



CONTENT BASED AUTOMATIC CLASSIFICATION OF RESEARCH ARTICLES

A MINI PROJECT REPORT

Submitted by

CHANDREESH S (1920106701)

NIVAS R (1920106060)

RAJA K (1920106711)

*in partial fulfillment for the award of the degree
of*

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

**SONA COLLEGE OF TECHNOLOGY,
SALEM-5 (Autonomous)**

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2023

BONAFIDE CERTIFICATE

Certified that this report "**CONTENT BASED AUTOMATIC CLASSIFICATION OF RESEARCH ARTICLES**" is the bonafide work of "**NIVAS R (1920106060), CHANDREESH S (1920106701), RAJA K (1920106711)**" who carried out the project work under my supervision.

SIGNATURE

Dr. J. Akilandeswari
Professor

HEAD OF THE DEPARTMENT

Department of Information Technology
Sona College of Technology,
Salem-636 005.

SIGNATURE

Dr. J. Jeba Emilyn
Associate Professor

SUPERVISOR

Department of Information Technology
Sona College of Technology,
Salem-636 005.

Submitted for Mini Project viva voce examination held from 16/11/2023 to 18/11/2023

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, we thank the **power of the almighty** for showing us inner peace and for all blessings. Special gratitude to our parents, for showing their support and love always.

We express our sincere thanks to Chairman **Sri.C.VALLIAPPA** and Principal **Dr.S.R.R.SENTHILKUMAR** for providing adequate facilities to complete the project.

We are immensely grateful to the Head of the Department of Information Technology, **Dr.J.AKILANDESWARI** for the continued encouragement to complete the project.

We heartfully thank our project coordinator **Dr.P.SHANMUGARAJA** for all of his support and guidance throughout this project. Your leadership has been instrumental in our success.

We express our heartfelt thanks to our project supervisor **Dr.J.JEBA EMILYN** for her valuable guidance and fruitful discussions throughout the course of the project work.

We feel proud of sharing this success with all our department faculty, staff members, and friends who helped directly and indirectly to complete this project successfully.

ABSTRACT

The primary objective of the 'Content-Based Automatic Classification of Research Articles' project is to create an advanced system that can autonomously categorize research papers based on the methodologies employed in the studies. This project addresses the need for more efficient and precise information retrieval in academic research by developing a machine learning-driven approach. By recognizing and extracting key details about research methodologies from research papers, The proposed method applied on Research Articles datasets as Training and Testing Datasets which enhance the classification process by using the term frequency - inverse document frequency (TF-IDF) along with features extraction and selection. After finding the concept weight of TF-IDF the Random Forest classifier used to classify the Research articles to the pre-defined category.

LIST OF FIGURES

FIGURE NO.	TITLE	PG NO.
3.1	Architecture of the proposed system for Content-Based Automatic classification of Research Articles	9
3.1	Block Diagram	10
3.2	Dataset Description	11
3.3	Dataframe with Abstract Column	14
3.4	Dataframe with TF-IDF score	16
4.1	Working of the Model (Input)	19
4.2	Pre-processing	19
4.3	Visualizing TF-IDF scores	20
4.4	Visualizing the Noun words with Frequency	20
4.5	Classified Domains of the Documents	21
4.6	Comparison between word2vec and Tf-idf model	21

LIST OF TABLES

TABLE NO.	TITLE	PG NO.
1.1	Sample body text for domain classes	2

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	IV
	LIST OF FIGURES	V
1.	INTRODUCTION	1
	1.1 Background & Motivation	1
	1.2 Problem Identification	3
	1.3 Objectives	3
2.	LITERATURE SURVEY	4
	2.1 Overview of Content based Classification	4
	2.2 Existing Techniques	4
	2.2.1 Word Embedding	4
	2.2.2 Word2Vec	5
	2.3 Challenges and Limitations in existing methods	6
3.	PROJECT DESCRIPTION	7
	3.1 Proposed Methodology	7
	3.2 Software Requirements	10
	3.3 Block Diagram	10
	3.4 Source of data & dataset description	11
	3.4.1 Data Pre-Processing	11
	3.5 TF-IDF	13

3.6 Modules and Packages	14
3.6.1 Gathering Data	14
3.6.1.1 PyPDF2	14
3.6.1.2 Pandas	14
3.6.2 Cleaning Data	14
3.6.2.1 Regex	14
3.6.2.2 Porter Stemmer	15
3.6.2.3 Removing stop words	15
3.6.2.4 Tokenization	15
3.6.2.5 Part-of-speech Tag	15
3.6.3 Analyzing Data	16
3.6.3.1 Visualizing the dataset	16
3.6.3.2 TF-IDF score	16
3.6.4 Training the model	17
3.6.4.1 Train and Test split	17
3.6.4.2 CountVectorizer	17
3.6.4.3 TfidfTransformer	17
3.6.4.4 Making a Pipeline	17
3.7 Classified Documents	18
4. RESULTS & DISCUSSION	19

4.1 Dataset upload	19
4.2 Model Input	19
4.2.1 Perform Pre-processing	19
4.2.2 Visualization of TF-IDF score	20
4.2.3 Dataframe for noun words	20
4.3 Predicting the Document domain	21
4.3 Comparative Analysis	21
5. CONCLUSION AND FUTURE ENHANCEMENTS	22
5.1 Conclusion	22
5.2 Future Enhancement	23
APPENDICES	24
Appendix 1 Sample Script	24
REFERENCES	31

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND & MOTIVATION

Automatic classification of Document is the process of allocating documents to a pre-defined category automatically. With the rapid growth and improvement in the Web 2.0 it becomes essential to make the document classification automatically. Proper classification of Research articles, e-documents, online news and reviews, blogs, e-mails, and digital libraries need text mining, machine learning and natural language processing techniques to acquire meaningful information by classifying in predefined categories. To minimize the document complexity and handling mechanism the transformation method into document vector from full text version is encouraged, this is one of the pre-processing steps of the document representation.

As the volume of research literature continues to grow, content-based automatic classification will play an increasingly important role in organizing, managing, and accessing scientific information. By automating the classification process, researchers can save time and effort, while also improving the accuracy and consistency of classification results. Systematic Reviews: Automated classification can assist researchers in identifying relevant articles for systematic reviews, reducing the time and effort required for manual screening. Automated classification can be used to monitor research trends and developments in specific fields, enabling researchers to stay up-to-date with the latest findings. This model can identify patterns and relationships between articles, facilitating knowledge synthesis and discovery. Also it can connect researchers with similar interests and expertise, fostering collaboration and knowledge exchange.

Steps involved in sentiment analysis:

✓ Data Collection and Preparation

Dataset obtained from the Internet.

✓ Pre-processing the dataset

Dataset is pre-processed and modified according to the project.

✓ Analyzing the Data

Dataset is Analyzed and done feature selection.

✓ Visualizing the Data

Packages used: NumPy, pandas

Table 1.1 Sample body text for domain classes

DOMAIN	TEXT BODY
Astrophysics of galaxies	"We perform a joint analysis of the abundance, the clustering, and the galaxy–galaxy lensing signal of galaxies measured from Data Release 11 of the Sloan Digital Sky Survey III Baryon Oscillation Spectroscopic Survey in our companion paper, Miyatake et al. The lensing signal was obtained by using the shape catalog of background galaxies from the Canada France Hawaii Telescope Legacy Survey, which was made publicly available by the CFHTLenS collaboration, with an area overlap of about 105 deg ² ."
Robotics	"The aging population is growing at an unprecedented rate globally and robotics-enabled solutions are being developed to provide better independent living for older adults. In this study, we report the results from a systematic review of the state-of-the-art in home robotics research for caring for older adults."
Applications	"We attempt to identify the 25 most-cited statistical papers, providing some brief commentary on each paper on our list. This list consists, to a great extent, of papers that are on non-parametric methods, have applications in the life sciences, or deal with the multiple comparisons problem. We also list the most-cited papers published in 1993 or later. In contrast to the overall most-cited papers, these are predominately papers on Bayesian methods and wavelets. We briefly discuss some of the issues involved in the use of citation counts."

1.2 PROBLEM IDENTIFICATION

Traditional classification methods often rely on keywords, abstracts, or manual annotation, which can be time- consuming and less accurate. In contrast, our content-based approach leverages machine learning techniques and natural language processing to analyze the core methodology employed in research papers. This project introduces a novel approach for automatically categorizing research papers based on their content methodologies.

1.3 OBJECTIVES

Content-based classification of research articles is a machine learning technique that automatically assigns research articles to relevant categories or topics based on their content. This technique has several objectives such as, Content-based classification can help researchers and librarians organize and manage vast amounts of research literature by automatically grouping articles into meaningful categories. This makes it easier to find relevant articles on specific topics or by specific authors. Content-based classification can enhance literature search and discovery by providing a more effective way to query and filter research articles. By understanding the content and topics of articles, search engines and recommendation systems can provide more relevant and targeted results to researchers.

Content-based classification can facilitate knowledge synthesis and analysis by identifying patterns and trends in research. By analyzing the distribution of articles across different categories, researchers can gain insights into the overall landscape of research in a particular field. Content-based classification can automate knowledge management tasks, such as tagging, indexing, and categorization.

CHAPTER 2

LITERATURE SURVEY

This chapter addresses about the prior designed models and its limitations.

2.1 OVERVIEW OF CONTENT BASED CLASSIFICATION

Document classifiers has One of the biggest limitations of document classifiers is their accuracy. Document classifiers are not always accurate, especially when they are trained on a limited dataset or when the documents are complex or ambiguous. Another limitation of document classifiers is their robustness. Document classifiers can be biased towards the training data, and they may not perform well on new data that is different from the training data. This is because document classifiers learn to identify patterns in the training data, and if these patterns do not exist in the new data, the classifier may make inaccurate predictions. Finally, it can be difficult to understand why a document classifier makes certain predictions, especially for complex machine learning models. This is because document classifiers learn complex patterns in the data, and it can be difficult to reverse engineer these patterns to understand the rationale behind the predictions.

2.2 EXISTING TECHNIQUES

2.2.1 WORD EMBEDDING

Word embedding in NLP is an important term that is used for representing words for text analysis in the form of real-valued vectors. It is an advancement in NLP that has improved the ability of computers to understand text-based content in a better way. It is considered one of the most significant breakthroughs of deep learning for solving challenging natural language processing problems. In this approach, words and documents are represented in the form of numeric vectors allowing similar words to have similar vector representations. The extracted features are fed into a machine learning model to work with text data and preserve the semantic and syntactic information. This information

once received in its converted form is used by NLP algorithms that easily digest these learned representations and process textual information. Word embeddings are able to capture the semantic meaning of words. This is important for document classification, as it allows the classifier to learn relationships between words that are not explicitly mentioned in the training data. It was more robust to noise than other text representation techniques. This is important for document classification, as many real-world datasets are noisy and contain errors. They can represent documents of different lengths. This is important for document classification, as documents can vary widely in length.

2.2.2 WORD2VEC

Word2Vec is a recent breakthrough in the world of NLP. Tomas Mikolov, a Czech computer scientist and currently a researcher at CIIRC (Czech Institute of Informatics, Robotics and Cybernetics), was one of the leading contributors to the research and implementation of word2vec. Word embeddings are an integral part of solving many problems in NLP. They depict how humans understand language to a machine. You can imagine them as a vectorized representation of text. Word2Vec, a standard method of generating word embeddings, has a variety of applications, such as text similarity, recommendation systems, sentiment analysis, etc. Word2Vec builds word vectors, which are distributed numerical representations of word features. These word features may include words that indicate the context of the specific vocabulary words present individually. Through the generated vectors, word embeddings eventually assist in forming the relationship of a word with another word having a similar meaning.

2.3 CHALLENGES & LIMITATIONS IN EXISTING METHODS

Word embeddings have revolutionized the field of natural language processing (NLP), enabling machines to better understand and process human language. However, despite their significant advantages, word embeddings and techniques like word2vec also face certain challenges and limitations when applied to document classification tasks using machine learning.

Word2vec and similar techniques primarily focus on local context, meaning they capture the relationships between words that appear close together in a sentence or paragraph. While this is useful for understanding word meanings, it may not adequately capture broader semantic relationships and long-range dependencies within a document. Word2vec's performance can suffer when dealing with rare words, as these words have fewer co-occurrence statistics to learn from. This can lead to inaccurate word representations and affect the overall classification accuracy. Word embeddings trained on general-purpose corpora may not accurately capture the nuances and specialized vocabulary of specific domains. This can hinder classification performance in tasks that require domain-specific knowledge. Training word embeddings can be computationally expensive, especially for large datasets and high-dimensional vectors. This can make it challenging to scale these techniques to real-world applications.

Natural language is inherently ambiguous, with words having multiple meanings and contexts. Word embeddings may struggle to capture this ambiguity, leading to misinterpretations and classification errors. Word embeddings do not directly incorporate common sense knowledge or real-world understanding, which are crucial for certain types of document classification tasks. Word embeddings represent words as static vectors, which may not capture the dynamic nature of word meanings and their usage in different contexts.

CHAPTER 3

PROJECT DESCRIPTION

This chapter addresses about the requirements, modules, and packages essential for the project.

3.1 PROPOSED METHODOLOGY

The focus of the proposed approach is to enhance the multiple document classification and to classify the documents according to the pre-defined domains. The process of Electronic Research articles classification requires the supervised learning techniques to convert text into vector model. Random Forest Classifier is used for the classification. The Research articles are arranged as Vector model and their contents are represented as words or terms of vector notation. These notations are also known as bag-of-words. The bag-of-words are notated with a part of speech tag. The tagged notation contains both opinion and non-opinion terms. The opinion terms are considered for onward process and non-opinion words are removed which are also called stops-words like “of”, “on”, “the”, “an” etc. the stemming algorithm converts the word of different level to root level. The opinion word contains Noun, Verb, Adverb and Adjective. During this process various features are extracted and then TF-IDF is applied to select the relevant features. Using the Random Forest Classifier to classify the articles to the pre-defined category considering the contents of testing datasets mapping with training datasets.

Following are the steps of content-based classification of research articles using TF-IDF and Random Forest Classifier.

1. Select articles and perform processing steps for noise removal.
2. Split the noise free article text into sentences.
3. Remove stop-words from the tokens of the sentences.
4. Apply stemming algorithm to leaves out the word root form.
5. Part of speech tagging of all tokens using nltk Part-of-speech tagger.

6. Collect the noun, noun phrases, adjectives, verb, and adverb along with their word position in the sentence.
7. extract the features list from key noun phrases
8. Apply TF Algorithm to extract the most frequent terms.
9. Find the TFIDF (Term Frequency and Inverse Document Frequency) to calculate the weight.
10. Calculate the highest relevance sentence using maximum weight or frequency in the document and give maximum weight.
11. Train the classifier using the new weighted term vector.
12. Calculate similarity between the training and testing documents.
13. Assign the article to the relevant category.

Feature Extraction:

The opinion words list such as Noun, Verb, Adjective and Adverb are produced along with POS tag. This list is further used for selection of features using Term Frequency (TF) after selecting the Noun words from the list.

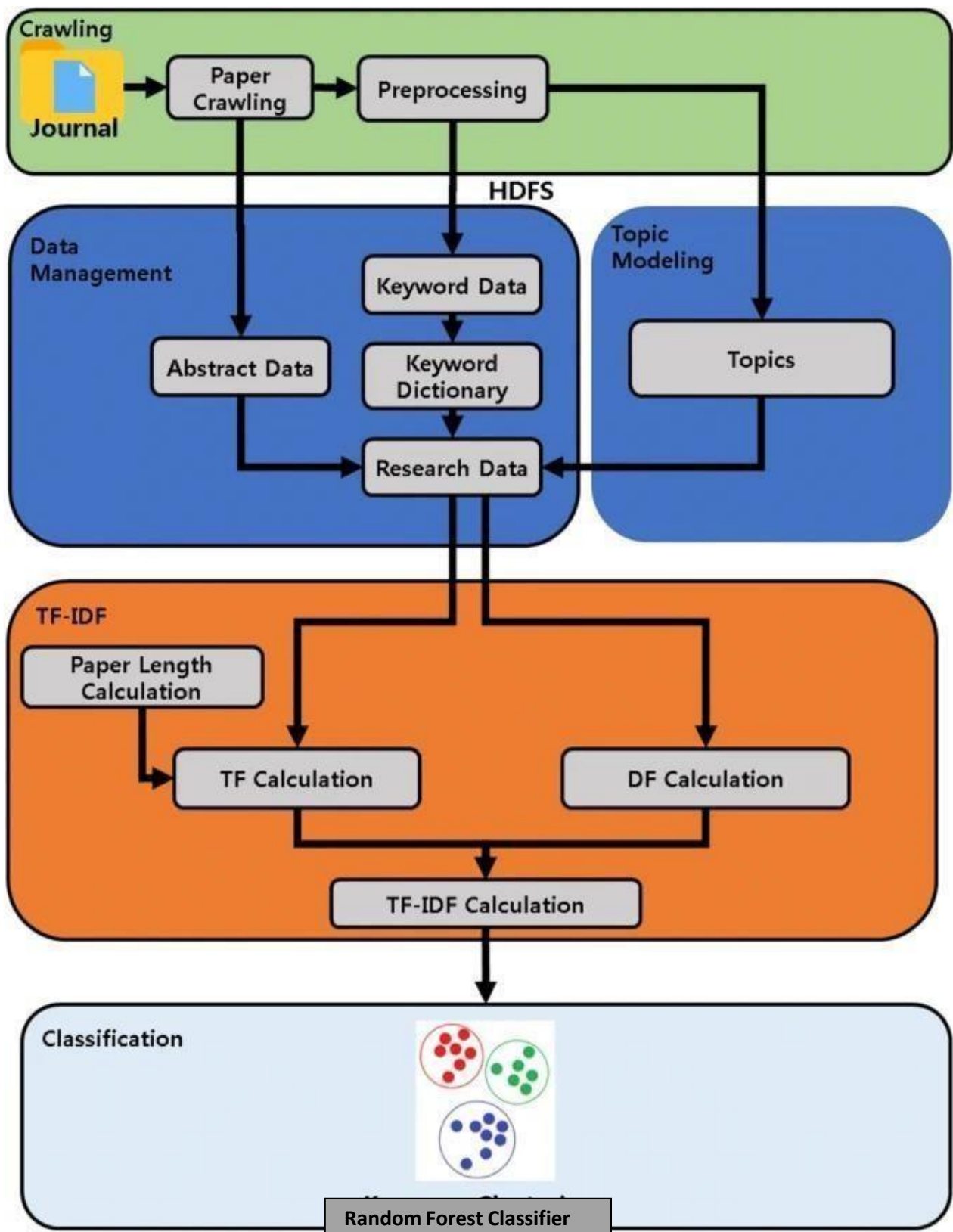
Frequency base Features Selection Process:

The produced features list consists of many features. The most frequent words which are greater than threshold value is selected as a frequent term and is considered for further classification steps.

Random Forest Classifier:

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Fig.3.1 Architecture of the proposed system for Content-Based Automatic classification of Research Articles

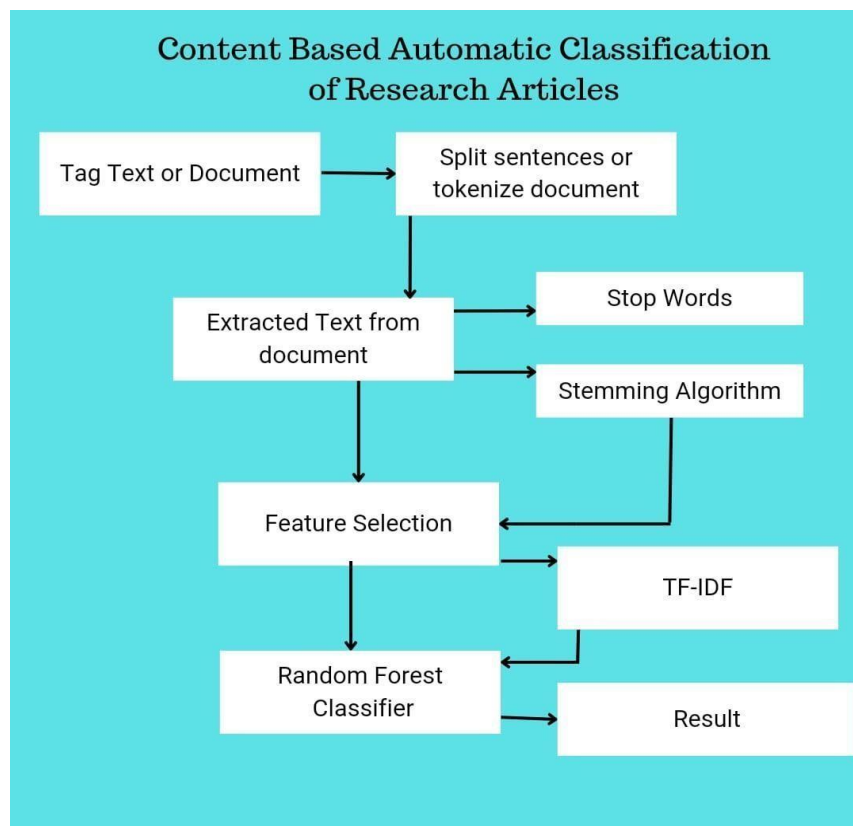


3.2 SOFTWARE REQUIREMENTS

- ✓ Linux Operating System/Windows
- ✓ Python Platform (Google Collab, Anaconda2, Jupyter)
- ✓ NLTK package
- ✓ PyPDF2 package
- ✓ Modern Web Browser

3.3 BLOCK DIAGRAM

Fig 3.1 Block diagram



3.4 SOURCE OF DATA AND DATASET DESCRIPTION

Researchers have access to large online archives of scientific articles. As a consequence, finding relevant articles has become more and more difficult. Tagging or topic modelling provides a way to give clear tokens of identification to research articles.

Each research article should be associated with relevant labels or categories, representing the research topics or themes. This labeled dataset serves as the foundation for supervised machine learning. To enable machine learning algorithms to process the textual content of these articles, the articles' text is preprocessed.

Fig 3.2 Dataset Description

	ABSTRACT	Label_val_2
0	a ever-growing datasets inside observational a...	Cosmology
1	we propose the framework considering optimal \$...	Data Structures and Algorithms
2	nanostructures with open shell transition meta...	Strongly Correlated Electrons
3	stars are self-gravitating fluids inside which...	Fluid Dynamics
4	deep neural perception and control networks ar...	Computer vision and Pattern Recognition

3.4.1 DATA PRE-PROCESSING

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data preprocessing is a technique that is used to convert raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for analysis. For achieving better results from the applied model in Machine Learning projects the format of the data must be in a proper manner. Some specified Machine Learning models need information in a specified format, for example, Random Forest algorithm does not support null values, therefore, to execute random forest algorithm null values must be managed from the original raw data set.

Data preprocessing consists of 3 methods:

- 1) Tokenization
- 2) Stemming
- 3) Part-of-speech (POS)

Tokenization: It is the process of breaking a stream of text up into words, symbols and other meaningful elements called “tokens”. Tokens can be separated by whitespace characters and/or punctuation characters. It is done so that tokens are taken as individual components that make up a tweet. Emoticons and abbreviations (e.g., OMG, WTF, BRB) are identified as part of the tokenization process and treated as individual tokens.

Stemming: Stemming is a process where words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix. It is a rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.

Part-of-speech (POS):

Part-of-speech (POS) tagging is an important Natural Language Processing (NLP) concept that categorizes words in the text corpus with a particular part of speech tag (e.g., Noun, Verb, Adjective, etc.)

POS tagging could be the very first task in text processing for further downstream tasks in NLP, like speech recognition, parsing, machine translation, sentiment analysis, etc.

The POS tag of a word can be used as a feature by various Machine Learning algorithms used in Natural Language Processing.

3.5 TF-IDF

TF-IDF defines importance of a term by taking into consideration the importance of that term in a single document and scaling it by its importance across all documents.

Importance of a term in a document (term frequency): $tf(t,d)$

Term frequency answers the question of, how many times does this word appear in this document among the number of times all words appear in this document? In other words, how important is this word to this specific document?

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Image by author. Mathematical definition of term frequency

Importance of a term across all documents (inverse document frequency): $idf(t,D)$

Inverse document frequency answers the question of, how common (or uncommon) is this word among all the documents I have?

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

Image by author. Mathematical definition of inverse document frequency

We have 2 documents at this point. The resulting inverse document frequency of the word 'house' is represented by $idf(\text{'house'}, D) = \log(2/1)$, because it appears in 1 out of the 2 documents in our collection.

Similarly, the inverse document frequency of the word 'the' is represented by $idf(\text{'the'}, D) = \log(2/2)$

3.6 MODULES AND PACKAGES

3.6.1 GATHERING DATA

3.6.1.1 PyPDF2

PyPDF2 is a pure-Python library for working with PDF files. It can be used to create, split, merge, crop, and transform PDF pages, as well as extract text and images from PDFs. PyPDF2 also supports encryption and decryption of PDFs, adding and removing annotations, and interacting with PDF forms. It is a powerful and versatile tool for working with PDF files. It is easy to use and install, and it is well-documented. PyPDF2 is also free and open-source software, making it a great choice for both commercial and non-commercial use.

3.6.1.2 Pandas

To retrieve the Abstract columns from the dataset and to visualize Extracted text from the document, A dataframe is created with a column called Abstract using **pandas**.

Fig.3.2 Dataframe with Abstract column

	ABSTRACT	Label_val_2
0	a ever-growing datasets inside observational a...	Cosmology
1	we propose the framework considering optimal \$...	Data Structures and Algorithms
2	nanosstructures with open shell transition meta...	Strongly Correlated Electrons
3	stars are self-gravitating fluids inside which...	Fluid Dynamics
4	deep neural perception and control networks ar...	Computer vision and Pattern Recognition

3.6.2 CLEANING DATA

3.6.2.1 Regex

A **preprocess** function created which removes symbols, numbers, and hyperlinks.

3.6.2.2 Porter stemmer

Stemming is a natural language processing technique that is used to reduce words to their base form, also known as the root form. The process of stemming is used to normalize text and make it easier to process.

3.6.2.3 Removing Stop Words

The NLTK package has a separate package of stop words that can be downloaded. NLTK has stop words in 16 languages which can be downloaded and used. Once it is downloaded, it can be passed as an argument indicating it to ignore these words.

3.6.2.4 Tokenization

Tokenization is the process by which a large quantity of text is divided into smaller parts called tokens. These tokens are very useful for finding patterns and are considered as a base step for stemming and lemmatization. We are using `RegexpTokenizer` from `nltk` library for this purpose.

3.6.2.5 Part-of-speech Tag

POS tagging is a powerful method in text processing for further downstream tasks in NLP, like speech recognition, parsing, machine translation, sentiment analysis, etc. The POS tag of a word can be used as a feature by various Machine Learning algorithms used in Natural Language Processing.

3.6.3 ANALYZING DATA

3.6.3.1 Visualizing the Dataset

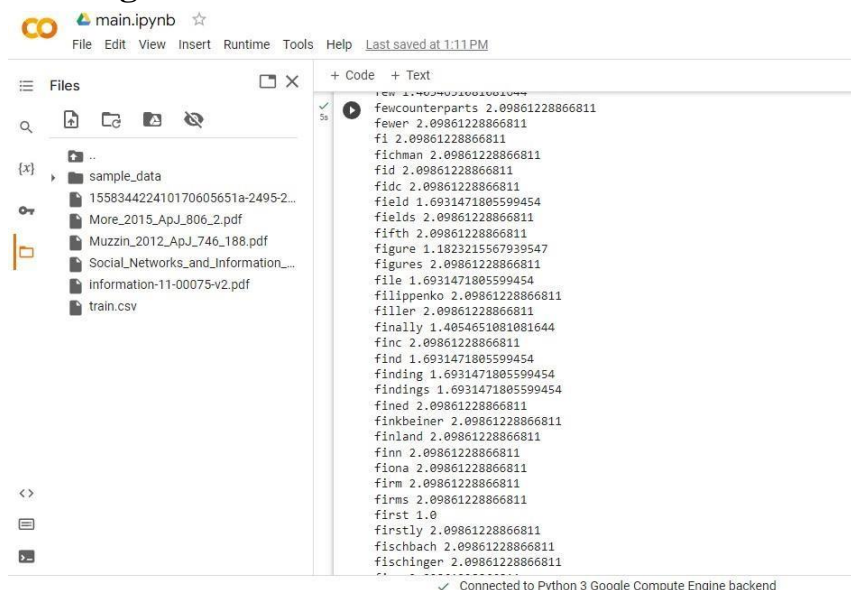
In the dataset, we have two Columns namely ABSTRACT, it contains the body of Research articles and its content and LABEL_VAL, it contains the category of the Research article, and it is used to train our model.

3.6.3.2 TF-IDF

Term Frequency-Inverse Document Frequency: TF-IDF determines how important a word is by weighing its frequency of occurrence in the document and computing how often the same word occurs in other documents. If a word occurs many times in a particular document but not in others, then it might be highly relevant to that document and is therefore assigned more importance.

By using `tfidfvectorizer()` function is used to calculate the tf-idf scores for each tokens (noun words) in the corpus and it is prepared for the prediction of document category, That is the another column by using it in the trained model.

Fig 3.3 Dataframe with TF-IDF score



3.6.4 TRAINING THE MODEL

3.6.4.1 Train and test split

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model. Then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

3.6.4.2 CountVectorizer

CountVectorizer is a text preprocessing technique commonly used in natural language processing (NLP) tasks for converting a collection of text documents into a numerical representation. It is part of the scikit-learn library, a popular machine learning library in Python.

3.6.4.3 TfidfTransformer

The TfidfTransformer is a machine learning library that transforms a count matrix of term/token counts into a tf-idf matrix. Tf-idf stands for term frequency-inverse document frequency. It is a statistical measure that reflects how important a term is to a document in a collection of documents. It works by first computing the term frequency of each term in each document. Then, it computes the inverse document frequency of each term. The inverse document frequency is a measure of how common a term is in a collection of documents. The more common a term is, the lower its inverse document frequency will be.

3.6.4.4 Making a Pipeline

Creating a Pipeline after Train and Test split of the Dataset, To put our CountVectorizer, TfidfTransformer, Random Forest Classifier together into that Pipeline. After this, fit the Train and Test data with the pipeline.

3.7 CLASSIFIED DOCUMENTS

Multiclass classification is a classification task with more than two classes.

Each sample can only be labeled as one class.

We are using Random Forest Classification for our model to predict the category (Domain) of our document.

Example:

The aging population is growing at an unprecedented rate globally and robotics-enabled solutions are being developed to provide better independent living for older adults. In this study, we report the results from a systematic review of the state-of-the-art in home robotics research for caring for older adults. This review aims to address two questions: (1) What research is being done towards integrating robotics for caring for older adults? (2) What are the research and technology challenges that robots are facing in the home? Sixty-three papers have been identified and studied in this review by following the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines. Common themes that are consistent across the reviewed papers are distinguished and consolidated as follows: (1) Ambient assisted living, where smart home environments and physical support tools are studied; (2) Robot ecosystem, where robotic devices are used to provide various services; (3) Social interaction, where the social isolation problem has been targeted. We also summarize the results of similar literature reviews we came across during our search. The results of this study present the current research trends and technologies used in each category. The challenges and limitations of robotics applications are also identified. Suggestions for accelerating the deployment of robots at home for providing older adults with independent care in the home are presented based on the results and insights from this study.

OUTPUT:

Predicted Domain of the Document: **ROBOTICS**

CHAPTER 4

RESULTS & DISCUSSION

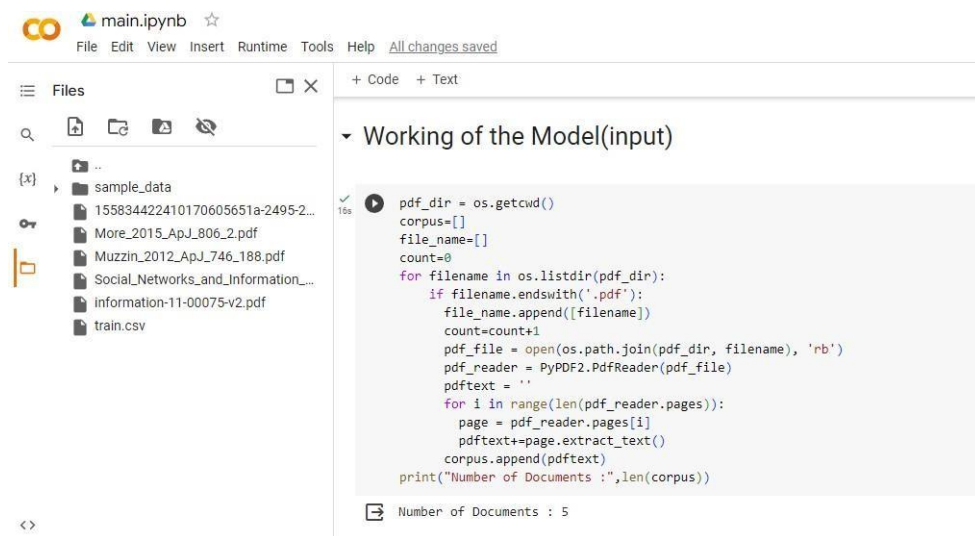
This chapter addresses the results of the project with provided interface.

4.1 DATASET UPLOAD

Used a set of research articles documents from the Internet and predicted using the model.

4.2 MODEL INPUT

Fig 4.1 Working of the Model (Input)



```
main.ipynb
File Edit View Insert Runtime Tools Help All changes saved

Files
sample_data
155834422410170605651a-2495-2...
More_2015_ApJ_806_2.pdf
Muzzin_2012_ApJ_746_188.pdf
Social_Networks_and_Information...
information-11-00075-v2.pdf
train.csv

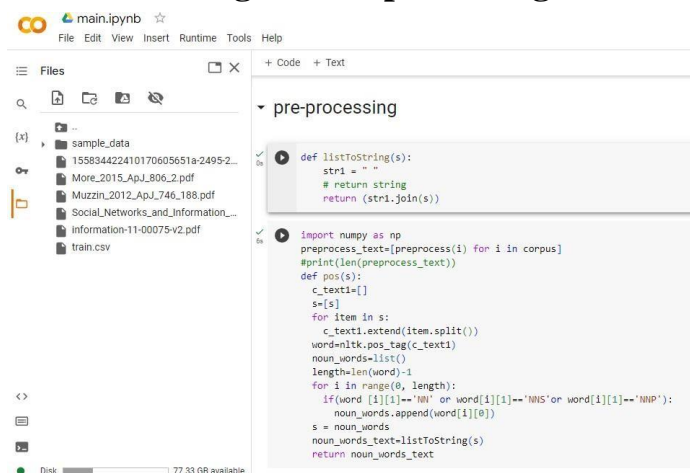
Working of the Model(input)

pdf_dir = os.getcwd()
corpus=[]
file_name=[]
count=0
for filename in os.listdir(pdf_dir):
    if filename.endswith('.pdf'):
        file_name.append(filename)
        count=count+1
pdf_file = open(os.path.join(pdf_dir, filename), 'rb')
pdf_reader = PyPDF2.PdfReader(pdf_file)
pdftext = ''
for i in range(len(pdf_reader.pages)):
    page = pdf_reader.pages[i]
    pdftext+=page.extract_text()
corpus.append(pdftext)
print("Number of Documents :",len(corpus))

Number of Documents : 5
```

4.2.1 PERFORM PRE-PROCESSING

Fig 4.2 Pre-processing



```
main.ipynb
File Edit View Insert Runtime Tools Help

Files
sample_data
155834422410170605651a-2495-2...
More_2015_ApJ_806_2.pdf
Muzzin_2012_ApJ_746_188.pdf
Social_Networks_and_Information...
information-11-00075-v2.pdf
train.csv

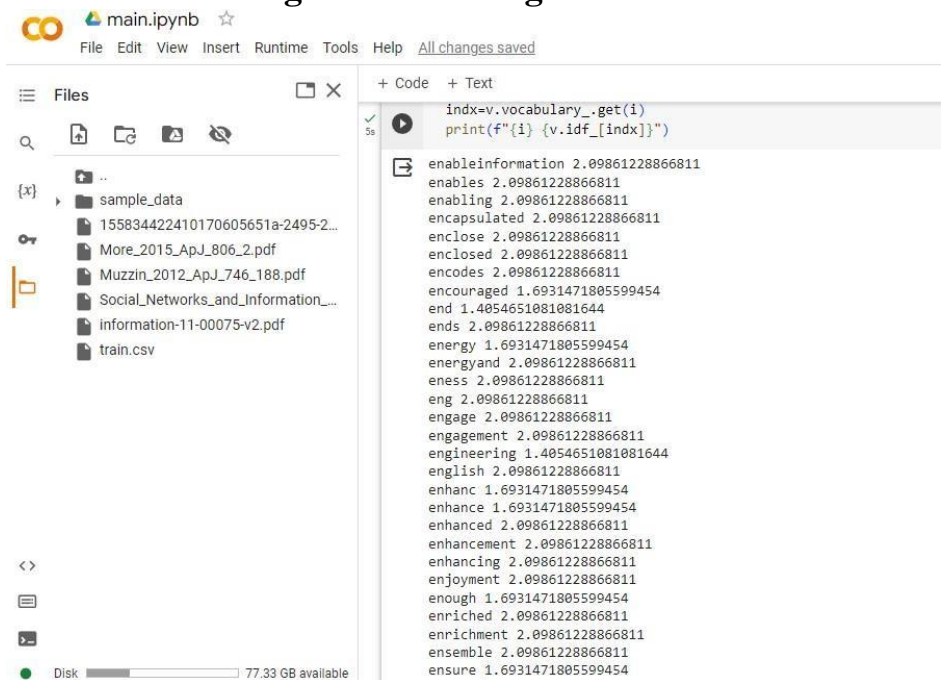
pre-processing

def listToString(s):
    str1 = " "
    # return string
    return (str1.join(s))

import numpy as np
preprocess_text=[preprocess(i) for i in corpus]
#print(len(preprocess_text))
def pos(s):
    c_text1=[]
    s=[s]
    for item in s:
        c_text1.extend(item.split())
    word=nlk.pos_tag(c_text1)
    noun_words=list()
    length=len(word)-1
    for i in range(0, length):
        if (word[i][1]=="NN" or word[i][1]=="NNS" or word[i][1]=="NNP"):
            noun_words.append(word[i][0])
    s = noun_words
    noun_words_text=listToString(s)
    return noun_words_text
```

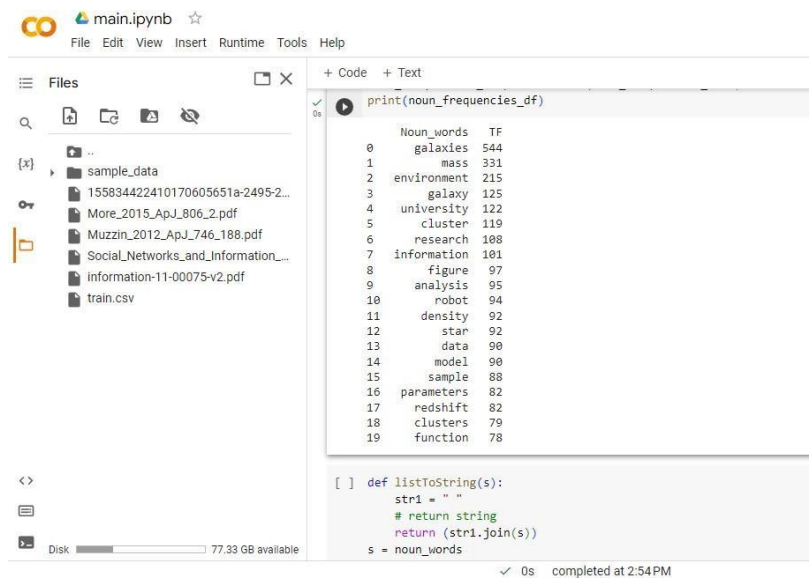
4.2.2 VISUALIZATION OF TF-IDF SCORES

Fig 4.3 Visualizing TF-IDF scores



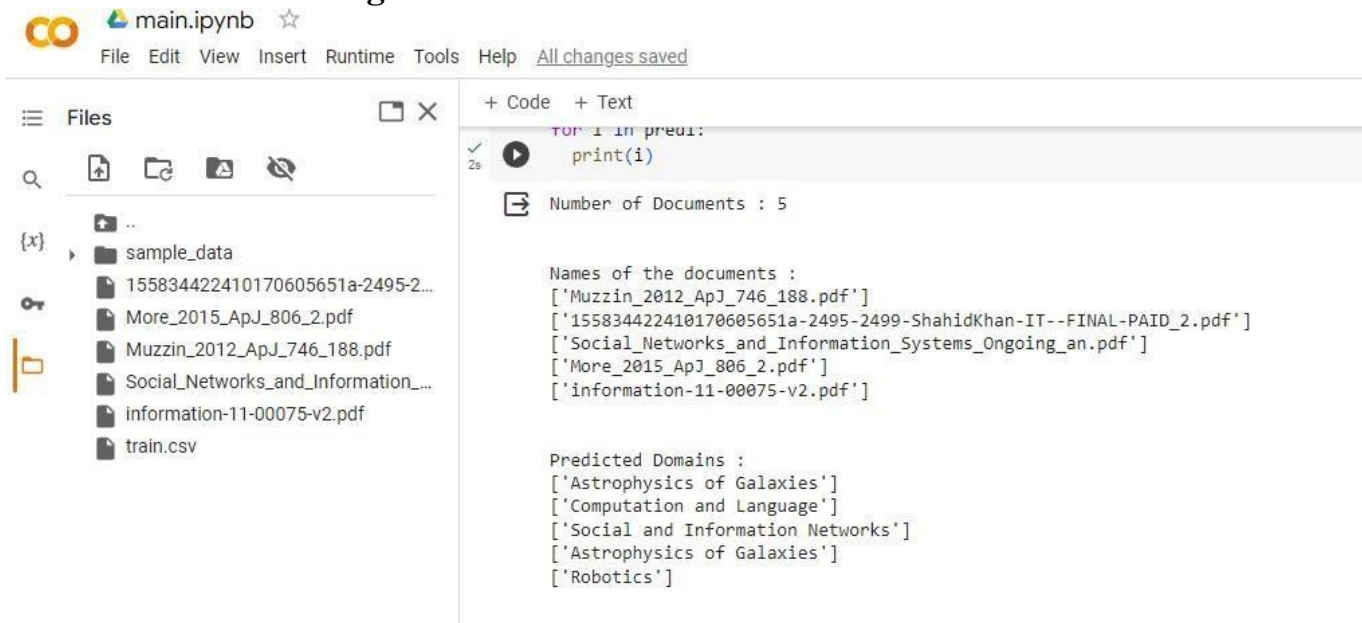
4.2.3 DATAFRAME FOR NOUN WORDS

Fig 4.4 Visualizing the Noun words with Frequency



4.3 PREDICTING THE DOCUMENT DOMAIN

Fig 4.5 Classified Domains of the Documents



4.4 COMPARATIVE ANALYSIS OF THE MODEL

Fig 4.6 Comparison between word2vec and Tf-idf model

accuracy 0.8045				
	precision	recall	f1-score	support
java	0.73	0.68	0.70	589
html	0.89	0.91	0.90	661
asp.net	0.93	0.94	0.94	606
c#	0.80	0.80	0.80	613
ruby-on-rails	0.83	0.90	0.86	601
jquery	0.72	0.71	0.72	585
mysql	0.87	0.81	0.84	621
php	0.81	0.84	0.82	587
ios	0.68	0.67	0.67	560
javascript	0.69	0.63	0.66	611
python	0.63	0.65	0.64	593
c	0.81	0.83	0.82	581
css	0.81	0.77	0.79	608
android	0.84	0.85	0.84	593
iphone	0.84	0.82	0.83	592
sql	0.68	0.65	0.66	597
objective-c	0.84	0.86	0.85	604
c++	0.90	0.95	0.92	610
angularjs	0.93	0.96	0.95	595
.net	0.81	0.84	0.82	593
avg / total	0.80	0.80	0.80	12000

Accuracy: 97.52%				
	precision	recall	f1-score	support
Analysis of PDEs	0.99	0.99	0.99	472
Applications	0.98	0.98	0.98	504
Artificial Intelligence	0.89	0.76	0.82	447
Astrophysics of Galaxies	1.00	0.98	0.99	457
Computation and Language	0.97	1.00	0.98	511
Computer vision and Pattern Recognition	0.91	0.98	0.94	454
Cosmology	1.00	1.00	1.00	466
Data Structures and Algorithms	0.96	0.99	0.98	508
Differential Geometry	0.99	1.00	1.00	453
Earth and Planetary Astrophysics	1.00	1.00	1.00	495
Fluid Dynamics	0.99	1.00	0.99	484
Information Theory	0.99	1.00	1.00	472
Instrumentation and Methods for Astrophysics	1.00	1.00	1.00	478
Machine Learning	0.79	0.72	0.76	478
Materials Science	1.00	0.99	0.99	495
Methodology	0.99	1.00	1.00	505
Number Theory Optimization	1.00	1.00	1.00	485
Optimization and Control	1.00	1.00	1.00	455
Representation Theory	1.00	1.00	1.00	488
Robotics	0.97	0.99	0.98	463
Social and Information Networks	0.99	1.00	0.99	491
Statistics Theory	0.99	1.00	1.00	507
Strongly Correlated Electrons	0.99	0.99	0.99	466
Superconductivity	1.00	1.00	1.00	464
Systems and Control	1.00	1.00	1.00	497
accuracy			0.98	11995
macro avg	0.97	0.97	0.97	11995
weighted avg	0.97	0.98	0.97	11995

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

The 'Content-Based Automatic Classification of Research Papers' project represents a significant step forward in the realm of academic research and information retrieval. The objectives set forth have been met with success, resulting in the development of an efficient and accurate system that automates the categorization of research papers based on their methodologies. This project has not only demonstrated the superiority of the content-based approach over traditional methods but showcase its practical applications in academia. It is a powerful tool for organizing and retrieving scientific knowledge. It has the potential to help researchers discover new and relevant articles, and to improve the efficiency of literature reviews.

In particular, when a keyword dictionary with both of the keywords extracted from the abstracts and the topics extracted, our classification system has better clustering performance and higher F-Score values. Therefore, our classification systems can classify research papers in advance by both of keywords and topics with the support of high-performance computing techniques, and then the classified research papers will be applied to search the papers within users' interesting research areas, fast and efficiently.

This work has been mainly focused on developing and analyzing research paper classification. To be a generic approach, the work needs to be expanded into various types of datasets, e.g., documents. Therefore, future work involves working upon various types of datasets in the field of text mining, as well as developing even more efficient classifiers for research paper datasets.

5.2 FUTURE ENHANCEMENT

Feature extraction is the process of converting the text of a research article into a set of features that can be used by a classifier. Current feature extraction algorithms are often time-consuming and computationally expensive, and they may not be able to capture all the relevant information in a research article. Developing more accurate and efficient feature extraction algorithms would improve the performance of content-based classification models.

Developing more robust classification algorithms would improve the reliability and accuracy of content-based classification models. By using other Multi Class Classification algorithms like Naïve Bayes, Logistic Regression, KNN Classification in our model for training and predicting the document category, and to choose the best Classification algorithm for the model.

Training classification models on larger and more diverse datasets would help them to learn the patterns and relationships in the data more effectively. This would make the models more robust and generalizable, and it would improve their performance on a wider range of tasks.

The model and its accuracy will be improved and its ability to predict the domains for the documents will be strengthened by using other algorithms, and the feature of the model to predict the multi-label for a single document will be worked in upcoming development, so the model can predict multiple domains (labels) such as inter-related class can also be predicted.

APPENDICES

APPENDIX 1

SAMPLE SCRIPT

Main.py

importing libraries and packages

```
import nltk
```

```
nltk.download('all')
```

```
import PyPDF2
```

```
import os
```

```
import nltk
```

```
from nltk.tokenize import RegexpTokenizer
```

```
from nltk.stem import WordNetLemmatizer,PorterStemmer
```

```
from nltk.corpus import stopwords
```

```
import re
```

```
from nltk.tokenize import sent_tokenize
```

```
import pandas as pd
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

load the training dataset

```
df=pd.read_csv('train.csv')
```

```
df.head()
```

```
print("Total words before pre-processing :",df['ABSTRACT'].apply(lambda x:  
len(x.split(' '))).sum())
```

Pre-processing

```
stemmer = PorterStemmer()
```

```
def preprocess(sentence):
```

```
    sentence=str(sentence)
```



```

sentence=sentence.replace('{html}','')
cleanr = re.compile('<.*?>')
cleantext = re.sub(cleanr, "", sentence)

rem_url=re.sub(r'http\S+', "",cleantext)
rem_num = re.sub('[0-9]+', "", rem_url)
tokenizer = RegexpTokenizer(r'\w+')
tokens = tokenizer.tokenize(rem_num)
filtered_words = [w for w in tokens if len(w) > 2 if not w in
stopwords.words('english')]
stem_words=[stemmer.stem(w) for w in filtered_words]
return " ".join(filtered_words)
text="
df['ABSTRACT']=df['ABSTRACT'].apply(preprocess)

```

Training the model

```

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

print("Total words after pre-processing :",df['ABSTRACT'].apply(lambda x:
len(x.split(' '))).sum())
X=df.ABSTRACT
y=df.Label_val_2
X=X.values.reshape(1,-1)
X = X.transpose()
X.shape

```

```

from imblearn.over_sampling import RandomOverSampler
ros=RandomOverSampler(random_state=0)
x_resampled,y_resampled=ros.fit_resample(X,y)
x_resampled=x_resampled.reshape(x_resampled.shape[:-1])
X_train,X_test,y_train,y_test=train_test_split(x_resampled,y_resampled,test_size=0.2,random_state=42)
X_train.shape,X_test.shape,y_train.shape,y_test.shape

```

Training the model and classification (Pipeline)

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

lr=Pipeline([('vect', CountVectorizer()),
             ('tfidf', TfidfTransformer()),
             ('clf', RandomForestClassifier())])

lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
accuracy=accuracy_score(y_test, y_pred)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
print(classification_report(y_test,y_pred))

```

Evaluating the model in the test dataset

```

X_test[:5]
y_test[:5]
y_pred[:5]

```

Working of the Model (input)

```
pdf_dir = os.getcwd()
corpus=[]
file_name=[]
count=0
for filename in os.listdir(pdf_dir):
    if filename.endswith('.pdf'):
        file_name.append([filename])
        count=count+1
pdf_file = open(os.path.join(pdf_dir, filename), 'rb')
pdf_reader = PyPDF2.PdfReader(pdf_file)
pdftext = ""
for i in range(len(pdf_reader.pages)):
    page = pdf_reader.pages[i]
    pdftext+=page.extract_text()
    corpus.append(pdftext)
print("Number of Documents :",len(corpus))
```

Preprocessing the documents

```
import numpy as np
preprocess_text=[preprocess(i) for i in corpus]
#print(len(preprocess_text))
def pos(s):
    c_text1=[]
    s=[s]
    for item in s:
        c_text1.extend(item.split())
    word=nlTK.pos_tag(c_text1)
    noun_words=list()
```

```

        length=len(word)-1
        for i in range(0, length):
            if(word[i][1]=='NN' or word[i][1]=='NNS' or word[i][1]=='NNP'):
                noun_words.append(word[i][0])
        s = noun_words
        noun_words_text=listToString(s)
        return noun_words_text

def listToString(s):
    str1 = " "

    # return string
    return (str1.join(s))

```

Predicting the domain of the documents (Output)

```

print("Number of Documents :",len(corpus))
print("\n")
preprocess_text1=[pos(j) for j in preprocess_text]
#print(len(preprocess_text1))
pred=[]
print("Names of the documents :")
for n in file_name:
    print(n)
print("\n")
print("Predicted Domains :")
for i in preprocess_text1:
    i=[i]
    pred.append(lr.predict(i))
pred=np.array(pred)
pred1=pred.tolist()

```

```

for i in pred1:
    print(i)
# Visualizing the model
c_text=list(map(preprocess,corpus))
print(c_text)
c_text1=[]
for item in c_text:
    c_text1.extend(item.split())
    print(c_text1)
word=nlk.pos_tag(c_text1)
print(word)
noun_words=list()
length=len(word)-1
for i in range(0, length):
    if(word [i][1]=='NN' or word[i][1]=='NNS'or word[i][1]=='NNP'):
        noun_words.append(word[i][0])
print(noun_words)
df=pd.DataFrame(noun_words, columns=['noun_words'])
# Calculation of Tf-Idf score
v=TfidfVectorizer()
transformed_output=v.fit_transform(noun_words)
print(v.vocabulary_)
v=TfidfVectorizer()
tokenized_word=v.fit_transform(corpus)

feature_s=v.get_feature_names_out()
for i in feature_s:
    indx=v.vocabulary_.get(i)
    print(f"{i} {v.idf_[indx]}")

```

Frequency of Noun words

```
df=pd.DataFrame(noun_words, columns=['Noun_words'])
pd.set_option("display.max_rows",None)
noun_frequencies=df['Noun_words'].value_counts()
noun_frequencies1=(noun_frequencies[:20])
noun_frequencies_dict=({'Noun_words':noun_frequencies1.index,'TF':noun_frequencies1.values})
noun_frequencies_df=pd.DataFrame(noun_frequencies_dict)
print(noun_frequencies_df)
```

converting list to string

```
def listToString(s):
    str1 = " "
    # return string
    return (str1.join(s))
s = noun_words
noun_words_text1=[listToString(s)]
print(noun_words_text1)
```

REFERENCES

- 1) Aurangzeb khan, Baharum Baharudin, Khairullah khan, “Efficient Feature Selection and Domain Relevance Term Weighting Method for Document Classification” Second International Conference on Computer Engineering and Applications, 2010 Crown Copyright DOI 10.1109/ICCEA.2010.228
- 2) T. Masuyama and H. Nakagawa , "Cascaded Feature Selection in SVMs Text Categorization" , LNCS 2588, pp. 588-591, 2003.
- 3) D. Lewis, "Feature selection and feature extraction for text categorization" . In Proceedings of a Workshop on Speech and Natural Language, 1992, pp. 212-217, San Mateo, CA: Morgan Kaufmann.
- 4) D. Lewis, "An evaluation of phrasal and clustered representations on a text categorization task" . In Croft et. al. (Ed.), Proceedings of SIGIR- 95, 15th CM International, 1992
- 5) Abdul Wahab, Aurangzeb Khan, Muhammad Faheem Khan And Shahid Khan, “Important Feature Extraction During Sentiment Analysis” Sci.Int(Lahore),26(2),959-964,2014
- 6) Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang- Rui Wang, and Chih-Jen Lin.2008. “Liblinear: A library for large linear classification”. Journal of Machine Learning Research, Volume 9, 6/1/2008, Pages 1871-1874
- 7) Linyun Fu, Haofen Wang, Haiping Zhu, Huajie Zhang, Yang Wang, and Yong Yu. 2007. “Making more Wikipedians: Facilitating semantics reuse for wikipedia authoring”. In ISWC/ASWC 2007, pp.128-141.
- 8) Evgeniy Gabrilovich and Shaul Markovitch. 2007. “Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization”. "Journal of Machine Learning Research, 8(Oct):2297-2345, 2007.
- 9) D. Merkl, "Text data mining". In: A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text. Marcel Dekker, New York, 1998, pp.396-207.