a ‘***LockError***’ occurs, then and error message is published while trying to manually clear up any lock files that might be causing the problem. This is a form of ensuring realiability to the crawling operation that is about to happen.

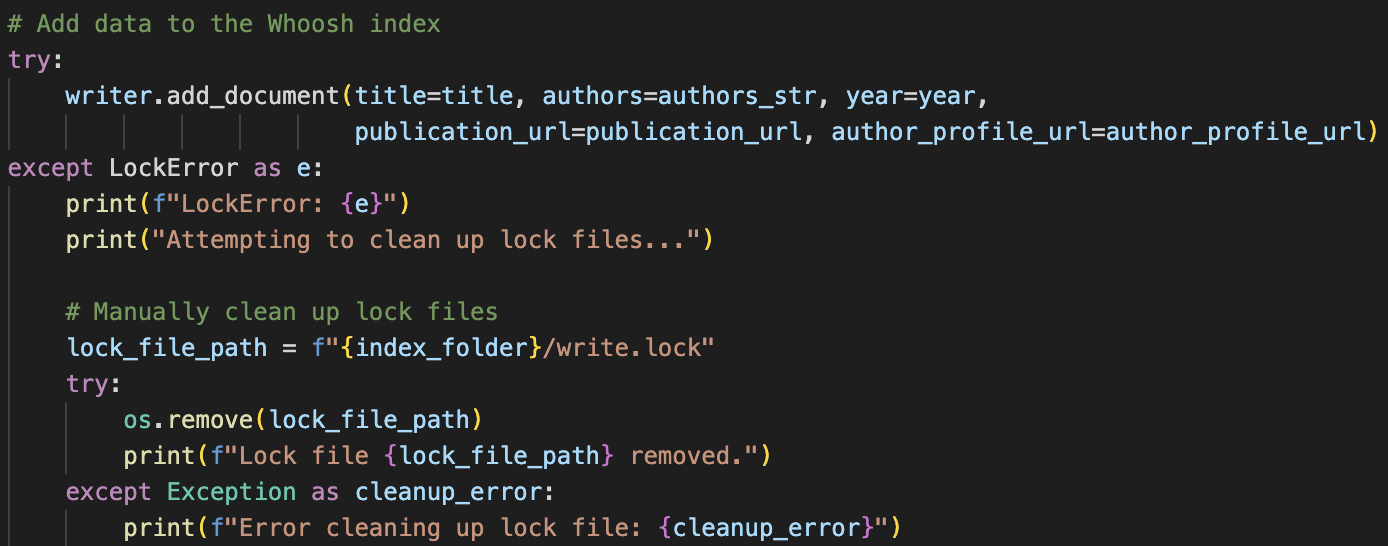


Figure 6: Try-Except Block to Handle Potential Errors.

### Scheduled Task

This search engine has a scheduled task feature that automatically updates the index with fresh publications. The duration of crawling is set to once per week, ensuring that the index is kept up to date without manual intervention. This automation improves system efficiency and guarantees that consumers receive the most up-to-date information throughout their searches.

Python's schedule library is used to implement scheduling. A function is created to execute the index update and schedule it to run at regular intervals with the “***schedule.every().week.do()***” method. Furthermore, figure 7 shows a part of the code that utilizes the schedule library to schedule the ‘***perform\_index\_update***’ function to run once a week. This is accomplished by utilizing the ‘***every().week.do()***’ method, which defines the frequency with which the function should be performed.

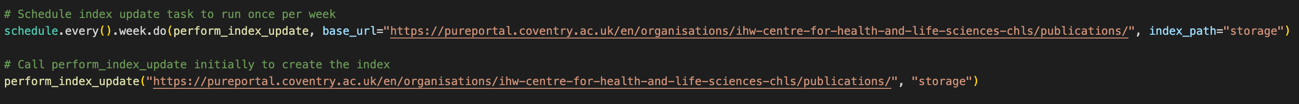


Figure 7: Scheduling Function to Run Once per Week.

Moreover, the '***perform\_index\_update'*** function receives the relevant inputs, which include the base URL and the path to the index folder. This guarantees that the index is often updated with new articles from the provided source URL. Further, the '***perform\_index\_update***' function is used first to construct the index before scheduling the operation. This guarantees that the index is loaded with the initial set of articles before the scheduled updates start.

### Pre-processing

Pre-processing is an important stage in information retrieval systems that improves the quality of search results and the user experience. This search engine performs pre-processing actions on both crawling material and user requests.

Further, ‘***BeautifulSoup***’ is used to extract relevant information from web pages, such as authors, publication year, and title. The data is then cleaned and normalized to ensure that indexing is consistent and accurate. Similarly, when processing user queries, pre-processing techniques like as tokenization, stop word removal, and stemming has helped in improving the relevancy of search results. This guarantees that consumers obtain meaningful and accurate responses to their requests.

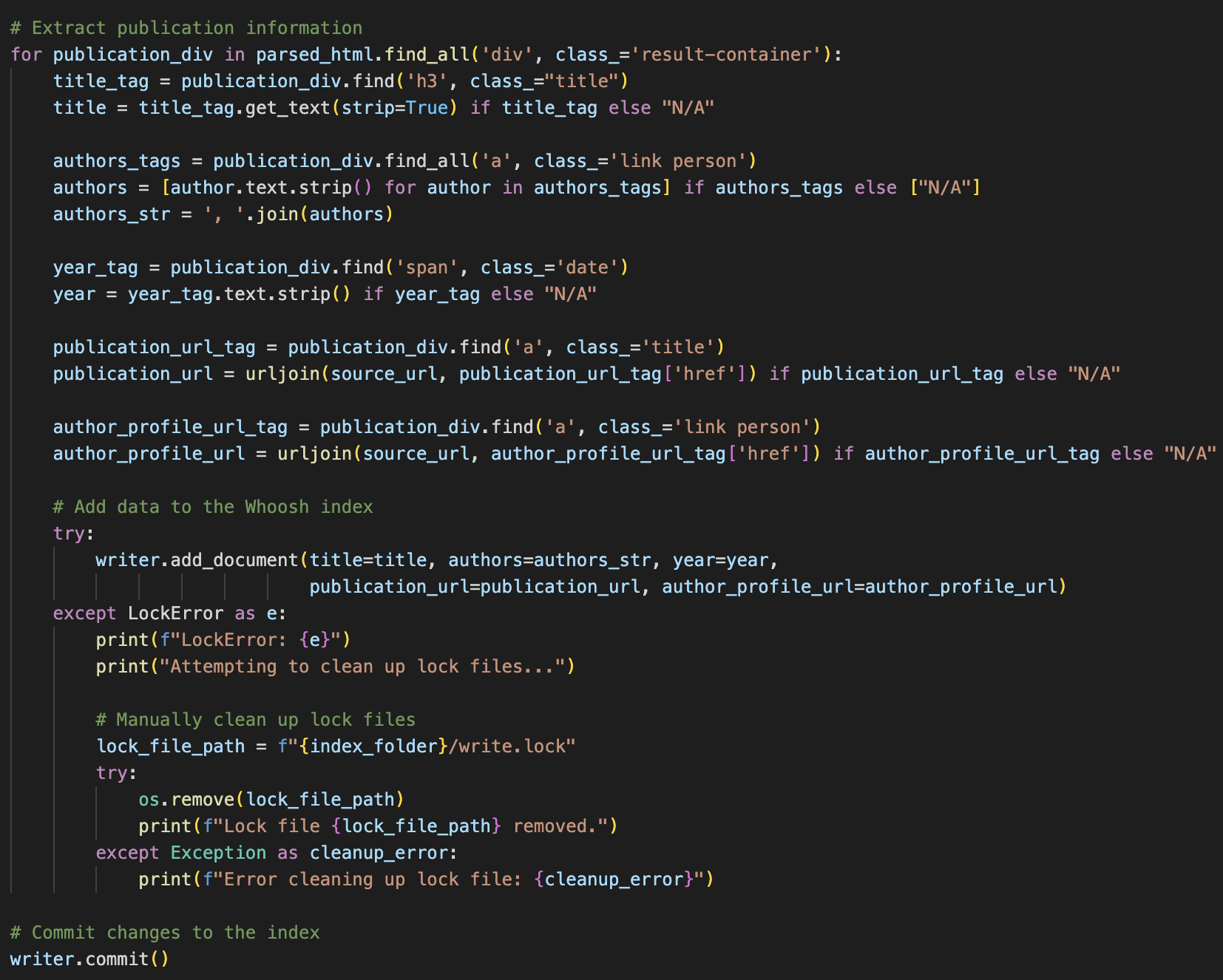


Figure 8: Pre-Processing for Crawled Data.

In Figure 8, the code snippet shows the use of '***BeautifulSoup***' to parse the HTML of a web page that contains publication information. Here, relevant data like as title, authors, publication year, publication URL, and author profile URL are extracted from the parsed HTML. Following extraction, the data is added to the Whoosh index.



Figure 9: Pre-Processing for User Queries.

In this section of the code shown in figure 9, the '**execute\_search**' function is defined to search the index depending on user requests. The user's query is processed and used to search with Whoosh's searcher object.

### User Interface

This search engine's user interface is meant to be simple and straightforward. Users can input queries/keywords relating to the resources they want to find, and the system will return relevant articles limited to RCHL member publications. A command-line interface is given for users to enter their queries, and the system shows the search results in an organized fashion. Users may also simply traverse the results by clicking on the offered links, which improves the system's overall usefulness.

Following figure, show the user interface for the search engine, where there is a search bar where user is allowed to input text and click in the search button to start the searching, As the user provides relevant information’s, i.e., that publication information that are present in the portal of Coventry University. Further, the interface is very user-friendly as not much information’s are kept on the page.

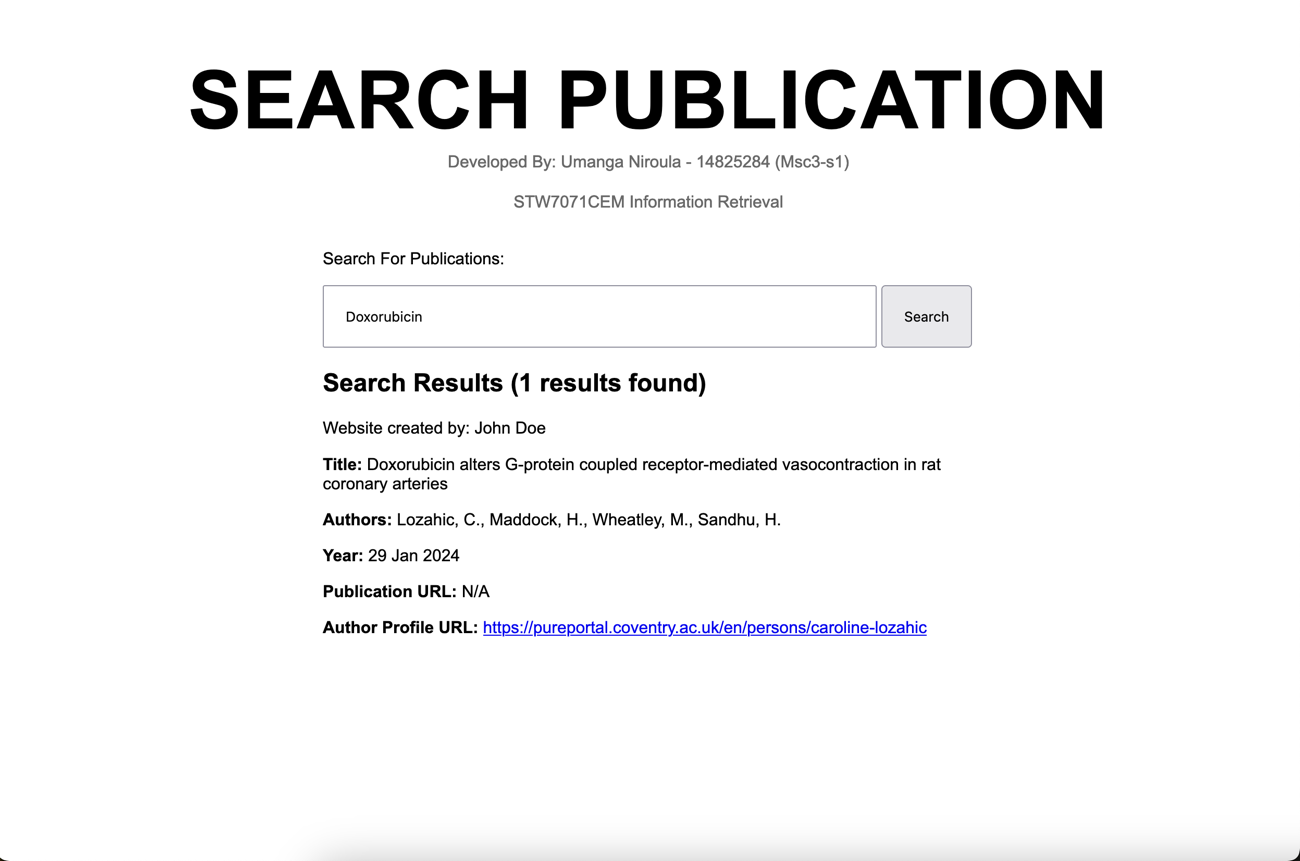


Figure 10: User Interface.

This interface was constructed via configuring a **Flask web application**, which included generating a Python file named '***app.py***' in which the Flask application and routes could be defined. Routes handle various URLs, like as providing an HTML file or accepting form submissions. To serve an HTML page, a route in '***app.py***' is specified to render the HTML file. When a user accesses a given URL, Flask renders the HTML file and provides it to the user's browser, as seen in the image below.

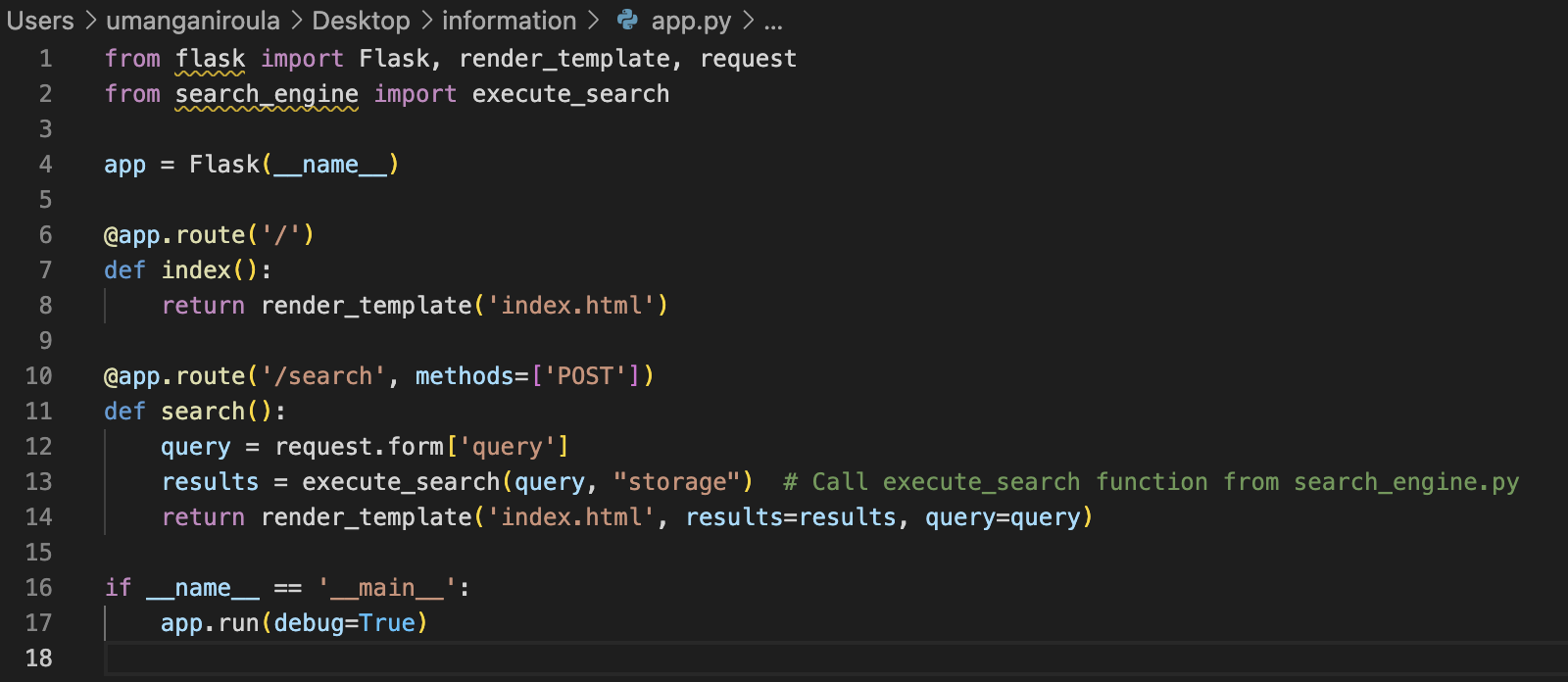


Figure 11: App.py File for Utilizing Flask Web Application.

# Part 2: Document Clustering

## Source Selection

Task 2 begins with cautious source selection to maintain the integrity of the document collection. Documents are carefully acquired from publically available sources such as BBC news websites, all while strictly conforming to copyright restrictions and conditions of use. Transparent citation techniques are used to credit the original producers of the works and uphold ethical standards. Once a large number of documents from various categories such as sport, health, and business have been gathered, the attention moves to the document gathering procedure. Each document has at least one phrase and adds to a large dataset, which is essential for efficient clustering. The manual collecting approach assures purposeful curation of various documents, resulting in a comprehensive dataset that truly reflects the variegated character of the selected categories. Further following figure shows, a portion of the dataset that was collected.



Figure 12: Sample of Collected Dataset.

## User Interface

Following the document collecting phase, the focus shifts to developing an intuitive user interface. The interface uses Tkinter, a popular Python GUI toolkit, and displays a simple text box for users to enter documents for clustering. After receiving the document, the system quickly processes it and shows the expected cluster and matching category with useful labels. This interactive interface promotes seamless interaction, allowing users to easily receive cluster forecasts.

The following figure show the visual representation of the GUI, where there are four components, one is the text box the big black area which is provided to the user for the purpose of providing text for the purpose of predicting its cluster category. Following the text box are two labels that each show the result once the user clicks on the button named as “Submit”, to start the process of classification.

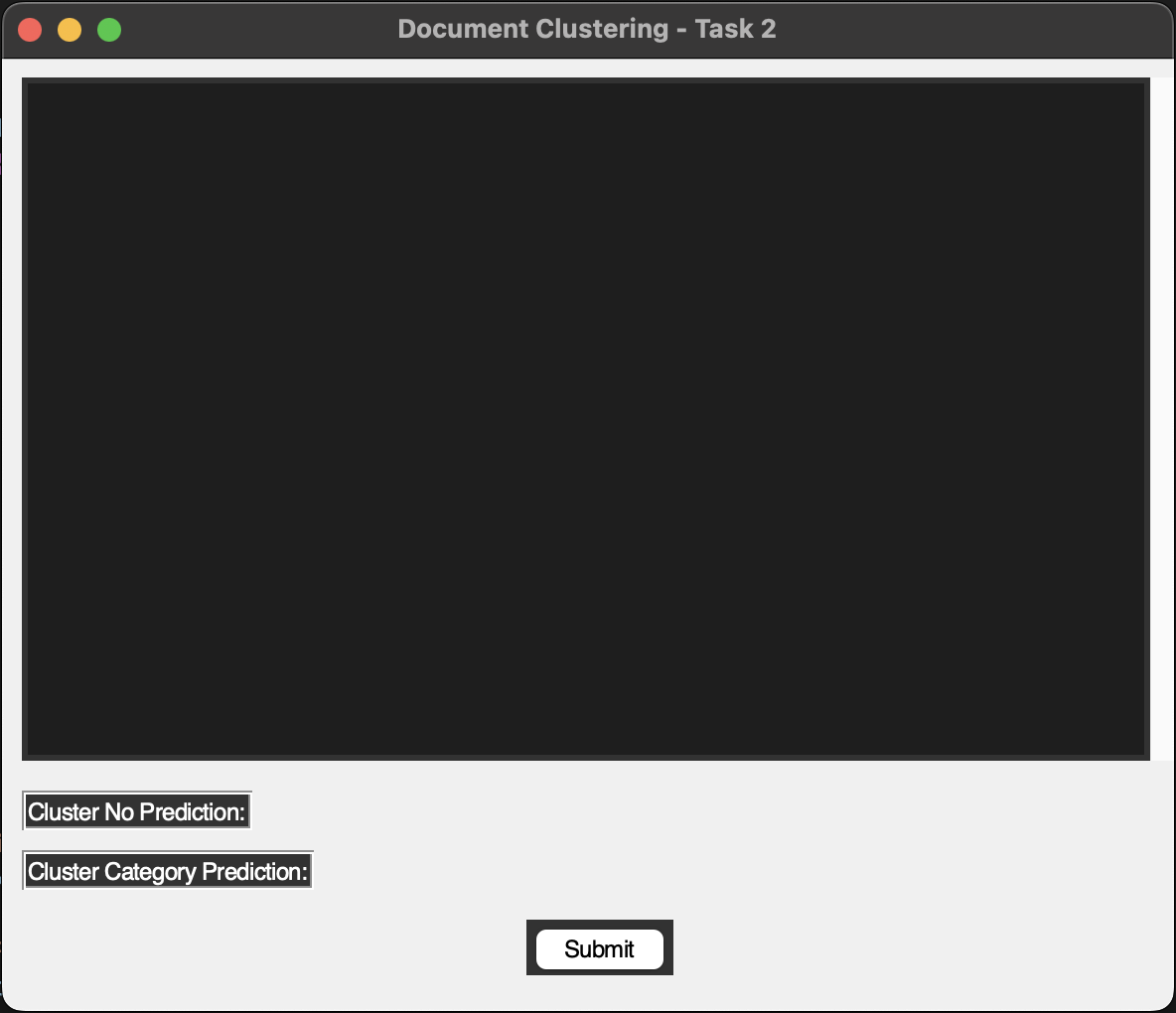


Figure 13: Tkinter GUI.

Further, in the GUI the user has to write down any sentence before clicking on the submit button or else it shows error message as the suggested by the following figure.

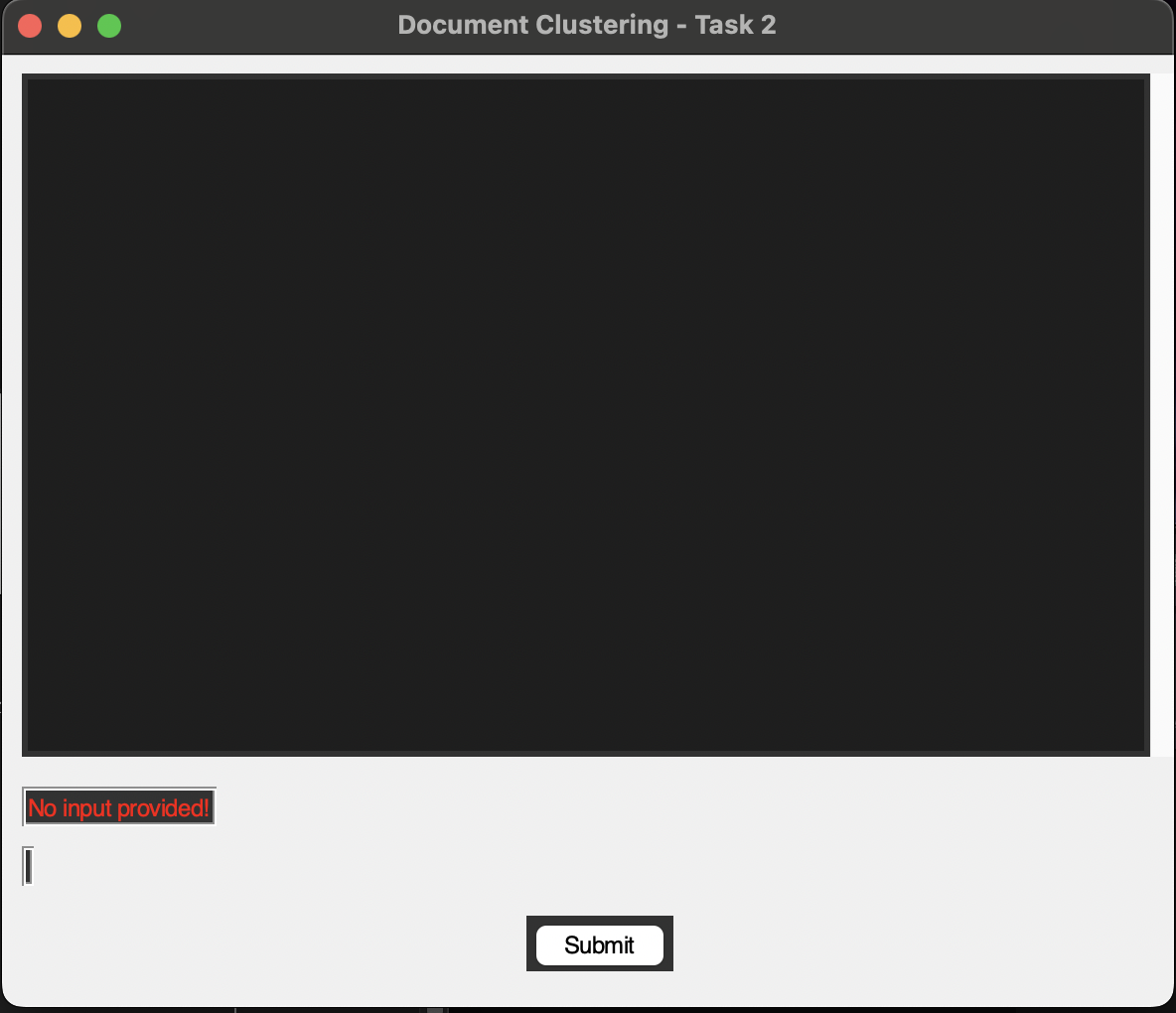


Figure 14: GUI Showing Error Message when No Input Provided.

Now, if we provide any sentence relating to any sports, health, or business-related news then the GUI will display the following output based on the input provided as shown in the following figure.

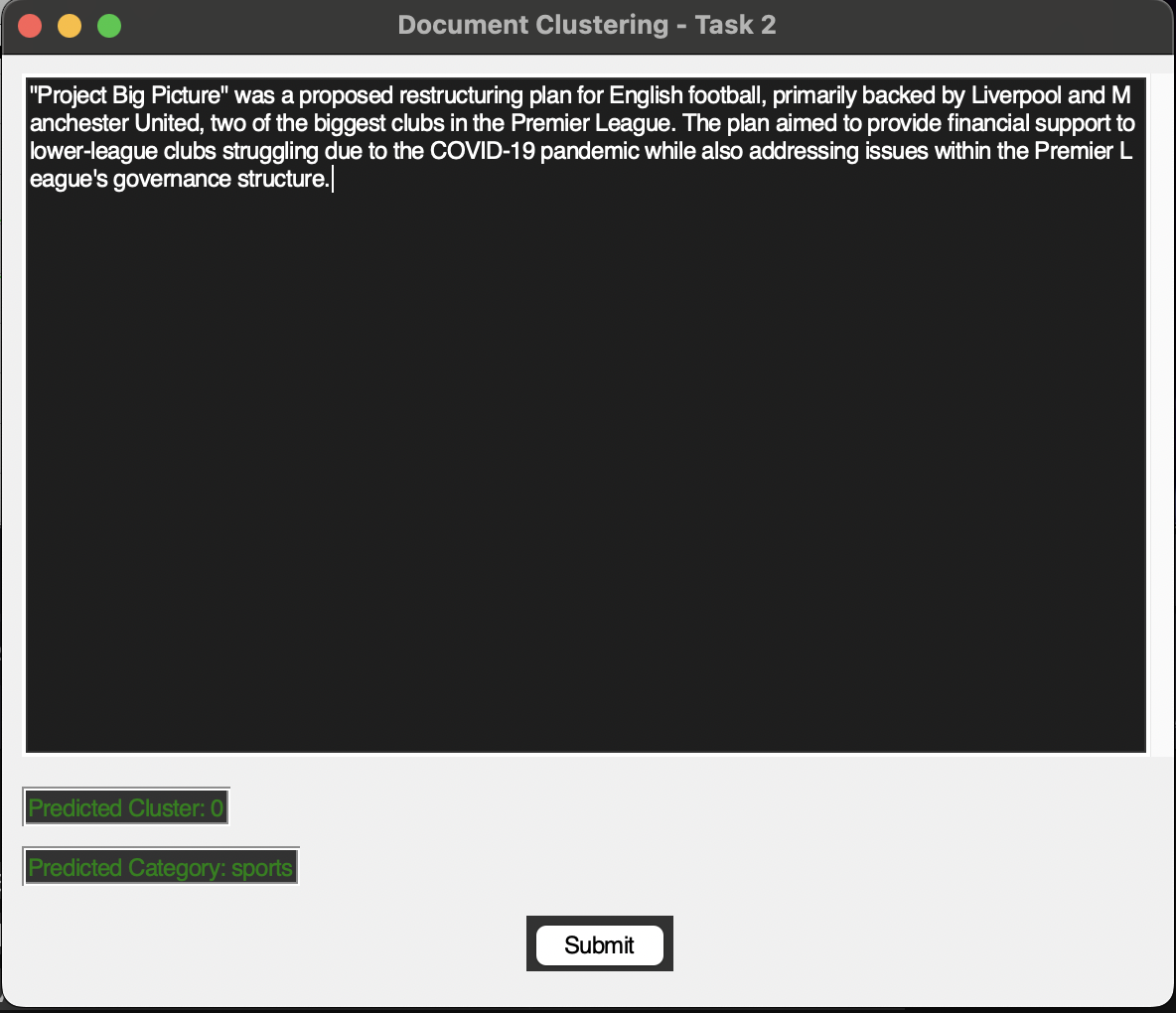


Figure 15: Successful Working of System.

## Assignment Process

The assignment process is at the heart of the system, and it consists of numerous essential processes. Initially, preparation measures are conducted to erase any missing data in critical columns like 'Title' or 'Excerpt.' The document data is next TF-IDF vectorized, which converts textual information into numerical vectors, which is an important step in the subsequent clustering process. The robust K-means clustering technique is used to classify documents into discrete groups depending on their content, allowing for efficient organization and categorization. Finally, the system uses the learned model to precisely forecast the cluster for freshly submitted documents, guaranteeing smooth integration into existing clusters.

To further understand this section, let us dive into the coding portion and understand how each of the process is well structed to meet the above expectation. Before starting the clustering process, make sure the dataset is clean and well-structured. In the code, missing values in critical columns such as 'Title' or 'Excerpt' are addressed using the 'dropna()' method, ensuring that only complete and relevant data is used for clustering. This phase helps to retain data integrity and avoid potential complications during further processing, as seen in the image below.

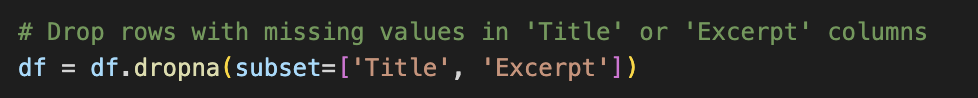


Figure 16: Preprocessing Dataset.

Following the pre-processing textual data, which is fundamentally unstructured, must be translated into a format that allows for mathematical processing. The TF-IDF (Term Frequency-Inverse Document Frequency) vectorization approach does this by converting textual data into numerical vectors. The '***TfidfVectorizer***' from the '***sklearn.feature\_extraction.text***' module is used for this purpose, which allows for the construction of TF-IDF matrices from document data, this action in shown in the following figure that illustrates a code snippet.

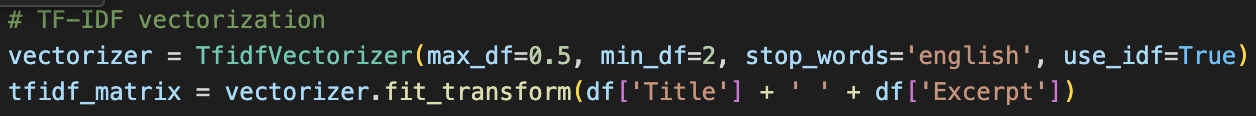


Figure 17: TF-IDF Vectorization.

Moreover, With the document data represented as numerical vectors, the K-means clustering technique is used to divide the documents into discrete groups based on content similarity. By iteratively updating cluster centroids and allocating data points to the closest centroid, K-means effectively organizes documents into clusters, allowing for efficient categorization and organizing.

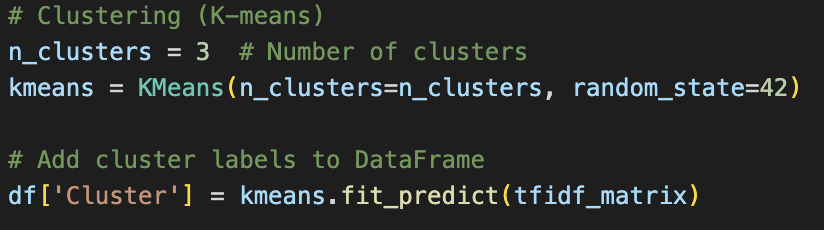


Figure 18: K-means Clustering.

Once the clustering model is trained, it can be utilized to predict the cluster for new documents. In the provided code, the '**predict\_cluster()**' function accepts a new document as input, transforms it into a numerical vector using the TF-IDF vectorizer, and then predicts the cluster using the trained K-means model. This prediction allows for seamless integration of new documents into existing clusters, ensuring the system's adaptability and scalability.

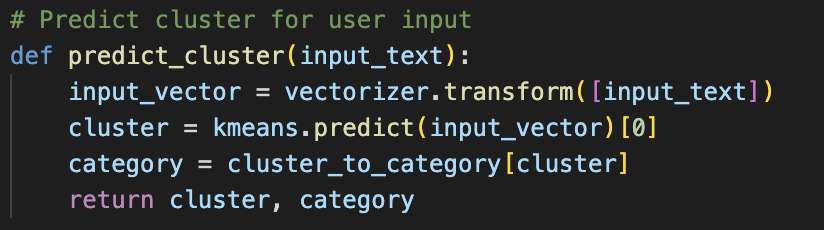


Figure 19: Cluster Prediction for New Documents.

## Robustness Analysis

A variety of input documents are evaluated to determine the system's robustness, those inputs are:

1. **Short Inputs**

Here, the following figure shows that a short version of inputs which is correctly predicted.

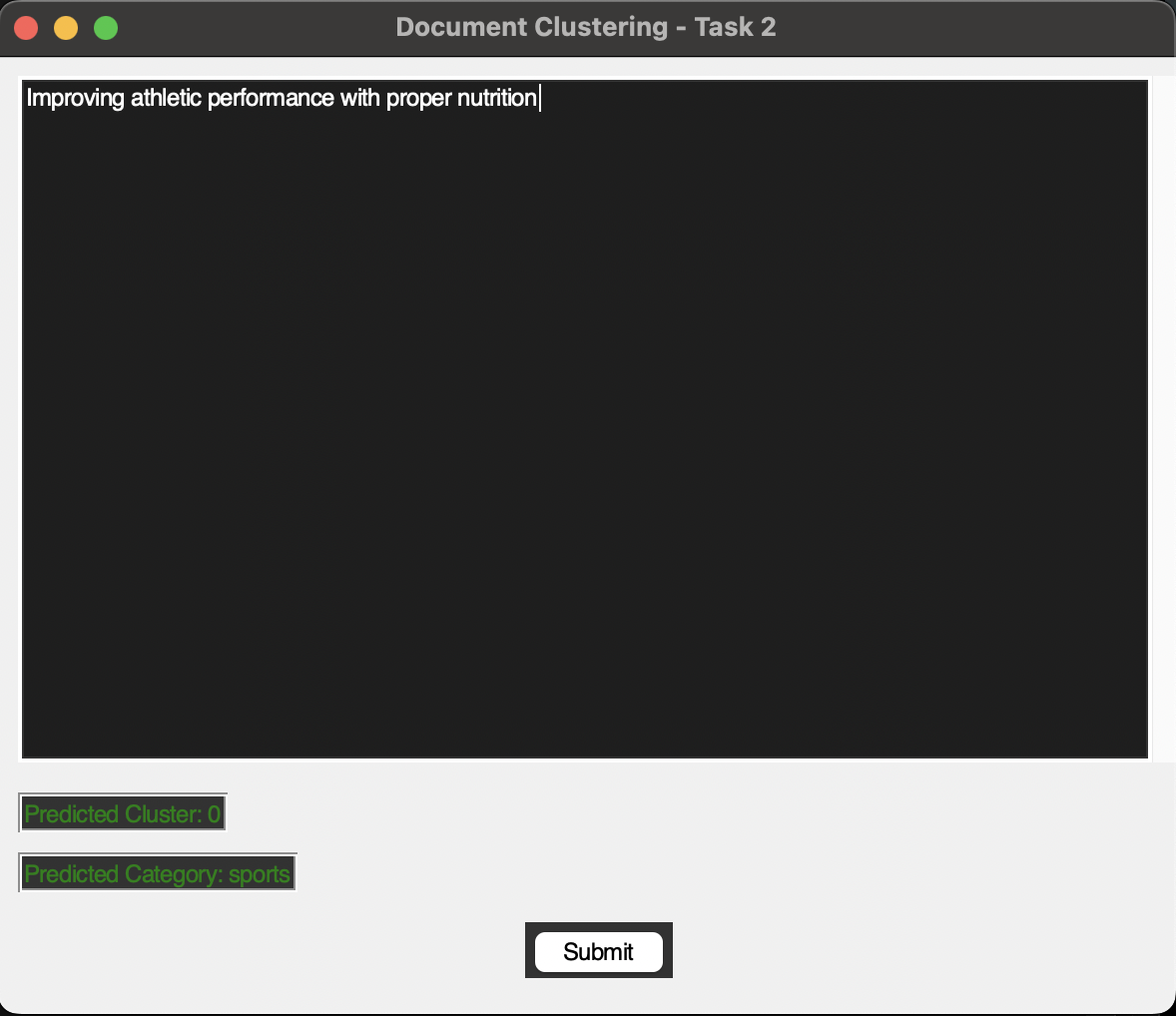


Figure 20: Short Input Test.

1. **Long Inputs**

Here, the following figure shows that a long version of inputs which is also correctly predicted



Figure 21: Long Format Input.

1. **Inputs without Stop-words**

Stop words include "the," "of," "and," "for," "overall," etc, hence in this part of the input, the emphasis is on not using such words that is shown in the following figure.

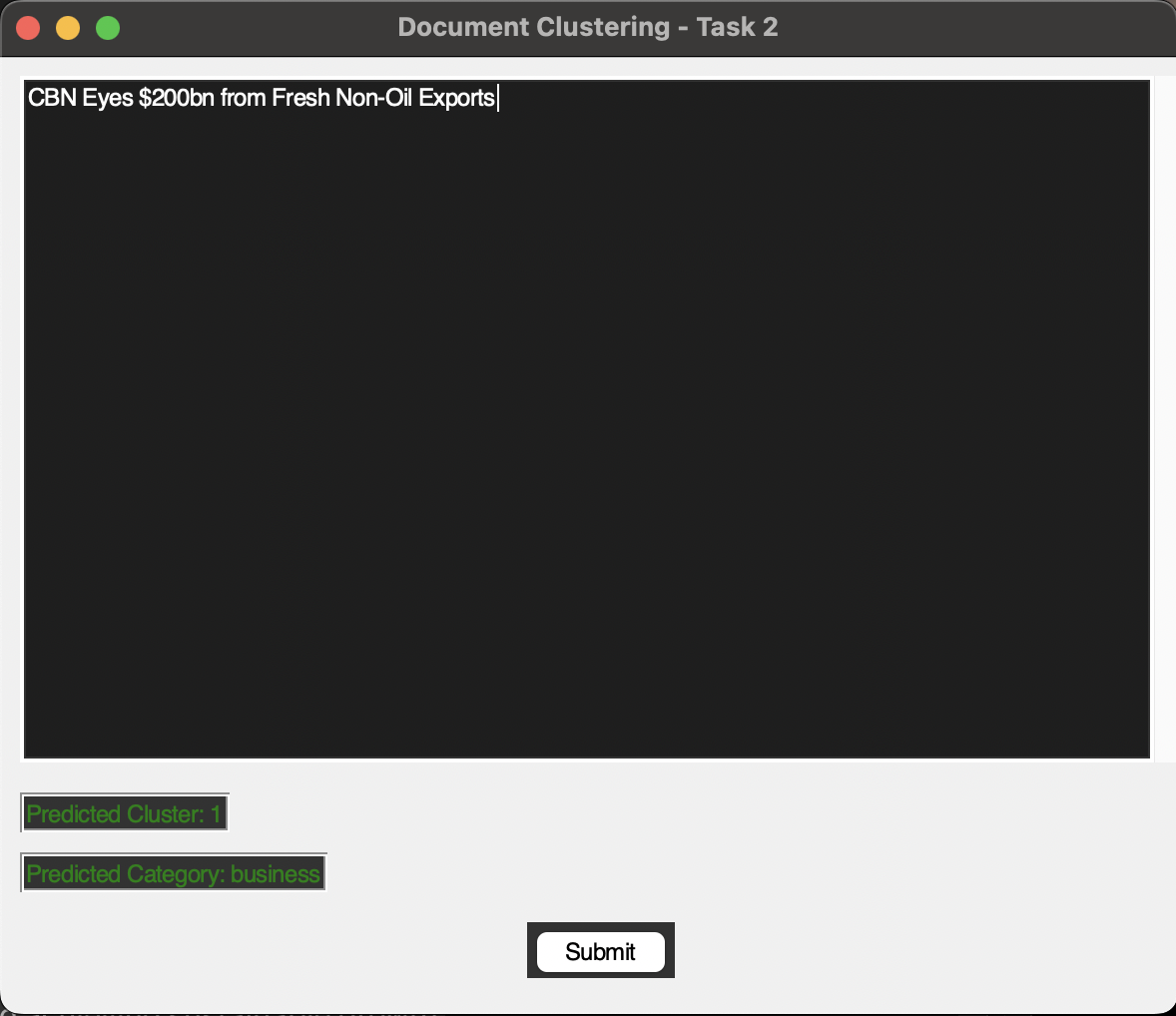


Figure 22: Inputs without Stop Words.

1. **Inputs with Stop-words**

Stop words make the phase even harder to understand any they don’t hold any singular meaning, so this is a tricky input, however the prediction is still on point as shown in the following figure.

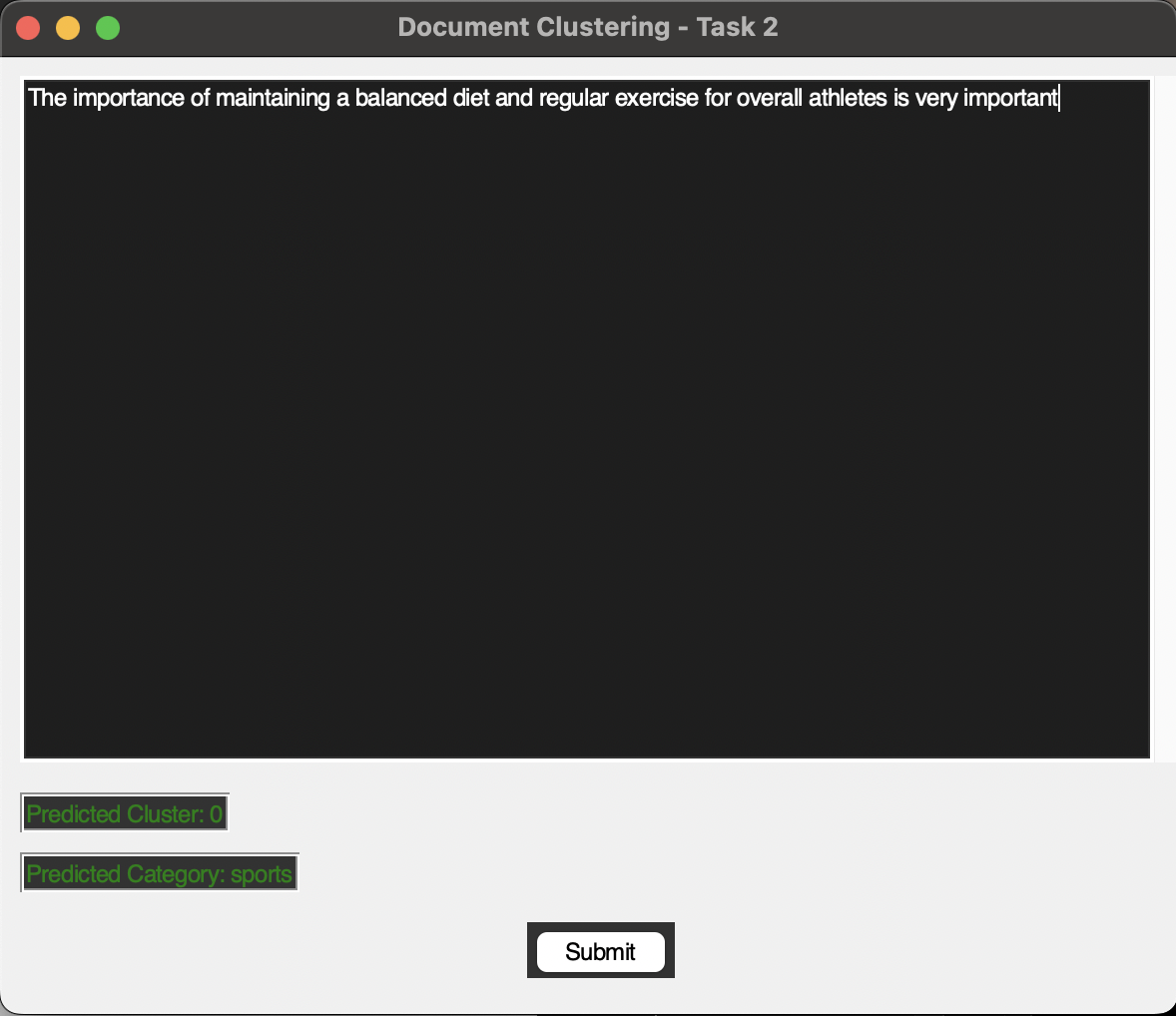


Figure 23: Inputs with Stop Words.

1. **Inputs form Different Topics**

Till now, we have analyzed for business related news and sports related new, now let us input something related to health, as shown in the following figure.

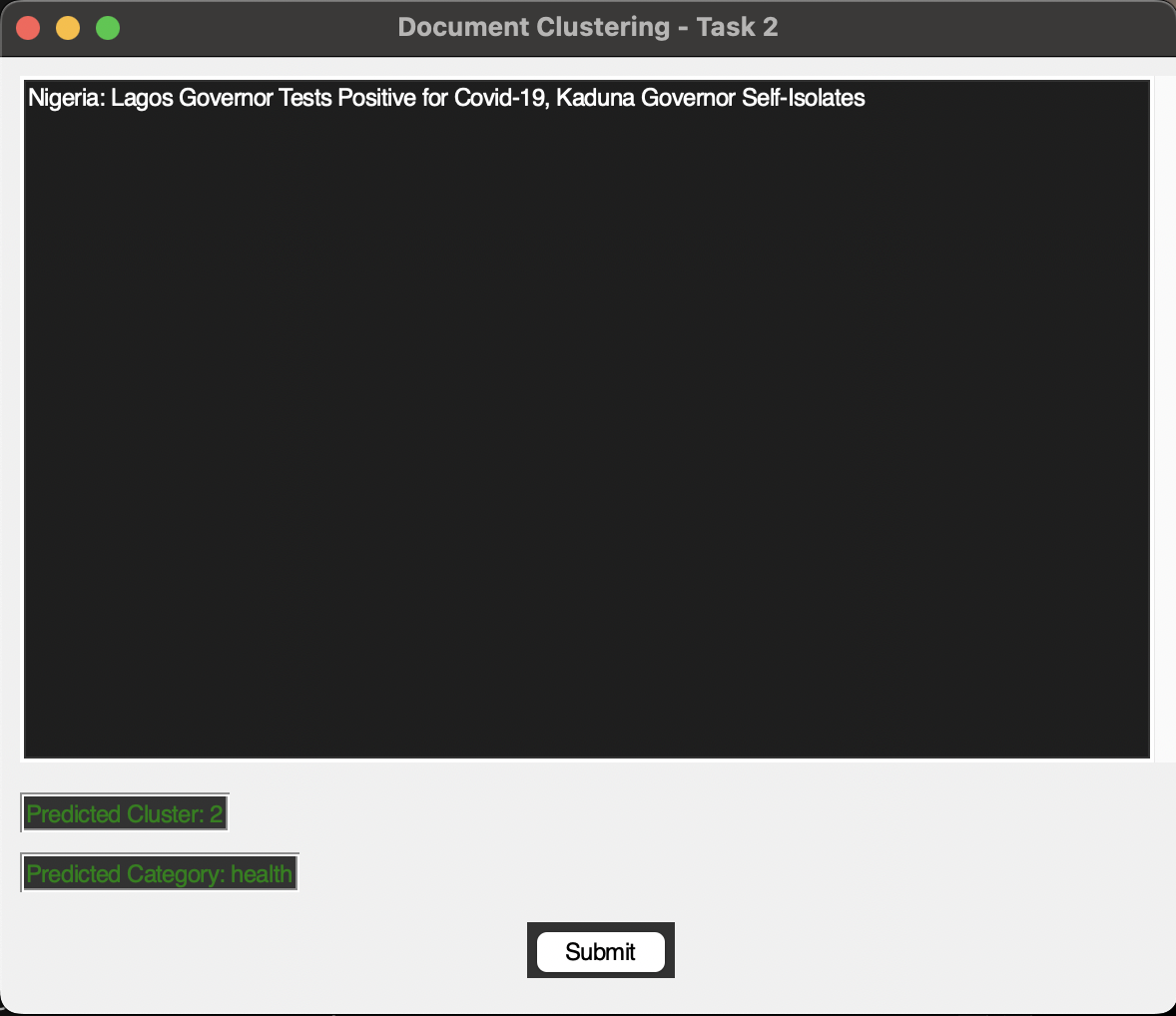


Figure 24: Health Realted Topic Input.

# Conclusion

In a nutshell, the implementation and the evaluation of the two separate system, which are the search engine and document clustering has been presented in this report. First the search engines provide a perfect platform to access the publication details from the website of Coventry University. That it also emphasizes on factors such as polite crawling and ensuring the correct output is obtained during the process of searching results. Furthermore, the document clustering offers the categorization of texts into three distinct categories based on content similarity. This system utilizes a very standard method for the process of clustering and predicting the result. Overall, both the system illustrates the capability of retrieving information, either by crawling and sorting data or by predicting clusters based on similarities with text.

# Reference

Long, B., & Chang, Y. (2014). *Relevance Ranking for Vertical Search Engines.* Maimi: Morgan Kaufmann.

Khder, M. A. (2021). *Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application.* Jordan: Al-Zaytoonah University of Jordan.