**A Project Report On**

**SUBJECT-BASED DOCUMENT CLASSIFICATION MODEL USING NATURAL LANGUAGE PROCESSING**

***Mini project submitted in partial fulfillment of the requirements for the***

***award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

**(2021-2025)**

**BY**

**B. S. PUNITH KUMARAN 21241A12H9**

**V. SRIRAM VARMA 21241A12K1**

**E. CLEMENT 21241A12E3**

***Under the Esteemed Guidance***

***of***

**Dr. N. RAJASEKHAR**

**Professor**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)**

**HYDERABAD**

**2024-25**



**CERTIFICATE**

This is to certify that it is a bonafide record of Mini Project work entitled **“SUBJECT-BASED DOCUMENT CLASSIFICATION MODEL USING NATURAL LANGUAGE PROCESSING”** done by **B. S. PUNITH KUMARAN(21241A12H9), V. SRIRAM VARMA (21241A12K1), E. CLEMENT(21241A12E3)** of **B.Tech** in the Department of Information of Technology, **Gokaraju Rangaraju Institute of Engineering and Technology** during the period 2021-2025 in the partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

**Dr. N. Rajasekhar Dr. Y J Nagendra Kumar**

Professor Head of the Department

(Internal Guide)

**(Project External)**

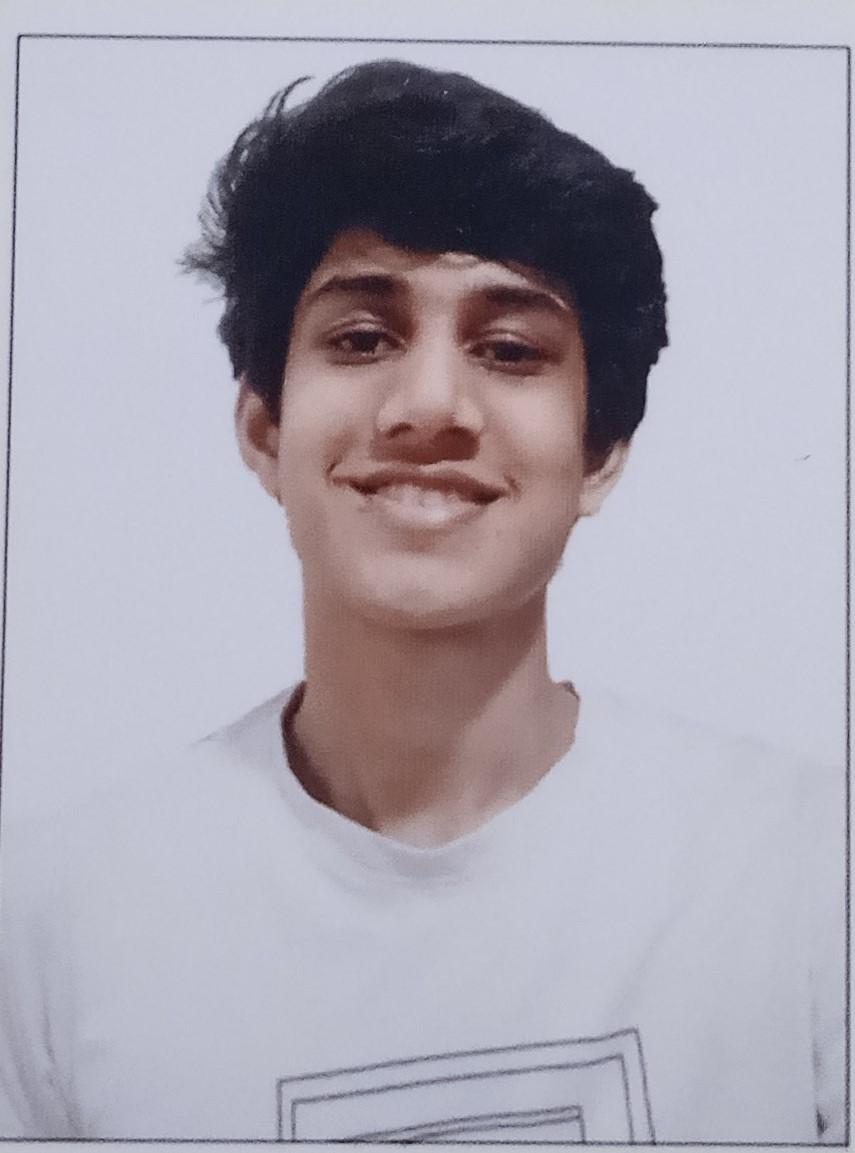
**ACKNOWLEDGEMENT**

We take immense pleasure in expressing gratitude to our Internal guide**, Dr. N. Rajasekhar, Professor, Dept of IT**, GRIET. We express our sincere thanks for his encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. Y J Nagendra Kumar,** HOD IT,our Project Coordinators **Dr. N. Rajasekhar** for their constant support during the project.

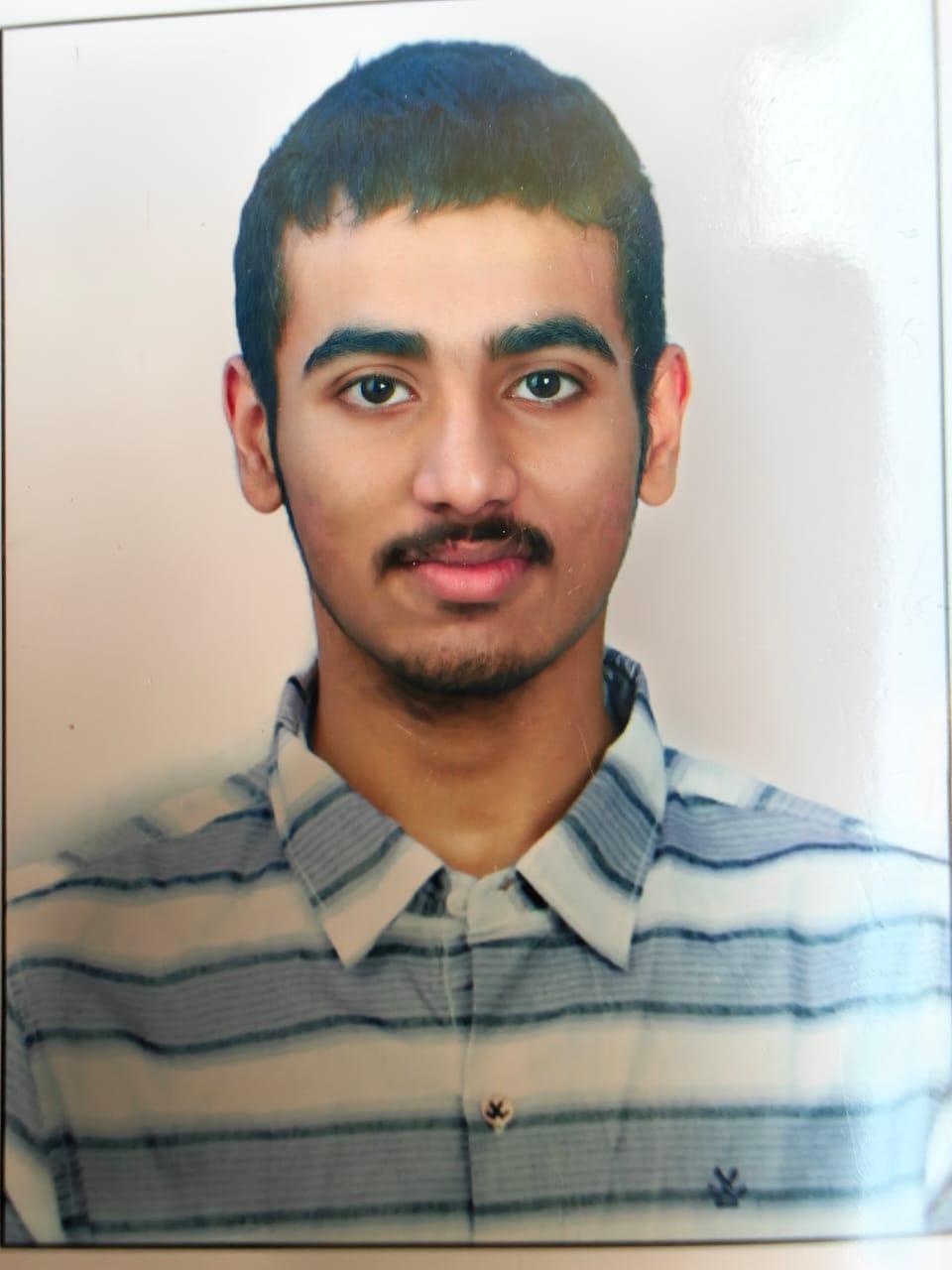
We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET**,** and **Dr. J. Praveen,** Principal, GRIET**,** for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

**Email**: punithkumarn0409@gmail.com **Email:** sriramvarma2125@gmail.com

**Contact No**: 9381964553 **Contact No**: 7993899180



**Email**: eadaclement@gmail.com

**Contact No.** 9948755668

**DECLARATION**

This is to certify that the mini-project entitled “**SUBJECT-BASED DOCUMENT CLASSIFICATION MODEL USING NATURAL LANGUAGE PROCESSING”** is a bonafide work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

**B. S. PUNITH KUMARAN 21241A12H9**

**V. SRIRAM VARMA 21241A12K1**

**E. CELMENT 21241A12E3**

**TABLE OF CONTENTS**

|  | **Name** | **Page no** |
| --- | --- | --- |
|  | **Certificates** | ii |
|  | **Contents** | v |
|  | **Abstract** | vii |
| **1** | **INTRODUCTION** | 1 |
| 1.1 | Introduction to project | 1 |
| 1.2 | Existing System | 2 |
| 1.3 | Proposed System | 2 |
| **2** | **REQUIREMENT ENGINEERING** | 3 |
| 2.1 | Hardware Requirements | 3 |
| 2.2 | Software Requirements | 3 |
| **3** | **LITERATURE SURVEY** | 4 |
| **4** | **TECHNOLOGY** | 7 |
| **5** | **DESIGN REQUIREMENT ENGINEERING** | 10 |
| 5.1 | UML Diagrams | 10 |
| 5.2 | Use-Case Diagram | 12 |
| 5.3 | Class Diagram | 13 |
| 5.4 | Activity Diagram | 14 |
| 5.5 | Deployment Diagram | 15 |
| 5.6 | Architecture | 16 |
| **6** | **IMPLEMENTATION** | 17 |
| **7** | **SOFTWARE TESTING** | 23 |
| 7.1 | Unit Testing | 23 |
| 7.2 | Integration Testing | 23 |
| 7.3 | Acceptance Testing | 24 |
| 7.4 | Testing on our system | 25 |
| **8** | **RESULTS** | 26 |
| **9** | **CONCLUSION AND FUTURE ENHANCEMENTS** | 27 |
| **10** | **BIBLIOGRAPHY** | 28 |

**11 LIST OF DIAGRAMS**

| **S No**  **1**  **2** | **Figure Name**  Applications of Python  Concepts of UML | | **Page no**  7  11 |
| --- | --- | --- | --- |
| **3** | Use Case Diagram | | 12 |
| **4** | Class Diagram | | 13 |
| **5** | Activity Diagram | | 14 |
| **6** | Deployment Diagram | | 15 |
| **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20** | Architecture  NLP Architecture  Extraction of Keywords  En\_core instance creation  Preprocessing file data  Removal of semantics  Extraction of Nouns  Importing the keywords  List of imported keywords  Similarity b/w topics and files  Scores of Similarity  Moving files to the location  Results  Final results | | 16  16  17  17  18  18  19  19  20  21  21  22  25  26 |
|  | |
|  | |
|  | |

**ABSTRACT**

This project explores an innovative approach to document classification by leveraging advanced natural language processing (NLP) techniques and machine learning algorithms. The primary objective is to develop a robust system that can automatically classify and organize large collections of documents based on their content. The core of the system employs sentence transformers for high-quality sentence embeddings, enhancing the contextual understanding of textual data. Utilizing the k-Nearest Neighbors (k-NN) algorithm, the project aims to achieve precise classification by identifying and grouping similar documents.

To streamline the document organization process, the project integrates several powerful NLP libraries including spaCy, NLTK, and RAKE. spaCy is employed for its efficient tokenization, named entity recognition (NER), and dependency parsing capabilities. NLTK (Natural Language Toolkit) is utilized for additional text preprocessing tasks such as stop-word removal, stemming, and lexical analysis. RAKE (Rapid Automatic Keyword Extraction) is incorporated to extract key phrases and significant terms that contribute to the document's thematic representation.

The project is implemented in Python, ensuring a versatile and scalable solution. The integration of these libraries within the Python ecosystem allows for a seamless workflow, from raw text processing to feature extraction and ultimately to document classification. By combining the strengths of sentence transformers, k-NN, and advanced NLP techniques, this project aims to deliver an effective and efficient document classification system capable of handling diverse and extensive document repositories. The system's performance and accuracy are evaluated through comprehensive testing on various datasets, demonstrating its potential for practical applications in fields such as information retrieval, content management, and digital archiving.

**Keywords:** Natural language processing, Machine learning, KNN, Tokenization, Pattern detection, Text Classification.

**Domain**: Machine learning, Document Classification.

**1. INTRODUCTION**

**1.1 Introduction to Project**

In the era of big data, the ability to efficiently organize and classify vast amounts of textual information is crucial for numerous applications, ranging from information retrieval to content management and digital archiving. Traditional document classification methods often struggle to cope with the complexity and volume of modern data, necessitating the development of more advanced and accurate techniques.

This project addresses this challenge by developing a sophisticated document classification system that leverages state-of-the-art natural language processing (NLP) techniques and machine learning algorithms. The primary aim is to create a robust system capable of automatically categorizing and organizing large collections of documents based on their content, thereby enhancing the efficiency and accuracy of information management.

At the heart of the system are sentence transformers, which generate high-quality sentence embeddings to capture the nuanced contextual relationships within the text. These embeddings serve as the foundation for the classification process, enabling a more profound understanding of the document content. The k-Nearest Neighbors (k-NN) algorithm is employed to classify the documents, capitalizing on its simplicity and effectiveness in identifying and grouping similar documents based on their embeddings.

To further enhance the document processing pipeline, the project integrates several powerful NLP libraries. spaCy is used for its efficient tokenization, named entity recognition (NER), and dependency parsing capabilities, which are essential for understanding the structure and entities within the text. NLTK (Natural Language Toolkit) provides additional text preprocessing tools, including stop-word removal, stemming, and lexical analysis, which are crucial for cleaning and preparing the data. RAKE (Rapid Automatic Keyword Extraction) is utilized to extract key phrases and significant terms, facilitating the identification of the document's main themes.

Implemented in Python, this project takes advantage of the language's versatility and the rich ecosystem of libraries available for data processing and machine learning. The integration of sentence transformers, k-NN, and advanced NLP techniques within the Python framework allows for a seamless and scalable solution to document classification.

This introduction sets the stage for a comprehensive exploration of the system's design, implementation, and evaluation. By combining cutting-edge NLP methods with machine learning algorithms, the project aims to deliver a highly effective and efficient document classification system, capable of meeting the demands of modern data management challenges.

**1.2 Existing System**

The existing models mainly leverage

1. TF-IDF and SVM Classifiers
2. Naive Bayes Classifiers
3. Latent Dirichlet Allocation (LDA)
4. Convolutional Neural Networks (CNNs)

**1.3 Proposed System**

In order to improve accuracy, along with Convolutional Neural Network, tokenization and similarity analysis using sentence transformers can be employed as a backbone. By experimenting different types of input datasets that include notes, documents, articles, etc., the model achieves less loss and more accuracy.

**2. REQUIREMENT ENGINEERING**

**2.1 Hardware Requirements**

* Processor – i5 and above (64-bit OS).
* Memory – 4GB RAM (Higher specs are recommended for high performance)

* Input devices – Keyboard, Mouse

**2.2 Software Requirements**

* Anaconda Navigator/Jupiter Notebook
* Python
* Python Libraries

**1.** pandas

**2.**numpy

**3.**spaCy

**4.**nltk  
 **5.**rake

**3. LITERATURE SURVEY**

Naïve Bayes can also be used to categorize documents where volumes of information can hardly be put in a respective category or class. This kind of method involves the application of Naïve Bayes formula when testing the probability that a particular text belongs to a particular category based on the occurrence of the word. In other words, it utilizes a model of Naive Bayes mathematical approach to determine the probability of occurrence of a particular category based on the parameters of the document. it has been widely used for text classification, classification applications such as spam Filters, sentiment analysis, and topic modeling. In this method, the algorithm is prepared using a labeled data set where a collection of documents is associated with their categories or classes Once prepared, the algorithm can be used to classify new documents into the right categories or classes. Just like the case with other methods of classification, the degree of accuracy of the classification will depend on the relevant features used and the extent to which they can be applied across the documents. Among the classifiers the Naive Bayes can be utilized in the process of smart document categorization as it does not require a large amount of training data and is very easy to apply. Therefore, based on the naive Bayes algorithm, the automated documents classification can be viewed as an efficient and quite reliable procedure for managing and sorting the big volumes of textual data.[1]

In summary, it could be proposed that RVL-CDIP should be considered for document image analysis may be akin to what ImageNet is to computer vision. It is probably the largest such matrix reported in the healthcare literature, if not the largest in the literature generally. The total number of Document Images in the data set is 400000. There are a lot of outliers, and there are high variations in the composition of the document classes in the dataset. Original, the dataset volume is ~100GB and containing the total of 16 classes of document types, each having 25000 samples. It covers classes like email , resume and invoice etc. Was able to come up with a model, with over 93% accuracy, which is above the benchmark score of 92%.[2]

Classification technique is the most important in case of base machine learning method with reference to supervised and semi supervised learning techniques. There are have many classification algorithms which has been introduced already for existing system. Class-label classification is one of the most fundamental problems in the field of machine learning; it allows one to assign a subset of candidate without label to an object. Different techniques for classification of various model for documents by short text, metadata, heading levels are these existing techniques which are discussed in the literature survey. Sometimes reading and processing the entire dataset may be time-consuming for classification, and therefore, it may prolong the time efficiency of the whole system. Our approach to improve the novel method of document classification involved the use of Natural Language Processing and machine learning with the aid of deep learning algorithms. In this work system has several attractive properties: the tools above first it extract some metadata from entire abstract section and built the training set. So, it is an optimization algorithm that handles all the document process when they are complete, all of them. RNN has now been employed to sort the individual object depending on the weights it has. And it gives final class label of the entire test datasets. Comparing with the classical machine learning algorithms, the experimental analysis shows that the proposed system of analysis provides a higher accuracy of data classification along with the least time complexity.[3]

The construction industries could easily benefit from automation of document handling; this is best undertaken in accordance to a classification process, and this step is precisely that. When it comes to machine learning, a vast number of studies have been made to extend the number of algorithms and techniques necessary to accomplish machine learning, but there are many applications and areas where those techniques would be useful that has been done. Within this study I assessed to how a certain extent the constructed machine learning algorithm multinomial Naïve-Bayes would be in a position to categorize 1427 documents that are grouped into 19 classes, all of which belongs to construction project. The experiment shown in the perturbation was 92% accurate. Description of their % accuracy was at 7% and the paper also draws a handle on how the accuracy can be enhanced. But data extraction became an issue and yielded only up to 6–66% of the documents for testing the classifier.[4]

Abstract classification is thus an input to an abstract which has a corresponding output which is the class of the paper for instance Physics, math. This will be done using artificial neural networks and machine learning algorithms to increase the models understanding on the classification of papers through the abstracts. while it may be challenging for a human to read each document and understand which classes can be suitable for it, these abstracts need to be reviewed one by one in order to make a decision about the classes for each of them, and this will take an unreasonable amount of time to have the results, even if the accuracy of the human decision is high. A significant number of papers possibly require categorization and shelving in libraries as well as other learning institutions. And because there may be hundreds of these articles coming in daily, a new approach must be developed toward their classification and organization that is both sound and efficient. It achieves this by an integration of a machine learning model that can classify documents into grouping papers into “categories” depending on the relevant topics and the results are depicted to the end user via a web base application. [5]

In order to tool the classification of documents automatically, the documents have to be put into a format that is interpretable by the classifier- a machine learning algorithm. The procedures for feature vector creation and the possibilities to use document to create feature vectors which will later help to classify documents are described in the report. For more specifically the above project focuses at classifying the results from three feature vectors namely- Binary, Count and TfIdf vectors. To determine which of the three mentioned feature vectors performs better each one of the feature vectors was tested on a 20-newsgroup dataset and converted into the three different forms. SUBJECT AREA: Artifices of Natural Language Processing for Information Retrieval Proposal For each feature vector representation option, the Naïve Bayes classifier was trained and the generated classifier was tested on the test documents. From the experiments, we observed that TfIdf did better than the Count vectorizer by a 4% and Binary vectorizer by a 6% if the stop words are removed. Without applying stop words remove TfIdf resulted +6% better accuracy than Binary vectorizer and 11% better than Count vectorizer. Furthermore, it has been also identified that when Count vectorizer is compared with Binary vectorizer, Count vectorizer yields a better result if there is a 2% removal of stop words but if the stop words are not removed then Binary vectorizer outperforms Count vectorizer by 5%. Consequently, we can determine that TfIdf is the approach for document representation and classification, which should be preferred over others.[6]

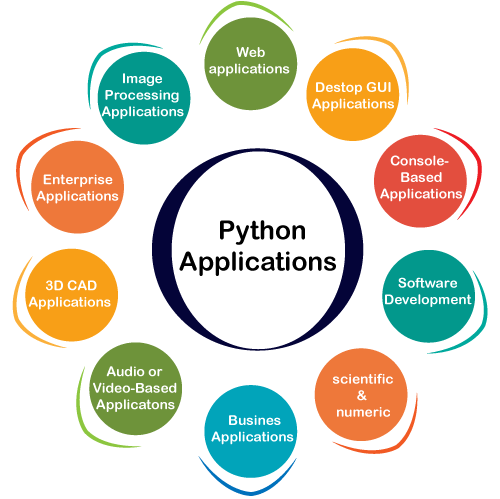
**4. TECHNOLOGY**

**4.1 ABOUT PYTHON**

Python’s environment has evolved over time, and it is becoming increasingly capable of statistical analysis. It strikes a good balance between scale and class. Python plays a premium on efficiency and readability. It has a design that emphasizes program readability and a simple syntax that is beginner-friendly and also lets programmers express the codes in fewer lines notably using with indentation. The speciality of this excessive degree language are functions of dynamic system, automatic-memory management system.

**4.2 APPLICATIONS OF PYTHON**

Python is used in many application domains. It makes its presence in every emerging field. It is the fastest-growing programming language and may be used to create any type of application.



*[Fig – 1]**Applications of python*

It is used in various fields:

* Web Applications. We can use Python to develop web applications. ...
* Desktop GUI Applications. ...
* Console-based Application. ...
* Software Development. ...
* Scientific and Numeric. ...
* Business Applications. ...
* Audio or Video-based Applications. ...
* 3D CAD Applications.

**4.3 PYTHON IS WIDELY USED IN DATA SCIENCE**

We use python data science in flexible and open source language.It gives functionality which deal with mathematics and scientific function.Most probably we use because of simple syntax and it gives huge libraries.It also consumes less time to code.

The major python libraries used in Data Science are as follows

**4.3.1 PANDAS**

A library in python that's used for statistics analyzing, cleaning, exploring and manipulating. generally, dataset incorporates many beneficial and useless statistics. Pandas cause them to readable and relevant.

**4.3.2 NUMPY**

A library in python that's used for statistics reading, cleaning, exploring and manipulating. commonly dataset carries many useful and vain data. Pandas make them readable and relevant.

**4.3.3 SPACY**

Spacy is a robust natural language processing (NLP) library for Python, designed to handle large-scale text processing with ease. It provides efficient tokenization, named entity recognition (NER), part-of-speech tagging, and dependency parsing capabilities. Developers harness its powerful APIs to build sophisticated applications for sentiment analysis, text classification, and more. With Spacy, Python enthusiasts delve into the intricacies of language data effortlessly, advancing projects in AI and computational linguistics.

**4.3.4 RAKE**

RAKE (Rapid Automatic Keyword Extraction) in Python is a powerful library for extracting keywords from text based on their frequency and co-occurrence with other terms. It employs algorithms to identify key phrases that encapsulate the main themes of a document or corpus. RAKE simplifies the task of prioritizing essential content for tasks such as document summarization, information retrieval, and content categorization.

**4.3.5 NLTK**

NLTK (Natural Language Toolkit) in Python is a comprehensive library for working with human language data. It offers tools for tokenization, stemming, tagging, parsing, and classifying text, making it indispensable for NLP tasks. NLTK's extensive corpus collections and language processing algorithms support research, education, and practical applications in text analysis and machine learning.

**4.4 Dataset Description**

A dataset has been taken from CBSE 11th and 12th grade NCERT Syllabus. The entire textbook was subdivided into several documents that include one chapter at once. Each subject under test i.e. Mathematics, Physics, and Chemistry consists of 20 documents each. So in total, the Dataset involves 1200 pages worth of data in docx format.

Input size: 142.8MB

Purpose of dataset:

1. To extract classifying keywords for each topic.
2. To test the software efficiency.

**5.DESIGN REQUIREMENT ENGINEERING**

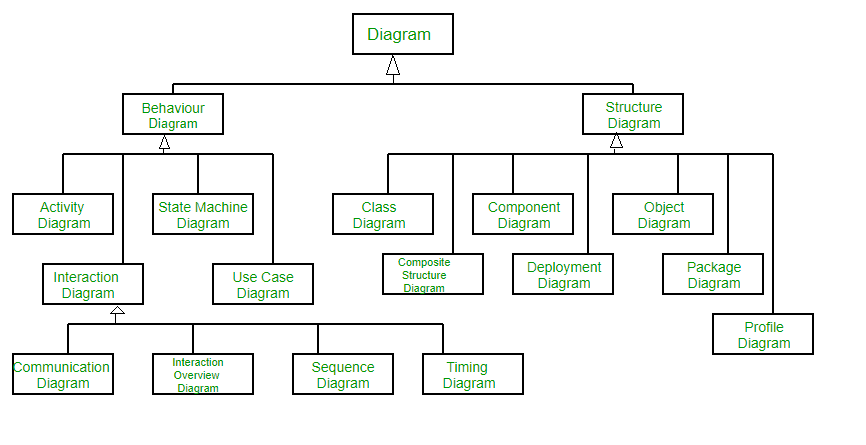
**Concept of uml:**

The aim of those diagrams, which are based on UML, is to visually represent the machine as well as its primary players, roles, moves, objects, or training, with the intention of better understanding, manipulating, preserving, or filing statistics about the machine.

**UML DIAGRAMS:**

The Unified Modelling Language (UML) is a language used to create models for various purposes. Its main objective is to provide a standard way of representing the structure of a system visually, much like blueprints used in other fields of engineering. In the case of complex applications, the involvement of multiple teams necessitates clear and straightforward communication between them. Businesspeople may not understand code, and this is where UML comes in. It helps to communicate the essential requirements, features, and procedures of the system to non-programmers. By visualizing processes, user interactions, and the static structure of the system, teams can save time in the long run.

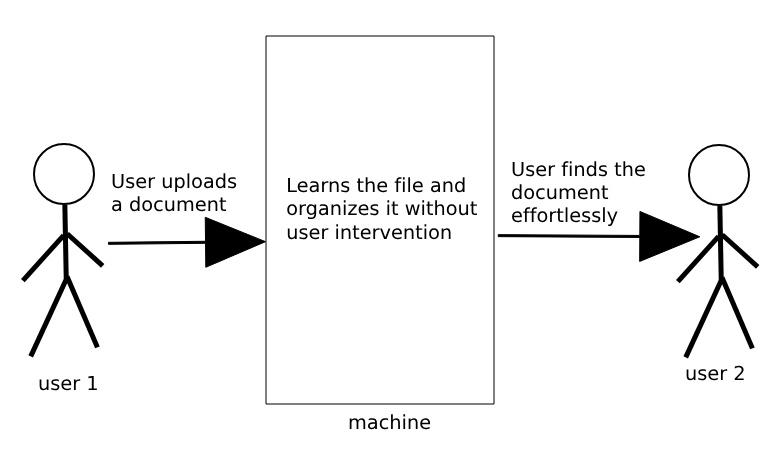
UML is linked with object-oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

****

*[Fig – 2]**Concepts of UML*

**5.1 Use case Diagram:**

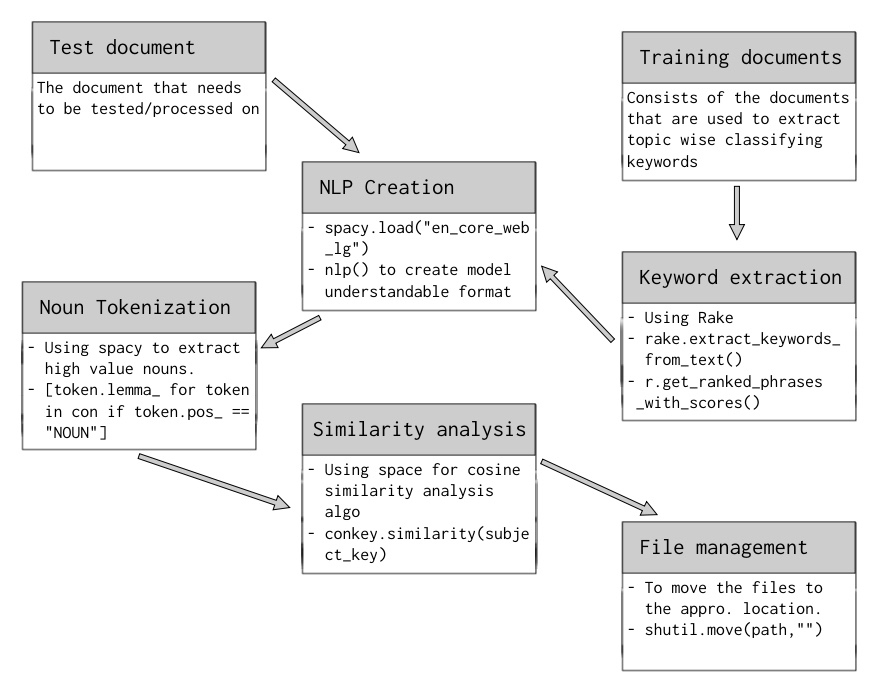
A use case diagram is a type of behavioral diagram that is a graphical explanation of the functionalities offered by the system in relation to the participants, their goals, and any dependencies between these cases.



*[Fig – 3]**Use case Diagram*

**5.2 Class Diagram:**

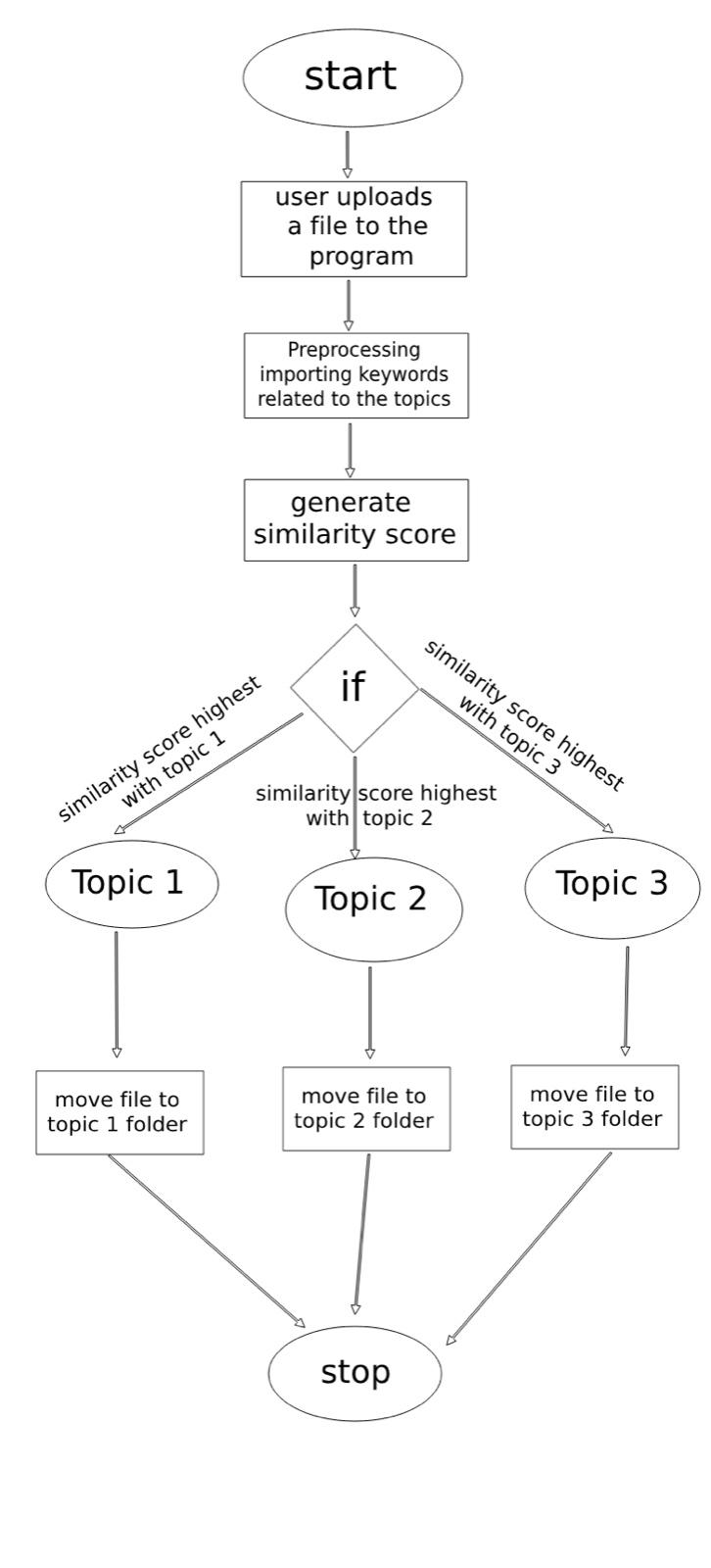
A class diagram is a static type of structural diagram that still depicts the format of a machine by means of illustrating the hyperlinks among the machine's lessons, attributes, operations, and instructions.

****

*[Fig – 4]**Class diagram*

**5.3 Activity diagram:**

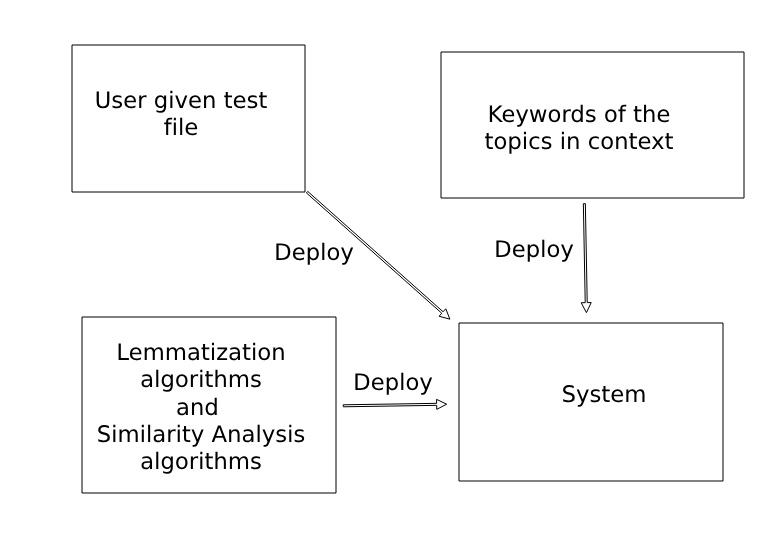
This diagram is a more complex version of a flow chart that depicts the flow of information from one activity to the next. It describes the coordination of activities in order to offer a service at various levels of abstraction.



*[Fig – 5]**Activity diagram*

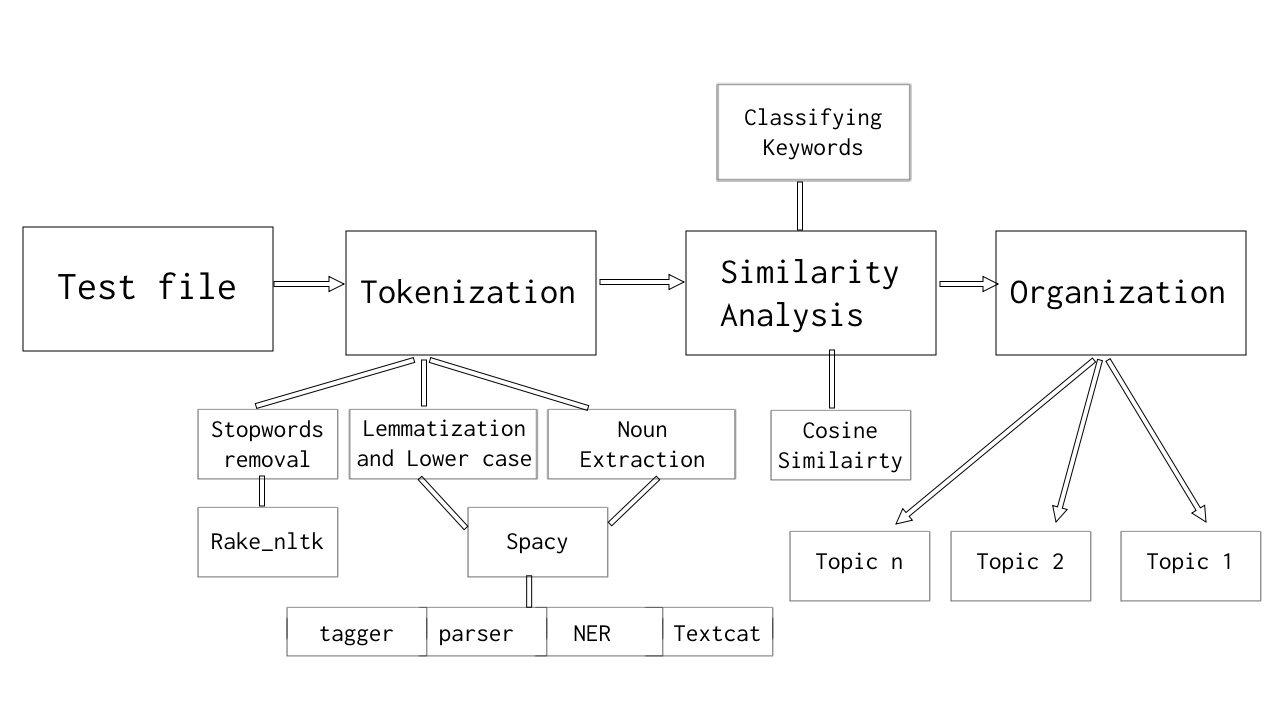
**5.4 Deployment Diagram:**

The deployment diagram of our machine in order to define distinct states of an object for the duration of its lifetime. It usually suggests how the kingdom of an object adjustments in its lifetime. It additionally logical view of functionality within the version/venture which includes paths, loops, situations and so on it.

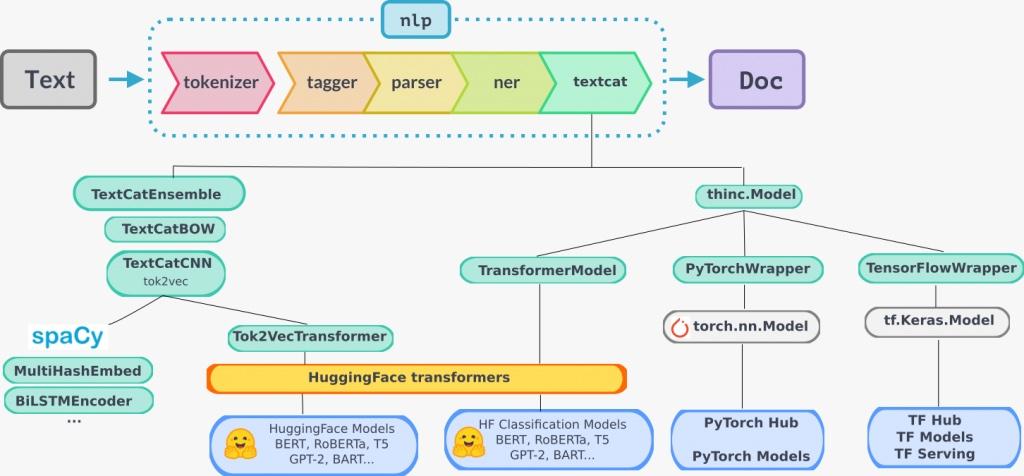


*[Fig – 6]**Deployment diagram*

**5.5 Architecture**



*[Fig – 7]**Architecture*

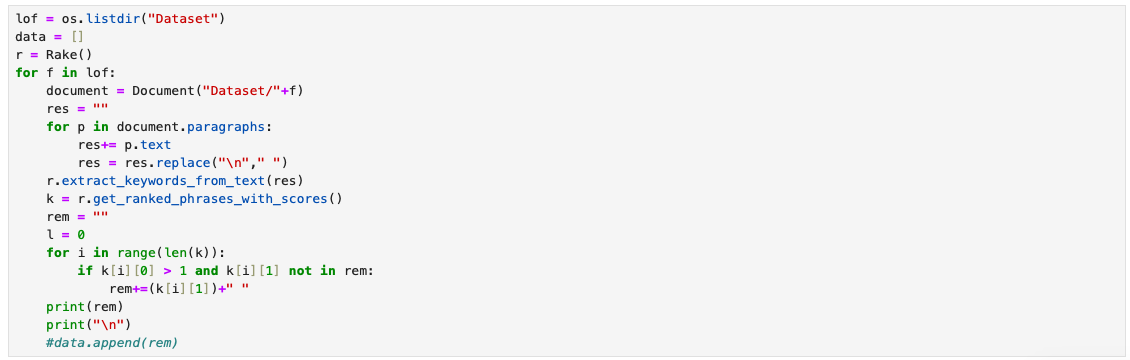
****

*[Fig – 8]**NLP Architecture*

**6. IMPLEMENTATION**

**Extraction of keywords for each topic:**

The below code snippet is used to extract a large set of keywords that are used to define each topic from a large dataset. To optimize the performance of the model, each keyword is returned along with a rank which makes it easier for extraction of most important keywords from the outliers.



*[Fig – 9] - Extraction of keywords from dataset*

**Creation of en\_core instance:**

The below code snippet is to employ a spacy tokenizing model that is further used to tokenize and leverage functions of spacy later in the code in a variable named “nlp”.

/Users/tanvi/Desktop/Screenshot 2024-06-18 at 11.12.05 PM.png

*[Fig – 10] - en\_core instance creation*

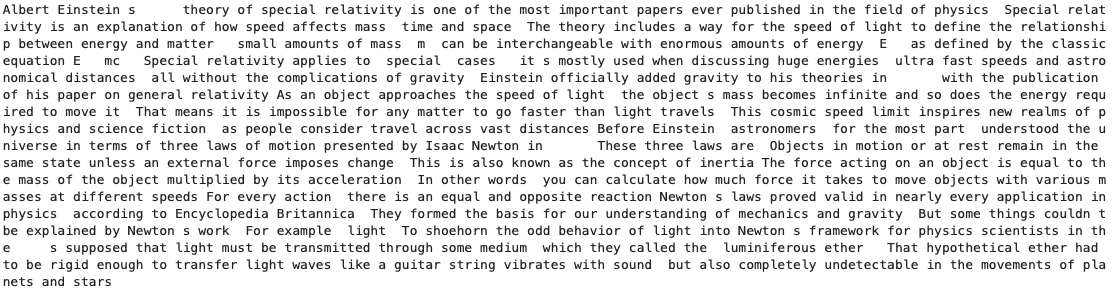
**Preprocessing of the test file:**

The below code snippet is to preprocess the given input file that needs to be acted on. The pre processing part involves:

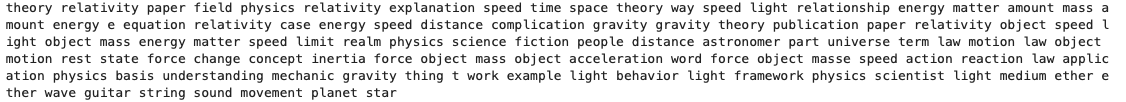
* Extraction of the entire text from the given file into a string.
* Removal of semantics like hyphens and question marks and other punctuation.
* Extraction of important nouns while avoiding other parts of grammar to get the most efficient keywords to work with.
* Tokenization of thus extracted keywords into the spacy’s encore nlp understandable format to perform KNN/similarity test.



*[Fig – 11] - Preprocessing of file data*



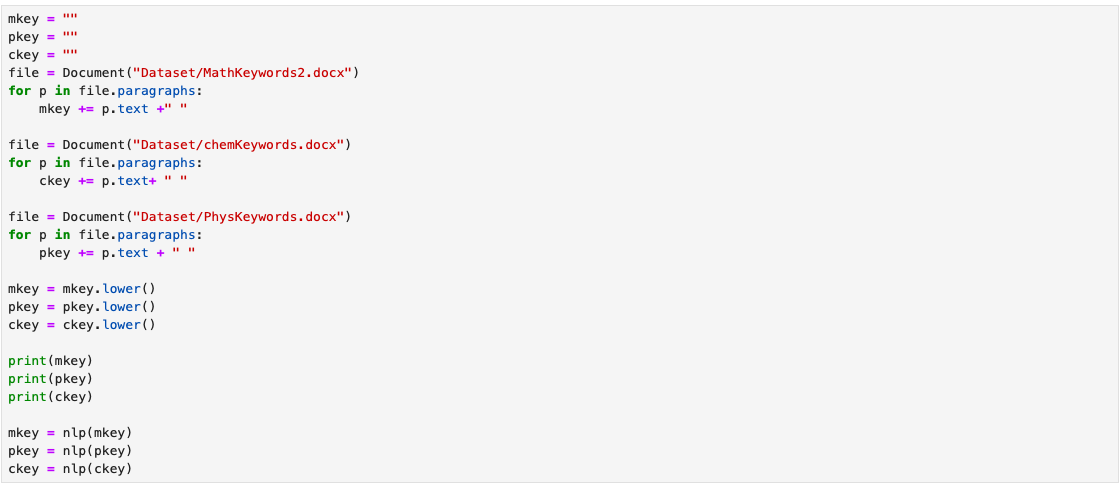
*[Fig – 12]- Partially preprocessed data after removal of semantics*



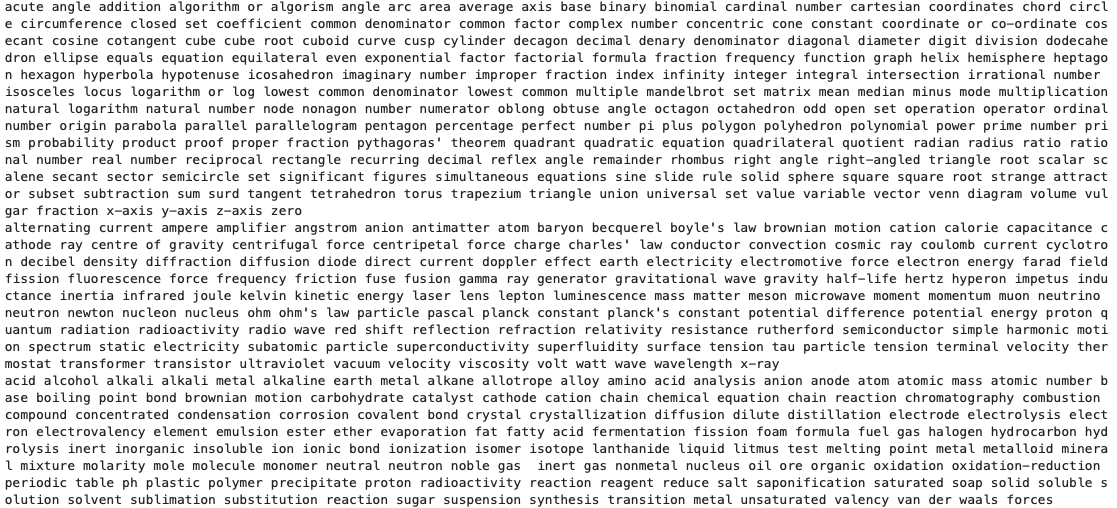
*[Fig – 13] - Fully preprocessed and extracted nouns from the given file*

**Importing the keywords of each topic:**

The below code snippet is used to import the previously extracted keywords from the dataset, and tokenize it into spacy’s encore nlp understandable format.



*[Fig – 14] - Importing the classifying keywords*

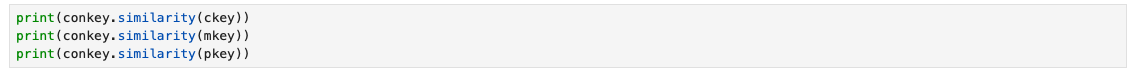


*[Fig – 15] - The list of imported keywords for classifying*

**Checking for similarity with the classifying keywords:**

The below code snippet is used to get a score how similar the given input file text keywords are to each of the topic’s keywords.

We get a float value between 0 and 1 where 1 represents the highest similarity while 0 represents the least.



*[Fig – 16] - Finding similarity between the topics and given file*

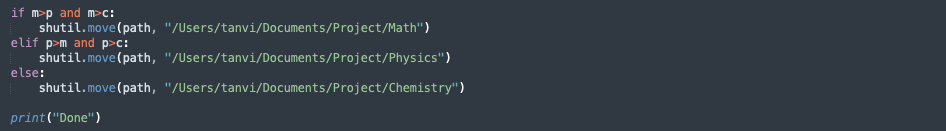
Screenshot%202024-06-18%20at%2011.58.23%20PM.png

*[Fig – 17] - Example of the returned scores of simalarity*

From the returned scores, we conclude that the one with the highest score is the topic the given input file belongs to.

**Moving the file to the appropriate location:**

The below code snippet is used to move the file to the appropriate location according to the returned similarity score.



*[Fig – 18] - Moving the file to the appropriate location*

**7. SOFTWARE TESTING**

Software testing is a testing that is done before actual software is completely executed. The main objective for doing the software testing is the requirements of the expected output is free from errors and defects.

**7.1 Unit Testing:**

Unit testing is the first stage to test a module by testing each individual unit the module. Each module or method of a procedure is tested to get the expected output. It helps to fix the defects and errors the module of each unit.

Unit testing can be used to test the many parts or operations that make up the document classification algorithm in the context of document classification utilizing topic-wise generated keywords. The following are some instances of unit testing in this situation:

To make sure that the imported topic tokens and similarity ananlysis are properly loaded, normalized, and ready for additional processing, test the document loading and preprocessing functions. Check that preprocessing operations like tokenization, keyword extraction, noun extraction, and removal of stopwords.

Implement and test the evaluation metrics for the document classification algorithm to determine how well it performs. Test functions that compute precision, recall, F1 score, or intersection over union (IoU) for accuracy in measuring the performance of the algorithm.

**7.2 Integration Testing:**

After unit testing, integration testing is carried out. As a result, the output of unit testing serves as the input for integrated testing. The functional requirements are taken as input. Individual units of code in a module are gathered or integrated for testing in this method.

The interaction and integration between the various parts or modules of the document classification system and integration during the integration testing for document classification utilising similarity analysis. This includes validating the input and output mechanisms, testing the integration with evaluation and performance metrics and ensuring the smooth integration of preprocessing and feature extraction components.

It also includes ensuring the correct integration of machine learning models or rule-based algorithms with other components. The objective is to guarantee that the parts function in harmony, that data is accurately transferred across modules, and that the desired results are produced.

By performing integration testing, any problems or disperancies in the integration process may be found and fixed, resulting in a reliable and effective document classification system.

**7.3 Acceptance Testing:**

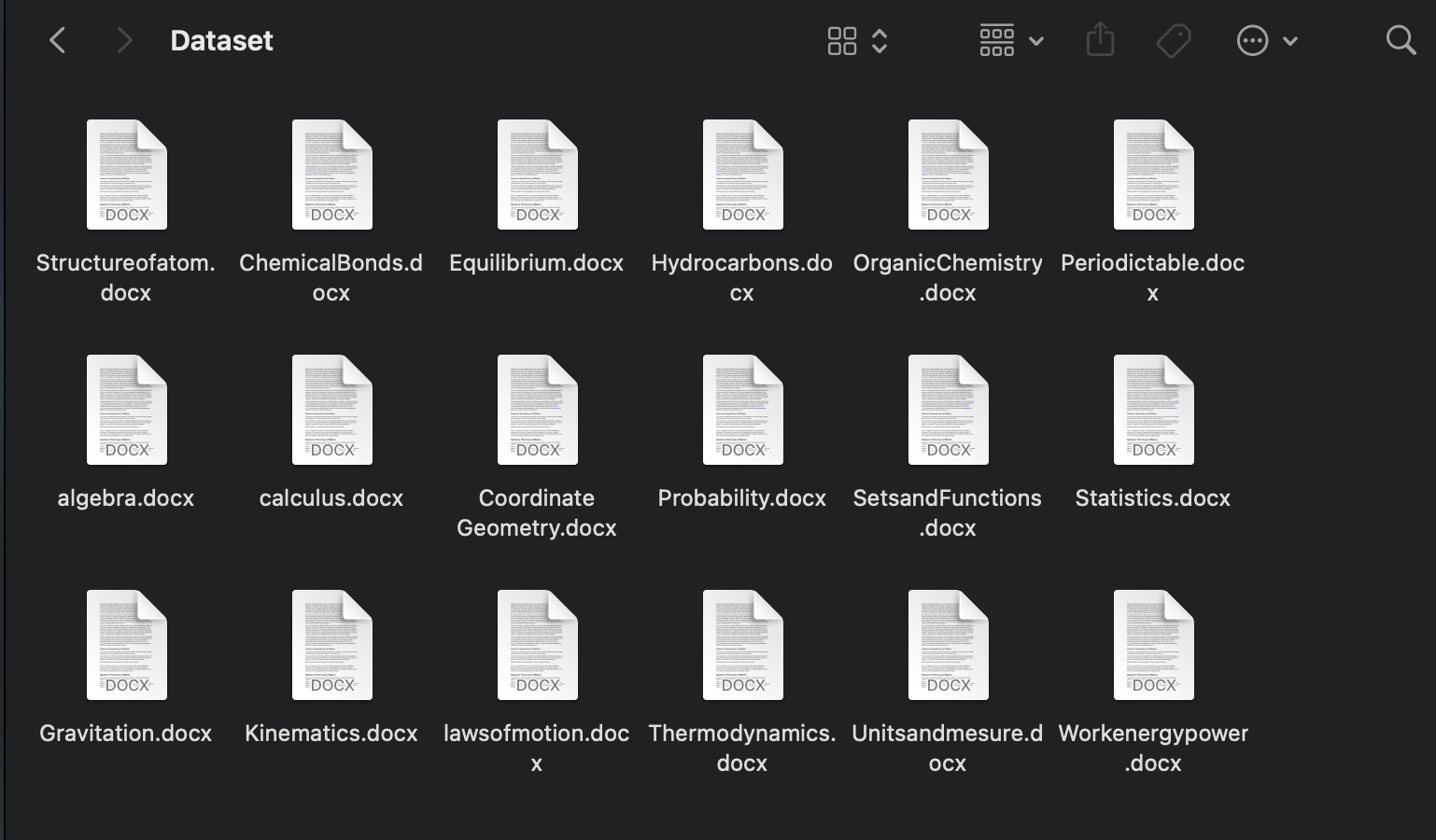
Acceptance testing is achieved with the consequences of system trying out as a starting point. this is completed to check that the anticipated and assumed necessities suit.

Examining the document classification system to make sure it satisfies the intended requirements and acceptance criteria is the main goal of acceptance testing for document classification utilising similarity analysis. During this testing step, the system's overall performance, including its correctness, dependability, and robustness, are evaluated. It might entail putting the system to the test using a variety of keywords/tokens that are extracted from the important nouns from the vast dataset of the topics in context. The objective is to confirm that the system can accurately and efficiently detect topics, classify them accordingly, manage the storage for the organization purposes and work consistently in a variety of documentation formats. Acceptance testing assists in ensuring that the document classification system satisfies end-user requirements and is prepared for implementation in practical applications.

**7.4 Testing on our System:**

After integrated testing, machine checking out is finished. As a end result, each purposeful and nonfunctional trying out are included in the process. The incorporated checking out output is used as the input for system checking out. This checking out is accomplished on the gadget's design or behavior.

**8. RESULTS**

****

*[Fig – 19]**Results*

**Consists of:** 36 Documents in fields of Math Physics and Chemistry taken from 11th & 12th Grade NCERT

**Document size:** 30-40 pages each

**Total number of pages processed:** 1250-1300 pages

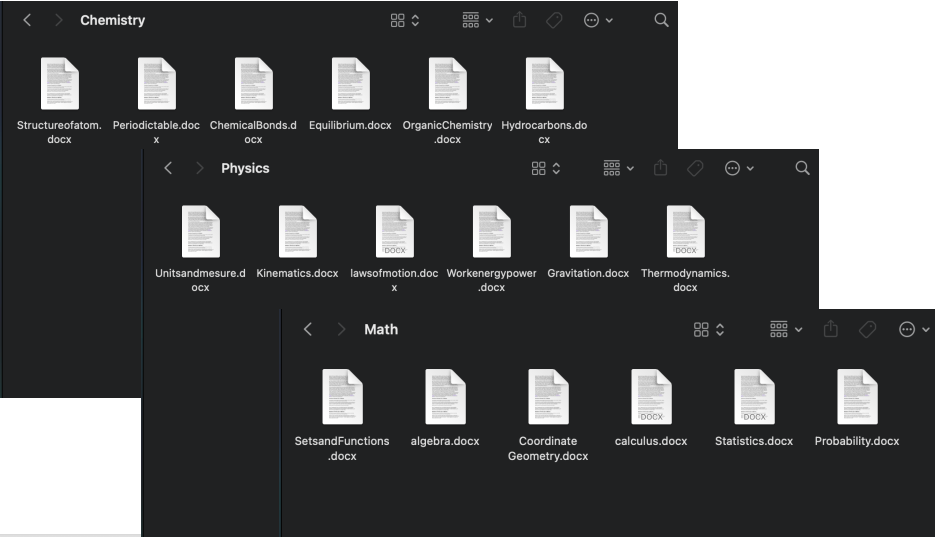
**Size of the documents in MB:** 63.2 MB

**Total number of documents classified correctly**: 34

**Total number of document classified incorrectly:** 2

**Accuracy:** 95%

**Loss:** 9.3%

****

*[Fig – 20]**Final Output*

**9. CONCLUSION AND FUTURE ENHANCEMENTS**

With the above approach where the power of lemmatization, tokenization, Cosine similarity analysis and KNN is leveraged, we were able to achieve 95% accuracy. There is a long way to go to achieve a cent percent accuracy level through extraction of better and more keywords, improving similarity analysis, and reducing the run time.

Future features we’d like to include comprise, a fully functional GUI that blends cohesively in the Operating system environment that can be used easily, creation of folders automatically according to the topic of the document if the folder it needs to reside doesn’t exist, naming the documents and the folders they shall reside in automatically using topic modelling techniques.

.

**10. BIBLIOGRAPHY**

[1][https://www.researchgate.net/publication/371255754\_SMART\_DOCUMENT\_CLASSIFICATIO](https://www.researchgate.net/publication/371255754_SMART_DOCUMENT_CLASSIFICATION)N

# [2]<https://medium.com/analytics-vidhya/how-i-built-a-document-classification-system-using-deep-convolutional-neural-networks-e1d9a83cbabd>

# 

[3]<https://www.researchgate.net/publication/348431225_A_Document_Classification_using_NLP_and_Recurrent_Neural_Network>

[4]<https://www.diva-portal.org/smash/get/diva2:1370362/FULLTEXT01.pdf>

[5]https://scholar.ppu.edu/bitstream/handle/123456789/8980/Abstract%20Classification.pdf?sequence=1&isAllowed=y

# [6] <https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1531&context=etd_projects>

[7] https://pypi.org/project/rake-nltk/

[8] <https://www.nltk.org/>

[9] <https://spacy.io/>