Palestine Polytechnic University

College of IT and Computer Engineering

Introduction to graduation projects

Project name:

Abstract Classification

Project team:

Iyas Salah, Yamen Wazwaz & Amro Amro

Supervisor: Dr. Hashem Tamimi

Abstract

As a result of the accumulation of unindexed research papers and public articles in libraries and research centers, a classification method is required to classify these research papers into their proper class.

Abstract classification takes an abstract as an input and outputs a classification (e.g., physics, math) that is the class to which the given paper belongs. This will be done using AI and machine learning to teach the model how to classify papers through their abstracts.

manually reading through documents is notoriously difficult for humans, as they have to go through all of these abstracts one by one to be able to give a decision for the suitable classes for each abstract, this process will take an unreasonable amount of time to have the results, regardless of the accuracy the human can give.

Many papers need to be categorized and arranged in libraries and other study facilities. A new approach must be created for classifying and organizing these articles that is both accurate and time-efficient.

This project is able to group papers into “categories” based on the relevant topics using a machine learning model that can classify documents, the results are presented to the user using a web application.

In conclusion, software that classifies abstracts will offer a simple and painless approach to indexing and arranging a collection of research publications.

Table of contents

Table of tables

Table of Figures

Chapter 1: Introduction

1.1 Introduction

Text classification, also known as text tagging or text categorization, can automatically analyze text and assign a set of predefined classes based on its content; the classes are based on the category taxonomy of ArXiv.com (ArXiv.com, 2023). Text classification is becoming an increasingly important part of businesses as it allows them to easily get insights from data and automate business processes, but first, we need to define the classes that we will use for the text classifier examples: physics, computer science, biology, math, engineering, and medicine.

The set of possible classes is inferred from the provided data, which may contain classes explicitly provided by the author of the paper corresponding to the abstracts being classified, if they are not available, manual labeling can be used.

Suppose we had a lot of articles or scientific papers and wanted to classify them based on topics. It would take a lot of effort and time to group the papers based on their topics. The use of computer algorithms will make it much easier and less time-consuming. While the human factor can give different judgments for papers affected by situations such as exhaustion, the computer algorithm will provide a more consistent result.

The model's input consists of an abstract in plain text, which it then analyzes. By enumerating each word and assigning it a unique number, similar words have the same serial number and are classified accordingly (bag-of-words approach).

The user will be able to enter an abstract to be classified through a web page, and they will also be able to enter multiple abstracts at once to be classified in batches.

There are many problems with text-comparison-based direct approaches, mainly determining the classes, and the relationships between abstract contents and their classes are made even more tedious when faced with tens of thousands of documents. Automating the detection and clustering of classes may reduce the workload by orders of magnitude; hence, we have shifted to an artificial intelligence-based approach that may mitigate these issues.

1.2 Project aim

Libraries and other research centers have large numbers of papers that have to be classified and ordered. This process can take a long time, so a new method must be developed for classifying and ordering these papers that is both accurate and time-efficient.

1.3 Objectives

- Design and develop a software application that classifies papers according to their appropriate topic.
- The application must be able to give a response promptly.
- Design and develop a web-based application to enable the user to submit an abstract to receive a suitable class for the given abstract.
- The application's results must be as accurate as possible to be reliable.

1.4 Who will benefit from this

- Libraries with books that require classification.
- Research centers that have a large number of scientific papers that need to be classified.
- Students research papers in universities.

1.5 Timeline

Table 1.1 Timeline

Number of tasks	Task name	Required time in weeks
1	Proposal	2
2	Information planning and collecting	2
3	Background and Alternatives	2
4	Determine system requirements	1
5	System requirements analysis and design	3
6	System programming and configure	8
7	System Test	3
8	System documentation	All project period

Chapter 2: Background

2.1 NLP

Natural language processing (NLP) is a branch of artificial intelligence concerned with giving computers the ability to understand text and spoken words in much the same way humans can (What Is NLP: An Introductory Tutorial to NL , 2023).

2.2 Machine learning

Machine learning is a branch of artificial intelligence that focuses on giving computers the ability to imitate human learning, gradually improving the accuracy of the machine based on input data.

It is possible to categorize machine learning based on the output into two distinct categories: classification and regression, where regression is for models that predict continuous properties, and classification is for models that predict discrete properties (Machine Learning: What It Is and Why It Matters, n.d.).

We are currently working on a classification problem using NLP. Several NLP classification algorithms have been developed to solve various NLP problems. For example, naive Bayes has been used in various spam detection algorithms, and support vector machines (SVM) have been used to classify texts such as progress notes at healthcare institutions.

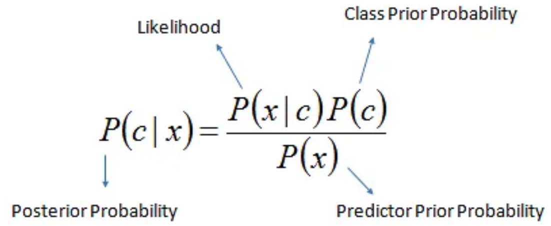
'Hyperparameters' are configuration options that affect the behavior of the model; for example, whether or not laplace smoothing is used in the Bayesian classifier, or the length of the N-grams used.

In this project, machine learning is used to solve a classification problem.

2.3 Naïve Bayesian Classification

A Bayesian classifier is a machine learning model based on Bayes' theorem of conditional probability. given an input vector, it returns the probability of it being of class C_k, more formally, given a vector, $V = \{x_0, x_1 \ldots x_n\}$, $P(C_k \mid X_0=x_0, X_1=x_1 \ldots X_n=x_1)$ where C_k is a possible category for that vector, for the initial test, we are only concerned with a single class (Team, 2020).

Equation 2.1 Bayes theorem equation


Likelihood
Class Prior Probability
$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

The naïve Bayes method is commonly used in statistical analysis for document classification. It is simple and fast. Compared to the much more computationally expensive AI-based semantic analysis, or NLP (Natural Language Processing).

Classification involves assigning fragments of text (documents) to classes based on the probability that they belong to the same class.

Each document contains multiple words (terms), which contribute to an understanding of its content. which eventually leads to assigning that document to a class.

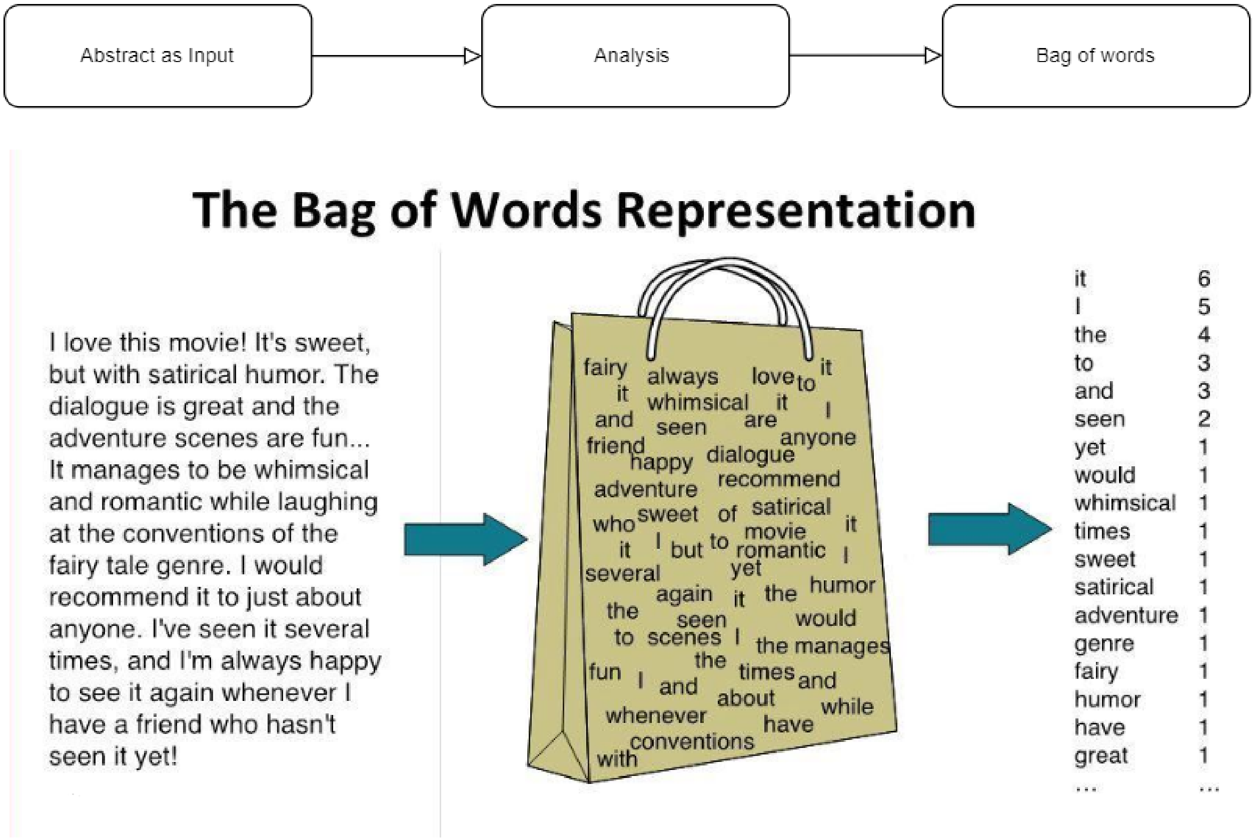
A problem that arises when using this method of classification is how to deal with new words that the model has not been trained on, because in such a case, the prior probability would be zero. For example, if a model was not trained on data that contains the word “apple”, then for the input “I ate an apple”, the prior probability would be $P(\ldots|X_i = 'Apple') = 0$, because it has never encountered that word before, and thus the probability of it appearing is zero based on the statistical knowledge of that model, resulting in a division by zero error.

This is solved using Laplace smoothing, in which we assume that each word has appeared an extra K times, more formally $P_{Laplace,\,K}\frac{c(x)+K}{N+k|X|}$, meaning that in the case where a word has never been in the training data before, it is assumed to have appeared at least K times, thus eliminating the zero-division.

2.4 Bag of Words Approach

The bag of words approach is a method for simplifying NLP (shown in figure 2.1), maintaining only the multiplicity of the elements, however, this does have the disadvantage of removing any information regarding the positions of words (Devin Almonor, n.d.).

Figure 2.1 The bag of words representation



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

The n-gram model can help encode and represent sequences, allowing it to retain some of the order of the original text; this is done by taking every *n* consecutive words together as one word, for example, “This is a sentence” with a two-gram model becomes {“This is”, “is a”, “a sentence”}.

Alternative approaches include Word2Vec and BERT, which are more advanced than that bag of words.

2.5 TF-IDF

TF-IDF (term frequency–inverse document frequency) is a product of two statistics, term frequency, and inverse document frequency. It is considered a term weighting scheme, it is given to words to help us statistically determine how important a word is in a document.

TF-IDF considers two factors in a word, the first is the frequency in which that word occurs, and the second is an inverse document factor that reduces the weight of common words that occur too frequently in a document, such as the words "the" or "and." For example, if we have a group of documents that we want to rank to decide which document is more relevant to the query (the wild cat), the weight of the word"the" is reduced by the inverse document factor because it is a ubiquitous and unmeaningful word in all the documents, while on the other hand, the words “wild" and "cat" are given more weight because they are less common and more meaningful (Borcan, 2020).

Term frequency equation:

Equation 2.2 TF equation

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

In equation 2.2, the numerator is the count of the term t in document d. And the denominator is simply the total number of terms in document d.

Inverse document frequency equation:

Equation 2.3 IDF equation

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

In equation 2.3, the numerator N is the number of documents, and the denominator is the number of documents D in which the term t appears.

Thus, the tf-idf equation is

Equation 2.4 TF-IDF equation

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

2.6 Tokenization and Analysis

Tokenization refers to the process of translating plain text into "tokens", tokens being the building blocks of natural and artificial languages. Typically, a token is a word in a series of words in an n-gram model, during tokenization (Chakravarthy, 2020).

‘Stop words’ carry little or no meaning and exist in languages for grammatical purposes, such as “the” or “a” in English. Such stop words are ignored during tokenization because they provide little information about the text at hand while adding a lot of noise and greatly increasing the size of the data.

Before tokenization, the text is analyzed for the sake of simplification; the case of the characters is ignored (all characters are converted to lowercase); the words are then stemmed to their roots, for example, "goes," "went,” and “going" becomes "go," which helps reduce the size of the bag.

2.7 Cross-Validation

Cross-validation is a set of techniques for evaluating how the results of a statistical model will generalize to a larger population using a multitude of metrics (3.1. Cross-validation: Evaluating Estimator Performance, n.d.), in our case, we are using the F1-score, accuracy, precision, and recall.

For cross-validation, we split the input dataset into a training set and a test set, with the test set being 20%-30% of the original set and the training set using the rest of the samples, which are used to fit the model. The test set is then used to generate predictions using the resultant model from the training set, and the performance metrics are then calculated based on the results.

A third set, called the 'validation set' may also be used to help fine-tune the hyperparameters of the model. It is used to measure the performance of the model and how it reacts to any adjustments made to the hyperparameters. It is not the same as the test set because it is used to help find the best parameters for the model, resulting in some bias for that dataset. As such, a third dataset is needed to help objectively measure how well the model generalizes.

2.7.1 Precision and accuracy

Precision is a metric used in evaluating the performance of statistical models, Precision is calculated by dividing the number of correctly classified documents by the total number of documents with respect to a certain class, or more formally as shown in equation 2.5:

Equation 2.5 Precision equation

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

where tp is the true positive rate and fp is the false positive rate.

Accuray meanwhile is the total number of correct classifications divided by the total number of classified documents, more formally as shown in equation 2.6:

Equation 2.6 Accuracy equation

$$Accuracy \;=\; \frac{tp+tn}{tn+fn+fp+tp}$$

2.7.2 Recall

To understand recall let's first consider an example of recall, suppose that we have a data set that consists of 100 emails, 40 of which are spam. The goal here for the model is to classify them as either spam or not spam. The model classified 30 emails as spam, and out of those 30 emails, 25 were spam (True positives) and 5 were not spam (false positives) which means that there are 15 spam emails that the model failed to classify as spam (false negatives), now to calculate the recall for the model we use true positives (TP) and false negatives(FN) as shown in equation 2.7.

Equation 2.7 Recall equation

$$Recall \;=TP\,/\,(TP \;+\; FN)$$
$$Recall \;=\; 25\,/\,(25\,+\,15) \;=\; 0.625$$

Which concludes that the recall for this model is 62.5%, which means that out of all positive cases in the data set, the model has correctly identified 62.5% of them.

So a higher recall for the model means that it's better at capturing positive instances, but keep in mind that recall does not account for false positive instances, which is why other metrics are used alongside recall such as precision, accuracy, and F-scores to assess the overall effectiveness of a classification model.

2.7.3 F-Scores

F-scores are a metric used to evaluate the performance of statistical models; they combine two important metrics that we previously talked about, which are recall and precision, into a single metric, providing a balanced measure of the models' performance, the F1-score is an F-scores that balances precision and recall equally, and it is often used when both metrics are considered equally important, It is calculated as the harmonic mean between precision and recall as shown in equation 2.8.

Equation 2.8 F1-score equation

$$F1 - score = \frac{TP}{TP + 0.5 * (FN + FP)}$$

2.7.4 ROC Curve and AUC

ROC Curve (Receiver Operating Characteristic Curve):

The ROC curve is a graphical representation of the performance of a binary classification model. It plots the True Positive Rate (TPR) on the y-axis against the False Positive Rate (FPR) on the x-axis at various classification thresholds. The curve helps us assess the trade-off between sensitivity (recall) and specificity of a model across different threshold settings.

AUC (Area Under the ROC Curve):

The AUC is a metric that quantifies the overall performance of a binary classification model using the ROC curve. It represents the area under the ROC curve and provides a single value to measure the model's ability to distinguish between positive and negative samples. The AUC ranges between 0 and 1, where a higher value indicates better discrimination power of the model.

When evaluating a binary classification model, we often encounter a trade-off between correctly identifying positive cases (sensitivity) and correctly identifying negative cases (specificity). The

ROC curve helps us visualize this trade-off. It is created by plotting the TPR (also known as sensitivity or recall) against the FPR. TPR measures the proportion of actual positive cases correctly classified as positive by the model. It is calculated as the number of true positives divided by the sum of true positives and false negatives. In simpler terms, it tells us how good the model is at identifying positive cases.

FPR, on the other hand, measures the proportion of actual negative cases incorrectly classified as positive by the model. It is calculated as the number of false positives divided by the sum of false positives and true negatives. In simpler terms, it tells us how often the model mistakenly identifies negative cases as positive. The ROC curve allows us to assess the model's performance across different classification thresholds. By adjusting the threshold, we can prioritize sensitivity or specificity based on the specific requirements of our problem.

The AUC complements the ROC curve by providing a single numerical value to represent the model's overall performance. It quantifies the probability that a randomly chosen positive sample will be ranked higher by the model than a randomly chosen negative sample. An AUC of 1 indicates a perfect model, while an AUC of 0.5 suggests the model performs no better than random guessing.

2.9 Python

Python is a general-purpose programming language. Its high-level data structures, dynamic typing, and many other features make it equally useful for scripting or "glue code" that ties components together as they do for developing complex applications. Additionally, it can be enhanced to execute code written in languages like C, C++, or Rust and make system calls on almost all operating systems. It is especially useful in data science and machine learning thanks to its concise syntax, interpreted nature, and extensive library collection.

2.9.1 Numpy

Numpy is a Python library for manipulating multidimensional numeric data, Most relevant here is its proficiency at matrix operations.

2.9.2 SKLearn

SKlearn is a machine learning library written in C++ and controlled using Python. It implements many classical machine learning algorithms, both for educational and professional purposes.

2.9.3 NLTK

NLTK (short for Natural Language Processing Toolkit) is a popular NLP library for Python. It offers features such as word stemming and part-of-speech detection (NLTK :: Natural Language Toolkit, n.d.); for example, it can return the words “went” and “going” to "go" their root, it can also detect if a word is being used as a verb or noun, like in “He *ducked* under the table” and “I ate *duck* soup yesterday,” where the word “duck” is used as a verb and a noun in each sentence, respectively.

2.9.4 PyMuPDF

MuPDF is a library written in C for processing PDF files, it allows you to parse and extract text from PDF files, PyMuPDF is a Python library that implements a wrapper around MuPDF that allows it to be used in Python programs, we use it in conjunction with regular expressions to extract the abstract from a paper in PDF file format.

2.9.5 FastAPI

FastAPI is a python backend framework that can be used to build RESTFul and GraphQL APIs, it has built in support for authentication and authorization, and validation based on user defined schema.

2.10 One-Versus-Rest Classification

One-Versus-Rest Classification, OvR for short, is a technique that allows one to create a multi-label (multiple outputs per input) or multi-class (multiple possible different outputs, for example, “Math”, “Biology”, or “Physics” instead of just Yes/No classification) classifier out of a binary classifier, which is a classifier that outputs only 1 and 0 to indicate a “yes” or a “no”, in OvR, we create a separate binary classifier for each different class, and it compares the samples that have that are labeled with that class with the samples that do not are labeled with that class, hence the name, one versus rest, which allows you to find whether or not an abstract belongs to a specific class on a per class basis.

However, this comes at the cost of having to train an extra classifier for each class you need, which is why it's preferable to have fewer classes when using this technique, which also allows higher accuracy due per class.

Chapter 3: System Analysis and Design

3.1 Introduction

In this chapter, we will explore the design of our system, which comprises three sections: the UI design section, where we examine the user interface and its elements; the model design section, which walks through our model learning process; and the database design section, which provides an overview of our database.

Additionally, we will define the system requirements, encompassing both functional and non-functional aspects, and clarify them using use cases.

3.2 Requirements

3.2.1 Functional Requirement

1. The system must be realized as a website.
2. The website must allow users to submit an abstract to be classified, which can be submitted as a single abstract in the interactive input page or as a collection of PDF files in a batch processing page, using a trained classifier that can be updated regularly.
3. The website must provide admins with a statistics page, a page to register more admins and a page to rename classes.
4. The website should include a filtering option that allows users to view abstracts specific to a particular class, also it must allow the user to specify a threshold to display the classes that exceed it, in addition the results page must support a search by text in the batch abstracts results.
5. The website should output the correct classes for each abstract alongside the probability for each class in the results page, alongside a filtering option that allows the user to view abstracts specific to a particular class, and a text search operation.
6. The software must be able to maintain a large number of abstracts in JSON format in a database.

3.2.2 Non-functional Requirement

1. To be reliable, application results must have an AUC score of at least 90%.
2. A single PDF file (single paper) must be processed by the system in less than one second.
3. Direct database registration is required for the system root admin, and at API startup, the label mapper must be filled in. At first, the displayed name will be the same as the internal name; after that, the administrator can specify displayed names.
4. The user interface must support dark and light themes.

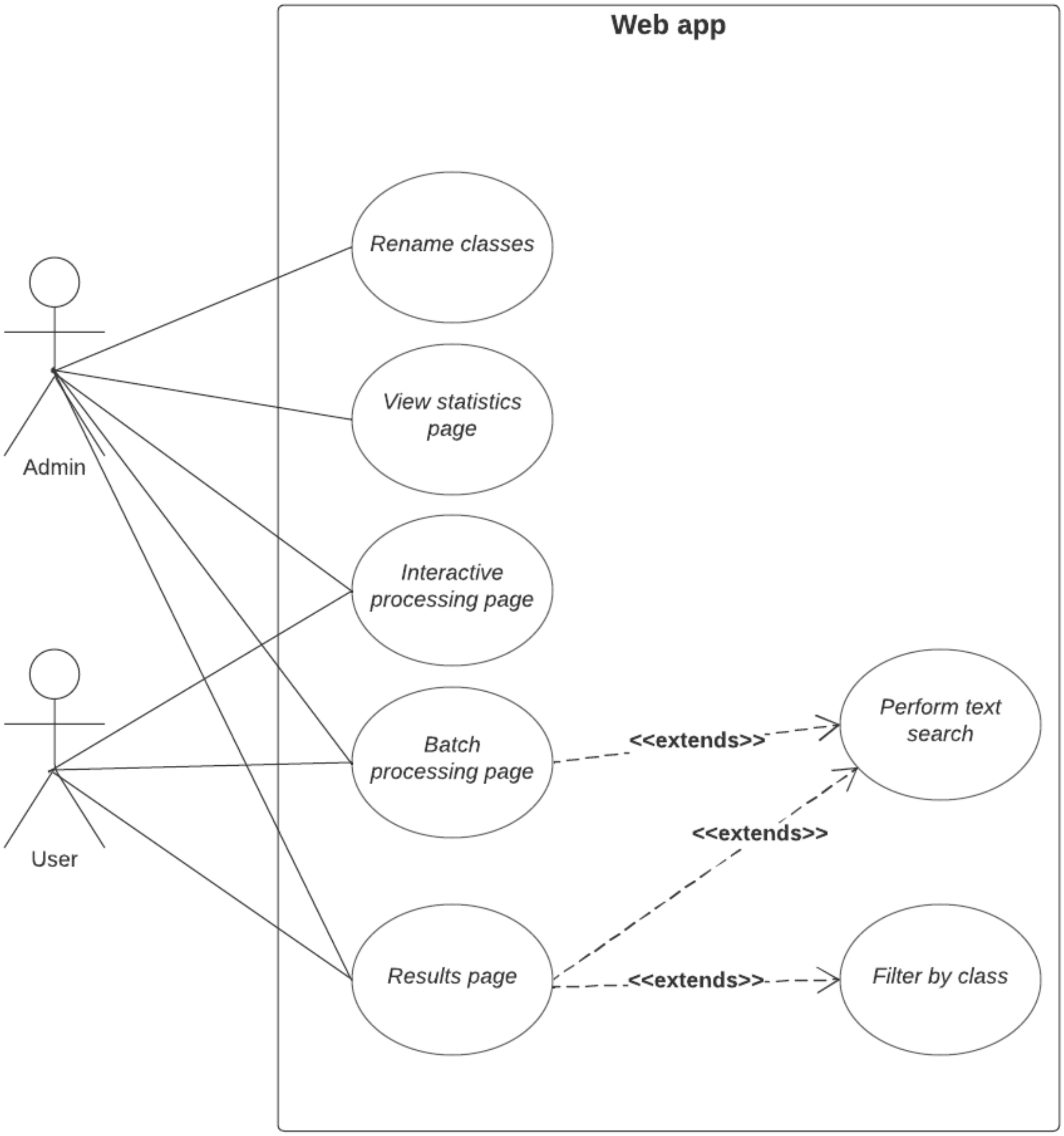
3.2 Analysis and use cases

3.2.1 Analysis

The figure 3.1 illustrates the key actors in our system, namely the administrators and users who interact with the web application. The user can use the interactive processing page to submit a single abstract or the batch processing page to submit a collection of pdf files. They can also view the results page to see the classification results after submitting a batch of PDF files on the batch processing page. They can also filter the results by class or by performing full-text search.

In addition to all the actions a user can perform, the administrator can perform extra actions such as renaming classes and viewing the statistics page.

Figure 3.1 admin and user use case diagram



3.2.2 Use Case 1: interactive processing page use case

Introduction: the actor in this use case can either be the administrator or the user, in this use case the actor is in the interactive processing page to submit an abstract for classification.

Use case description: the actor will enter their abstract in a textbox and submit it which sends the abstract from the web application(client) to the API, the api will send the abstract to the model for classification, the API will also insert the abstract into the database, then the model will return the results to the API which return the classes to the client and displays them for the actor.

User interactions:

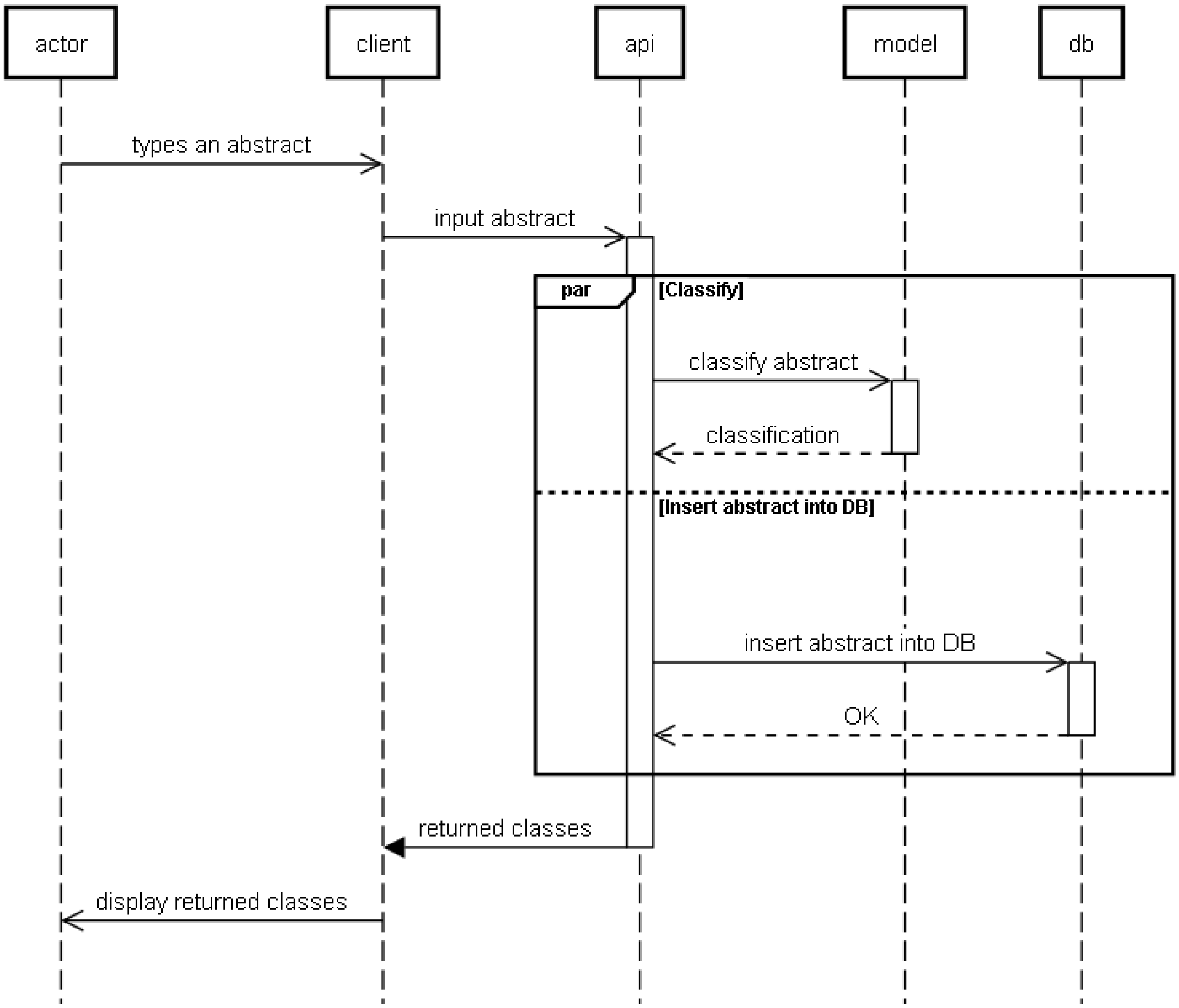
1. Actors are prompted to enter the abstract they want to classify. They can input the abstract into the provided text area within the component.

2. After entering the abstract the actor clicks on the classify button which sends the abstract for the back-end to be classified by the model.

3. The web application receives the response from the API, containing the predicted classes and displays them for the user.

These interactions are shown in the figure 3.2.

Figure 3.2 interactive use case sequence diagram



3.2.3 Use Case 2: Batch processing page use case

Introduction: The actor in this use case can either be the administrator or the user; in this use case, the actor is in the Batch processing page to submit a collection of PDF files for classification.

Use case description: the actor will upload the PDF files containing the abstracts he wishes to classify, each pdf file that finished uploading will be sent to the API which extracts the abstracts from the pdf file and send them to the parser, the API will also send the abstract to the database, then when the parser is finished the abstract is sent to the model for classification, after the model classifies the abstract it will send the results back to the API which sends them to the results page in the web application.

User interactions:

1. Actors are prompted to upload a collection of PDF files containing the abstracts they want to classify.

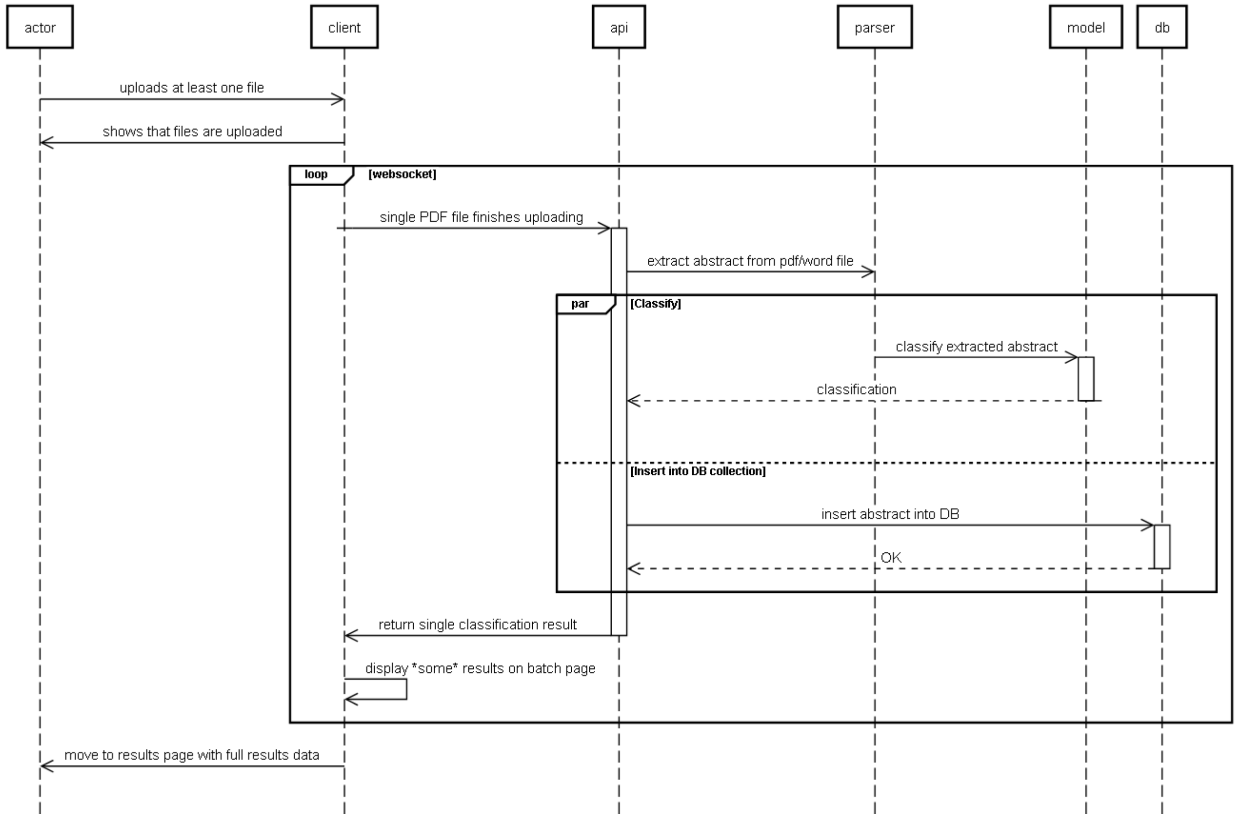
2. After uploading the user will be able to see the uploaded files in a table, they can see the state of the file (uploading, uploaded, error), remove the file, upload more files, or move to the results page if they are done uploading files.

3. When a file is successfully uploaded the web application front-end will send it directly to the API to be classified by the model then the result is directly returned to the web application and is then displayed on the result page.

4. When a user clicks on the "see results" button they will be taken to the result page which already contains the classification result for the uploaded PDFs.

These interactions are shown in figure 3.3.

Figure 3.3 Batch use case sequence diagram



actor
client
api
parser
model
db
uploads at least one file
shows that files are uploaded
loop
[websocket]
single PDF file finishes uploading
extract abstract from pdf/word file
par
[Classify]
classify extracted abstract
classification
[Insert into DB collection]
insert abstract into DB
OK
return single classification result
display *some* results on batch page
move to results page with full results data

3.3 Model design

3.3.1 Normalization

During this step in our process, the input will be stripped out of punctuation symbols and the letters will be converted to lowercase or uppercase, the point being that letter cases should be ignored as they are largely irrelevant. Also during this step, all words will be returned to their original root, for example “went”, “going” and “goes” will all be returned to their root, which is “go”.

Following stemming, words with direct synonyms are replaced with said synonyms, for example, “large” will be replaced with “big”.

The tokenization process also separates tokens that may appear to be a single word if we only consider spacing, for example, “3cm” will become {“3”, “cm”} when tokenized.

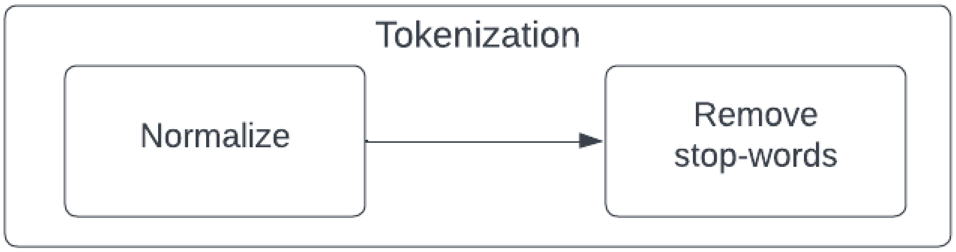
“I go to work at 8 a.m.” becomes {“i”, “go”, “to”, “work”,”at”, “8”,”am”}.

3.3.2 Remove stop-words

The output of the previous step will be processed to remove stop words which are words that carry little or no meaning, and exist in the language for a grammatical purpose like “the”, “or”,”and”,”is”, “are” and so on, these words will be ignored because they provide little information about the text and adds a lot of noise that increases with increasing the data size

Continuing from the example in section 3.3.1, {“i”, “go”, “to”, “work”,”at”, “8”,”am”} becomes {“i”, “go”, “to”, “work”, “8”, ”am”} and these two steps can be show in Figure 3.4.

Figure 3.4 Tokenization



3.3.3 Generate N-grams

During this step, the given text will be divided into blocks (N-grams). Unigrams are composed of one word; for example, the sentence “this is a sentence” will become “this”, “is”, “a”, “sentence”. Bigrams are composed of two words; if we take the same previous example, it will be “this is”, “is a” and ”a sentence”. Trigrams are composed of three words, if we take the same example as before, it will be “this is a” and “is a sentence”.

Continuing the example seen in section 3.2, if we convert the result to a 2-Gram form, {“i”, “go”, “to”, “work”, “8”,”am”} becomes {“i go”, “go to”, “to work”, “work 8”,”8 am”}

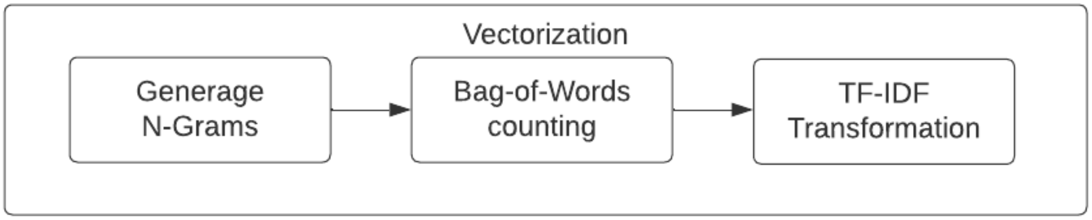
3.3.4 Bag-of-words counting

During this step, each word will be counted for how many times it appeared in the text; for example, the bag-of-words representation of the sentence “I like to watch movies, and you like watching movies too.”, will contain the word “like” with the number two because it was mentioned twice, and it will have the word “too” with the number one because it was mentioned only once in the sentence. If we continue the example seen in 3.3, the result would be that each 2-gram appears only once.

3.3.5 TF-IDF

During the TF-IDF transformation step, this formula is applied to the frequency input vector using the formula discussed previously, and we can see the last three steps, which we called *Vectorization* in figure 3.5.

Figure 3.5 Vectorization

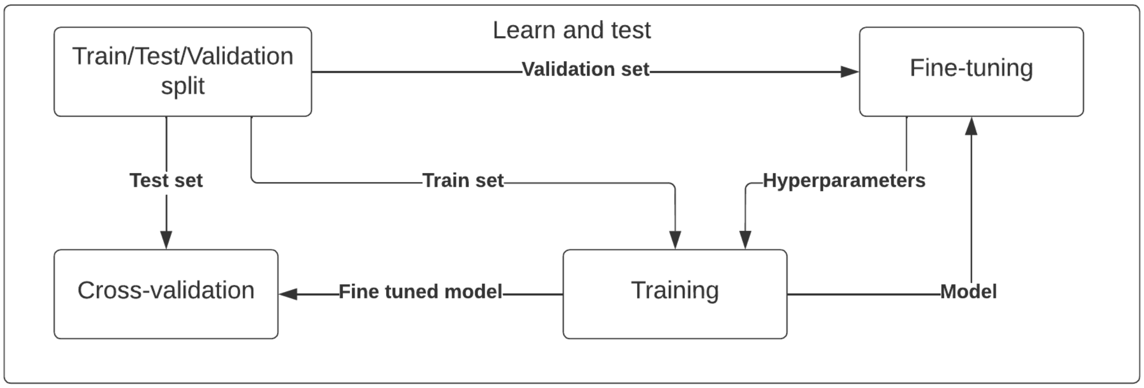


3.3.6 Learn and predict

During this step, the model is fitted using the data from the training set. The data from the validation set is used to finetune the hyperparameters, such as N-gram length and whether or not TF-IDF is used.

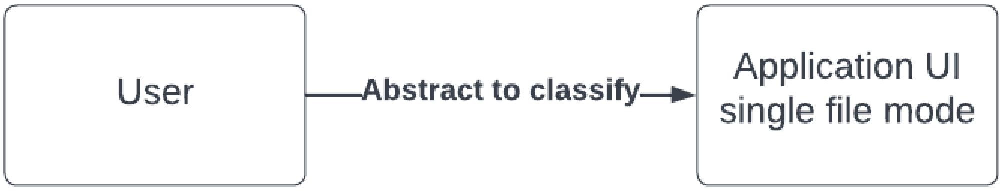
After the model is fully trained and fine-tuned, cross-validation begins, where the test set is used to evaluate the model using the previously disclosed measures, including but not limited to F1-Scores, accuracy, recall, and precision. This process can be seen in Figure 3.6.

Figure 3.6 Learn and test



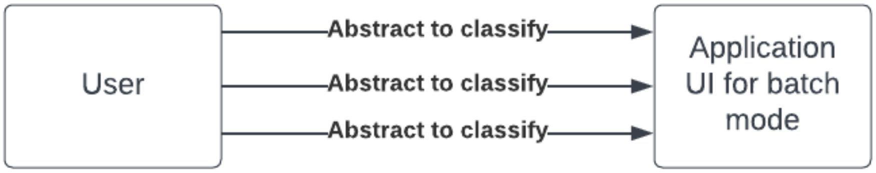
As we said before in the requirements, the system must be able to classify abstracts in two ways. The first one will allow the user to submit a single abstract to classify it, as we can see in figure 3.7.

Figure 3.7 Single Abstract/Interactive Mode



The second mode is batch mode, where we can submit multiple abstracts to classify them, as shown in Figure 3.8.

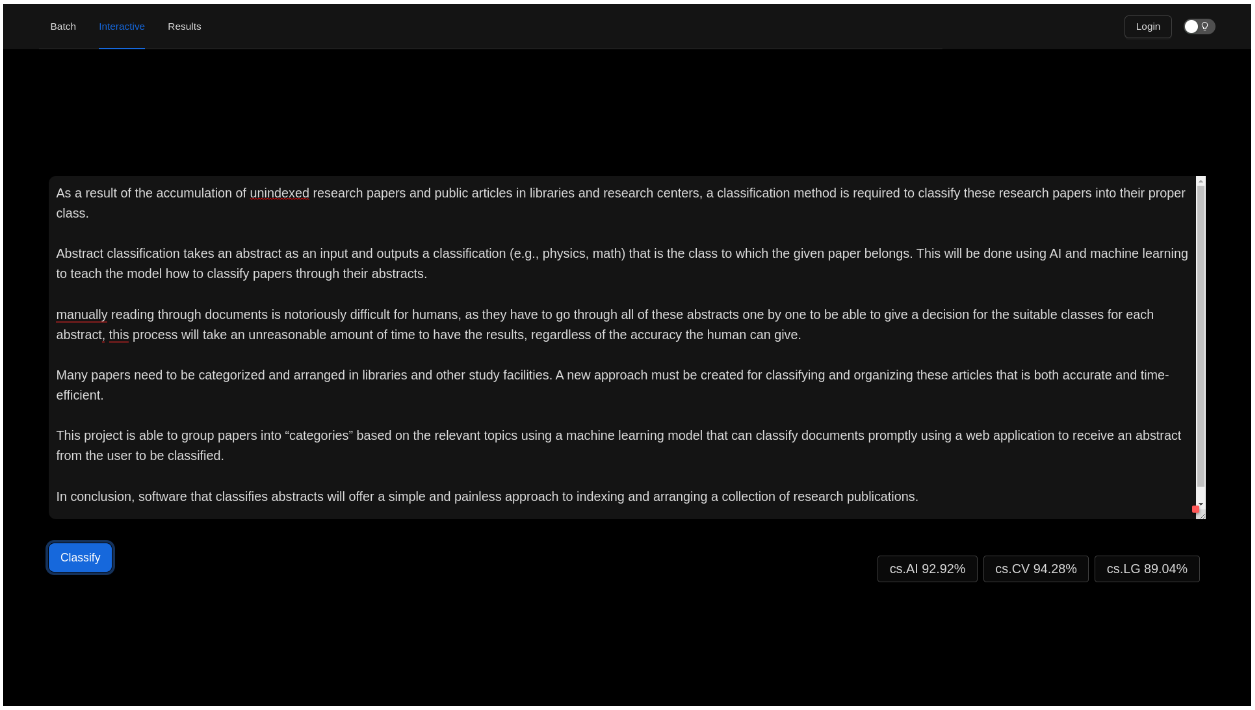
Figure 3.8 Batch mode



3.4 UI design

The interface shown in figure 3.9, demonstrates the interactive processing page. Here the user can type the abstract in the text box area and submit, which displays the results as shown.

Figure 3.9 Interactive Processing Interface



As a result of the accumulation of unindexed research papers and public articles in libraries and research centers, a classification method is required to classify these research papers into their proper class.

Abstract classification takes an abstract as an input and outputs a classification (e.g., physics, math) that is the class to which the given paper belongs. This will be done using AI and machine learning to teach the model how to classify papers through their abstracts.

manually reading through documents is notoriously difficult for humans, as they have to go through all of these abstracts one by one to be able to give a decision for the suitable classes for each abstract, this process will take an unreasonable amount of time to have the results, regardless of the accuracy the human can give.

Many papers need to be categorized and arranged in libraries and other study facilities. A new approach must be created for classifying and organizing these articles that is both accurate and time-efficient.

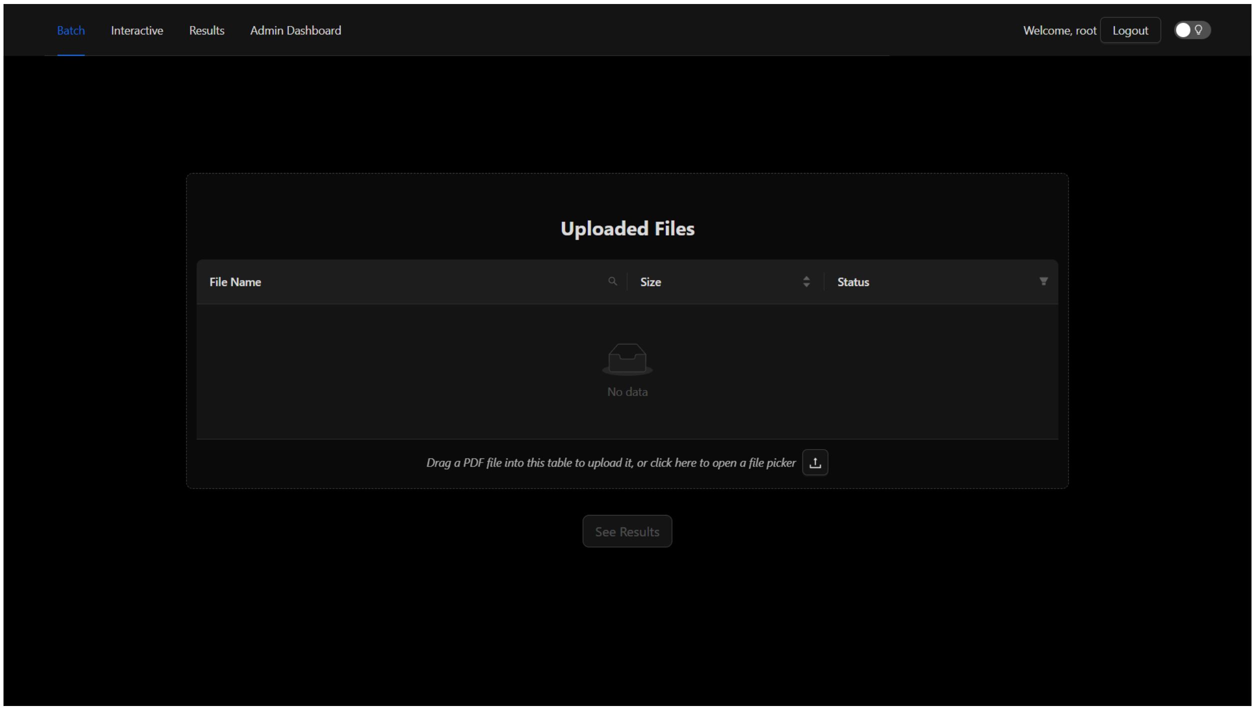
This project is able to group papers into "categories" based on the relevant topics using a machine learning model that can classify documents promptly using a web application to receive an abstract from the user to be classified.

In conclusion, software that classifies abstracts will offer a simple and painless approach to indexing and arranging a collection of research publications.

cs.AI 92.92% cs.CV 94.28% cs.LG 89.04%

In the interface shown in Figure 3.10, the user can upload a file through the upload file button to pick a pdf file or files containing the abstracts he wants to classify, and then they are directed to the results page where they can see the classification results.

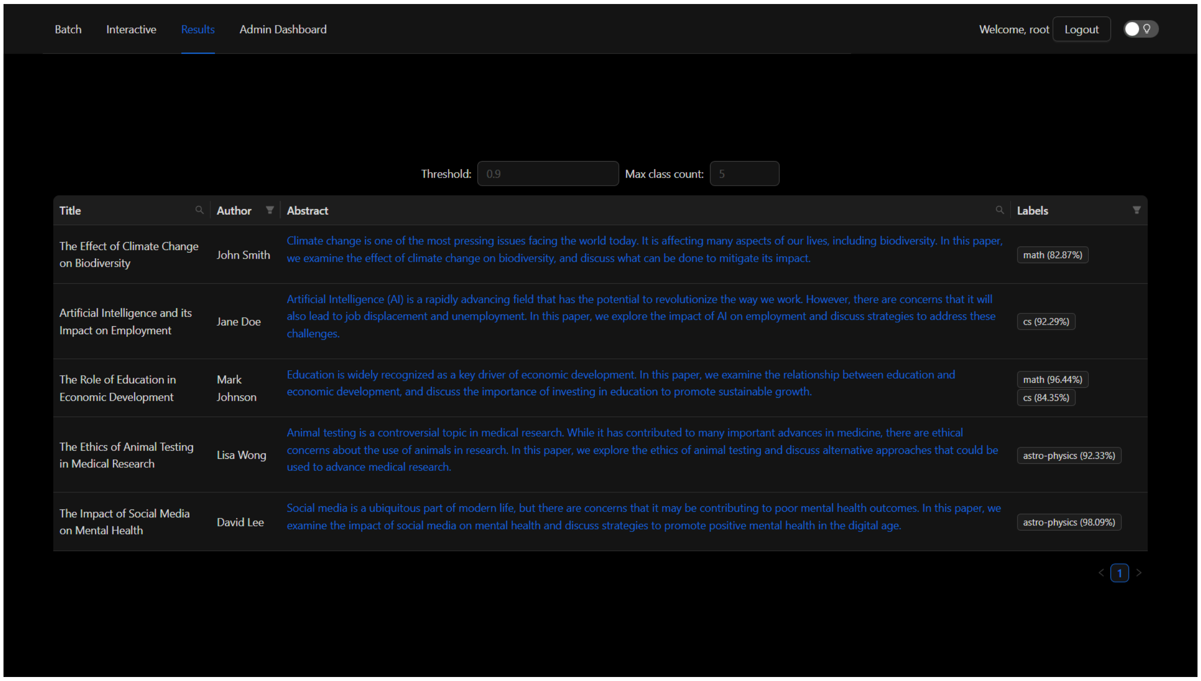
Figure 3.10 Batch processing interface



Drag a PDF file into this table to upload it, or click here to open a file picker

In the interface shown in Figure 3.11, the user can see the abstracts he uploaded alongside the classification results for each one. They are also offered the option to filter the abstracts according to their classes or perform text search on the abstracts.

Figure 3.11 Results interface

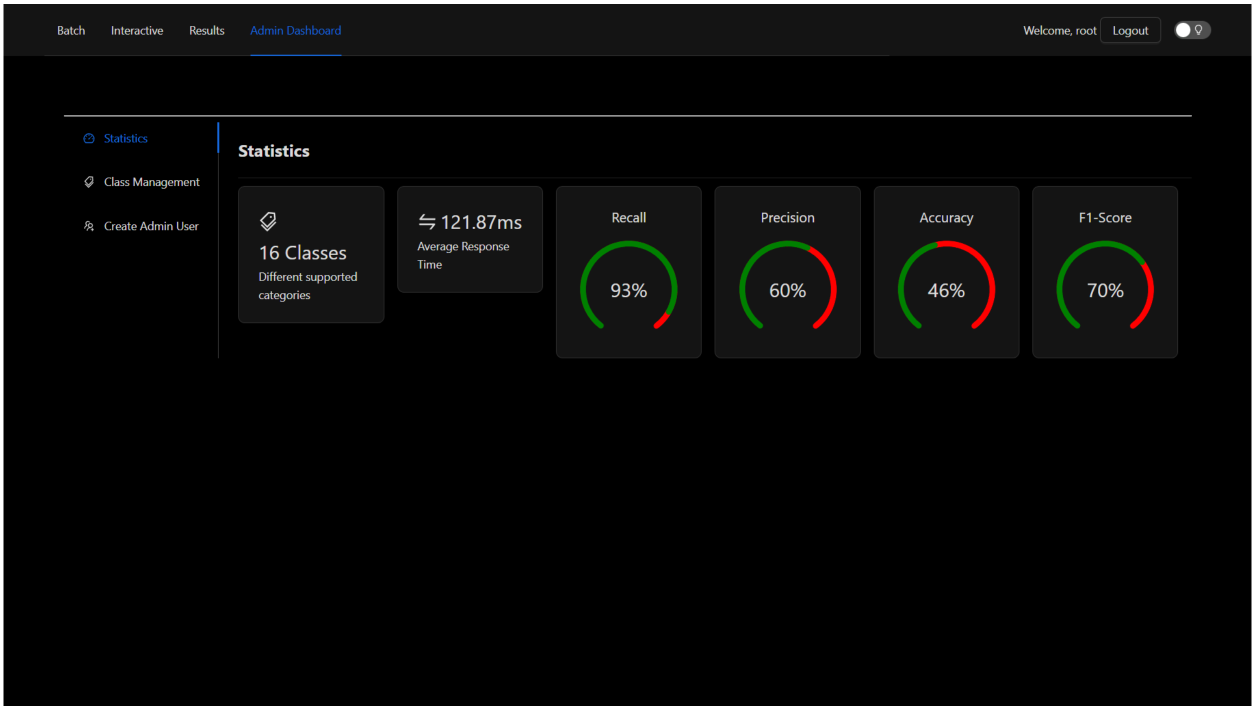


Threshold: 0.9 Max class count: 5

Title	Author	Abstract	Labels
The Effect of Climate Change on Biodiversity	John Smith	Climate change is one of the most pressing issues facing the world today. It is affecting many aspects of our lives, including biodiversity. In this paper, we examine the effect of climate change on biodiversity, and discuss what can be done to mitigate its impact.	math (82.87%)
Artificial Intelligence and its Impact on Employment	Jane Doe	Artificial Intelligence (AI) is a rapidly advancing field that has the potential to revolutionize the way we work. However, there are concerns that it will also lead to job displacement and unemployment. In this paper, we explore the impact of AI on employment and discuss strategies to address these challenges.	cs (92.29%)
The Role of Education in Economic Development	Mark Johnson	Education is widely recognized as a key driver of economic development. In this paper, we examine the relationship between education and economic development, and discuss the importance of investing in education to promote sustainable growth.	math (96.44%) cs (84.35%)
The Ethics of Animal Testing in Medical Research	Lisa Wong	Animal testing is a controversial topic in medical research. While it has contributed to many important advances in medicine, there are ethical concerns about the use of animals in research. In this paper, we explore the ethics of animal testing and discuss alternative approaches that could be used to advance medical research.	astro-physics (92.33%)
The Impact of Social Media on Mental Health	David Lee	Social media is a ubiquitous part of modern life, but there are concerns that it may be contributing to poor mental health outcomes. In this paper, we examine the impact of social media on mental health and discuss strategies to promote positive mental health in the digital age.	astro-physics (98.09%)

The administrator dashboard contains three components: the first is statistics which displays some metrics statistics for the model as shown in figure 3.12.

Figure 3.12 statistics interface



Statistics

16 Classes
Different supported categories

121.87ms
Average Response Time

Recall
93%

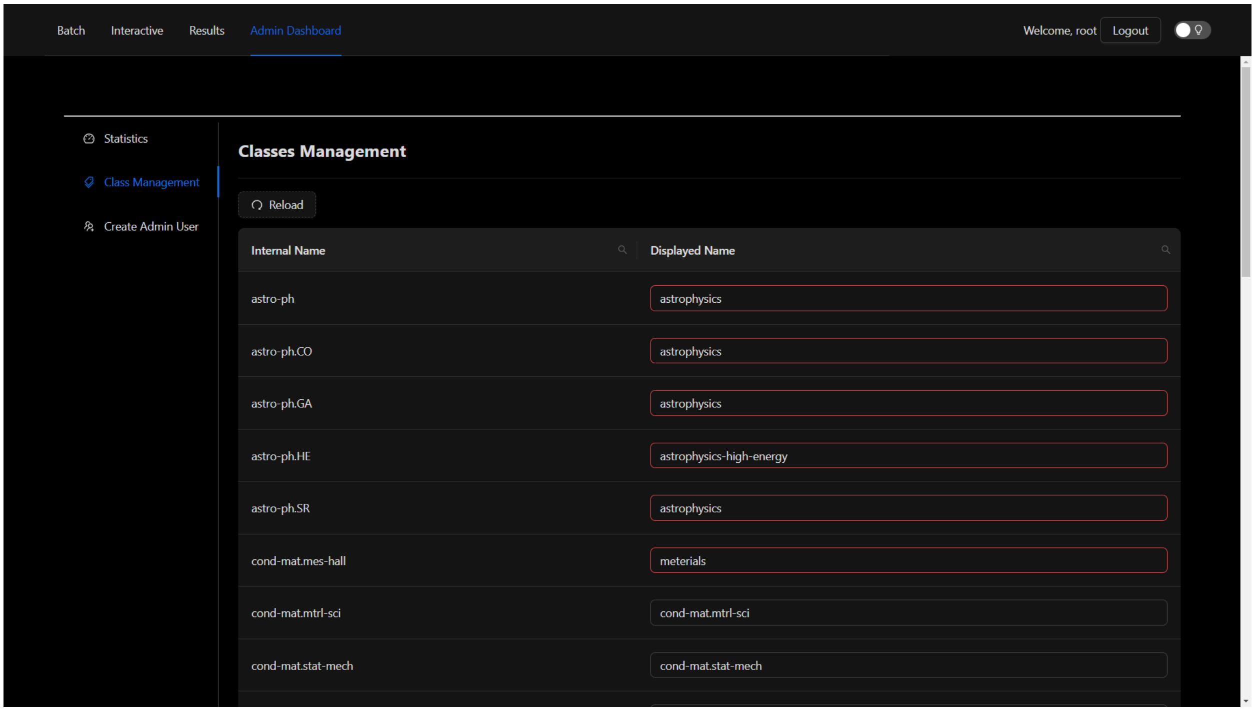
Precision
60%

Accuracy
46%

F1-Score
70%

The second component in the administrator dashboard is the class management component, here the administrator can change the name to be displayed for a certain class, as shown in figure 3.13.

Figure 3.13 class management interface

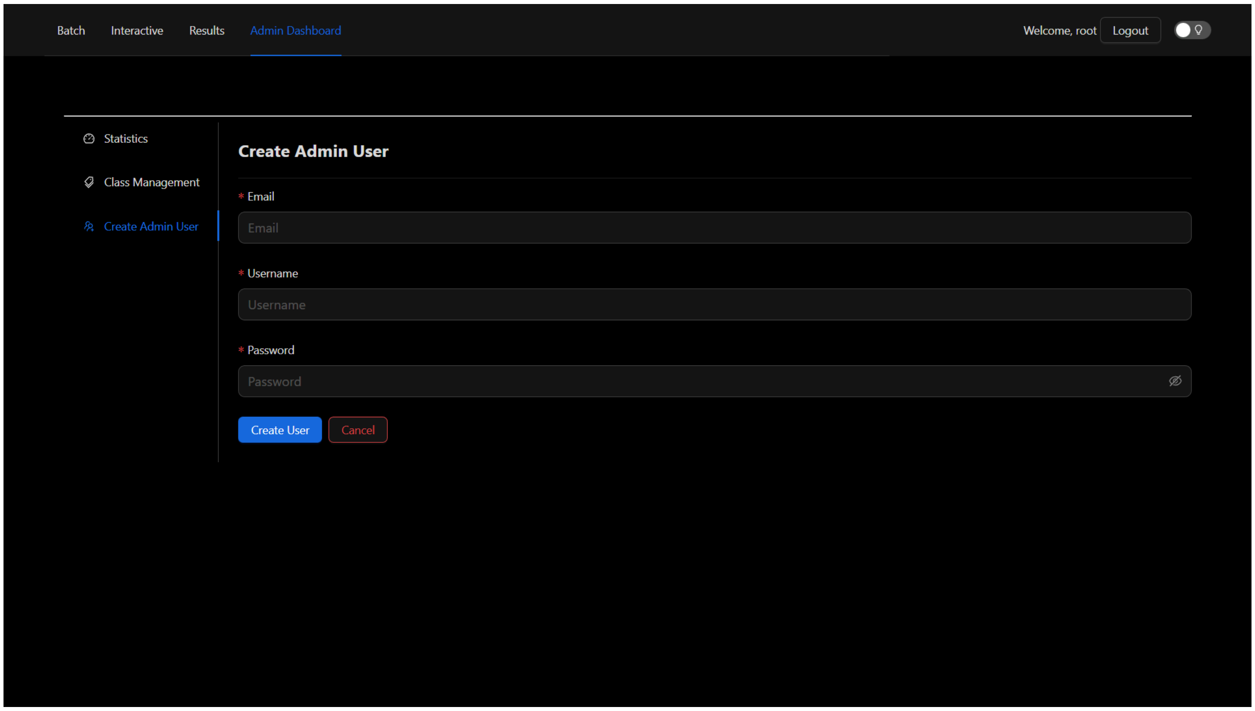


Classes Management

Internal Name	Displayed Name
astro-ph	astrophysics
astro-ph.CO	astrophysics
astro-ph.GA	astrophysics
astro-ph.HE	astrophysics-high-energy
astro-ph.SR	astrophysics
cond-mat.mes-hall	meterials
cond-mat.mtrl-sci	cond-mat.mtrl-sci
cond-mat.stat-mech	cond-mat.stat-mech

The last component in the administrator dashboard is the create admin user component, which allows the administrator to add a user administrator by typing in the email address and password for that admin as shown in figure 3.14.

Figure 3.14 create admin user interface



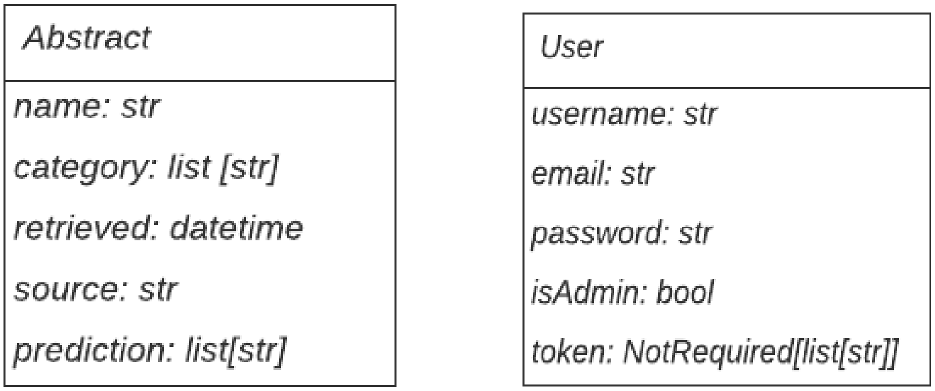
3.5 Database Design

The database used for the application will be a document-based non-relational database,so there is no restriction on field size. We used MongoDB, in which documents are stored in the BSON format, which is based on the JSON format. It supports the nesting of objects, automatic ID-assignment, and unstructured data; the database shall store abstracts in a plain text format. The datasets were retrieved directly from an online website called Kaggle.com that specializes in collecting raw datasets.

In figure 3.15, we can see the abstract and user object classes. The abstract object class stores information about the abstracts, such as the name of the abstract, a list of classes, the retrieved date time of the abstract, the source of the abstract, and the predicted classes for the abstract.

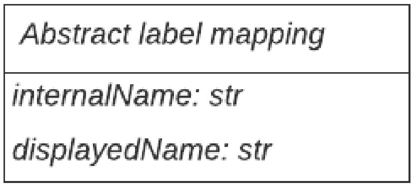
The user object class holds information concerning the user, such as user name, email of the user, hashed password, a boolean for admin status, and the token for the user.

Figure 3.15 Database Schema (non-relational)



In figure 3.16, we can see the abstract label mapping object class. The abstract label mapping object class stores the internal name and the displayed name, the internal name is the name that the model uses internally, and the displayed name will replace the internal name in the web application and can be edited by admin users.

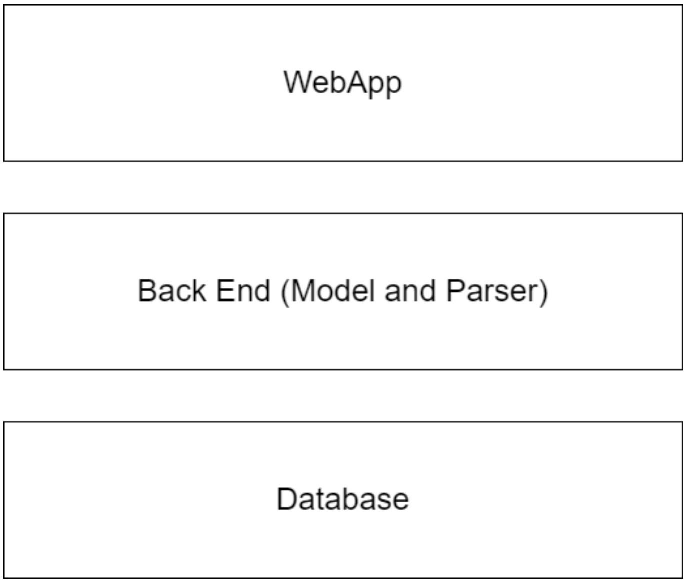
Figure 3.16 Database Schema (non-relational)



3.6 Software Architecture

The layered architecture consists of a web application that the client interacts with, a backend for handling API requests made by the client application (using RESTFul), and a database that is used to store data for the backend. Figure 3.17 illustrates the architectural model used for the system.

Figure 3.17 Architectural Model



Chapter 4: Testing

4.1 Data Cleaning

The most relevant classes were selected based on their representation in the dataset, the classes that appeared in more than 10% of the dataset were deemed reliable

4.2 Performance

Thanks to the simplistic nature of Naive bayes classification, it only only a few minutes to train a model on our dataset of 2.2 million abstracts, and parsing a PDF file with PyMudPDF and classifying it with the model one abstract using the model takes varied amounts of time based on the PDF file in question, the lowest amount was 80ms per classification, and the highest recorded was 521ms.

4.3 Cross-Validation Results

We used 5-fold cross-validation to calculate the precision, recall, F1-Score, and accuracy, the results were 60%, 92%, 70%, and 46% respectively, we are most interested in recall here as it best demonstrates how likely the model is to retrieve a specific document based on a search query, weighted averages were used to calculate the scores, which are calculated by multiplying the score for every individual class by the percentage of the samples it appears in.

The deficits in accuracy and precision can be compensated for by adjusting the threshold, this can be done on the client side by the user, the low accuracy is to be expected from a multilabel classifier given how many labels there are to test.

4.4 Hyperparameter Fine Tuning

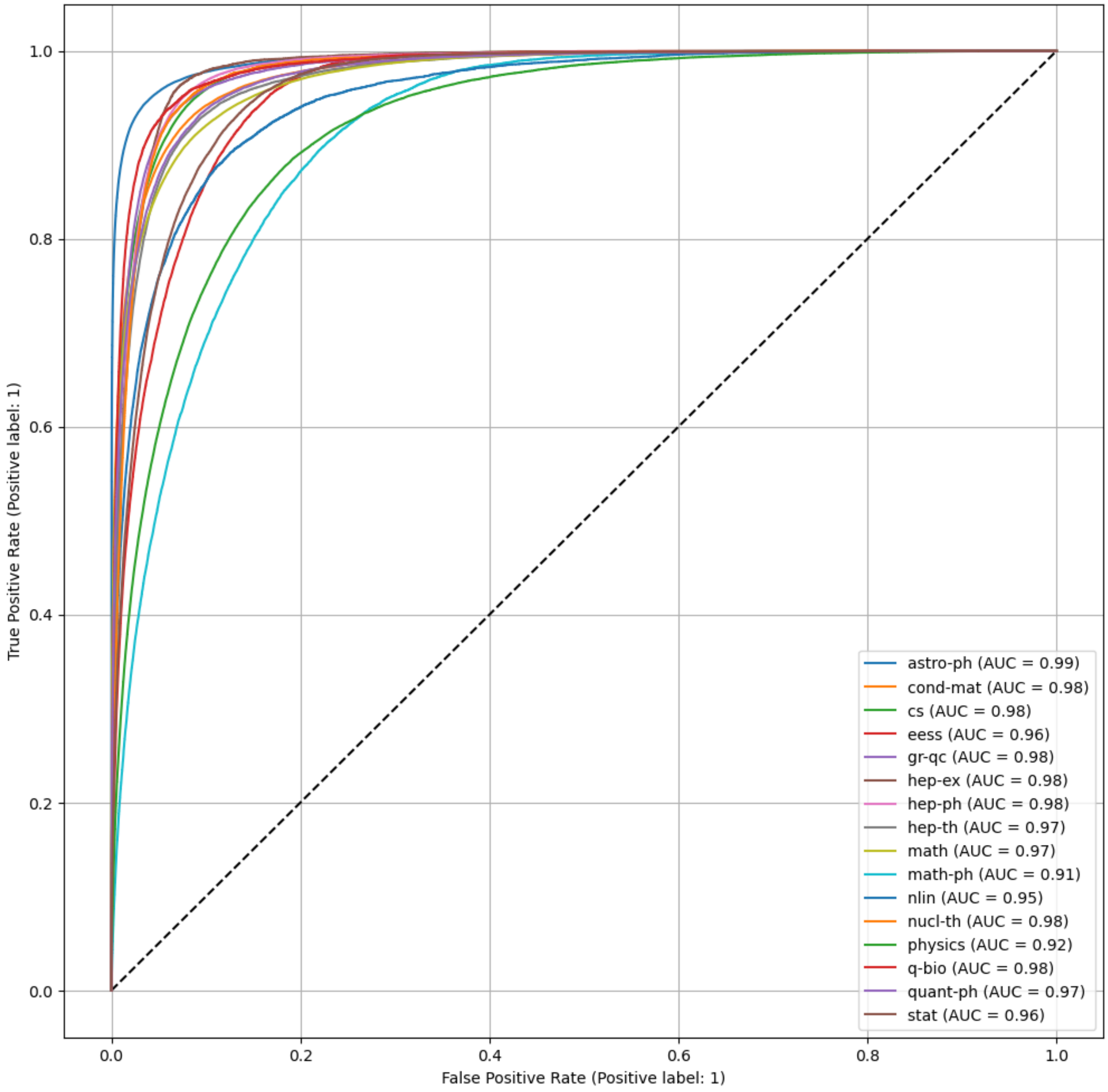
Using the built in grid search, we have determining that the best N-Gram range was (1, 3), meaning that have unigrams, bigrams, and trigrams in the same input, however, we used only unigrams and bigrams for performance reasons, because using trigrams required a lot of memory, far more than is possible to deploy on most affordable web hosting services.

4.5 ROC and AUC

The ROC and AUC were calculated on a per class manner, we have achieved AUC scores of at least 91% for every class, which allows us to achieve good precision and accuracy by adjusting the threshold for classification.

The ROC curve for every class is illustrated in a single chart in Figure 4.1, (the class names are in ArXiv format, for example, “astro-ph” means “Astrophysics”)

Figure 4.1 ROC Curve



Chapter 5: Conclusion and Future Plans

In conclusion, our graduation project focused on the development of an abstract classification system. The objective of the project was to design a model capable of accurately categorizing abstracts into appropriate classes. We successfully implemented a model that demonstrates promising results.

The project involved the creation of a user interface that allows users to conveniently submit abstracts for classification. Through the interface, users can interact with the system and receive classification results.

Throughout the development process, our model underwent rigorous training using a dataset of abstracts, ensuring its ability to handle the abstracts it was given. The accuracy and performance of our model were evaluated using appropriate metrics, demonstrating its effectiveness in accurately classifying abstracts.

Our project highlights the potential of automated abstract classification in various applications, such as academic research, information retrieval, and content organization. By automating the classification process, researchers and users can save time and effort in manually analyzing and classifying abstracts, ultimately enhancing productivity and efficiency.

There are several opportunities for our system's extension and enhancement going forward. This entails strengthening the performance of the model by upgrading our algorithm and adding user input for ongoing development. Additionally, the model has to be strengthened to make sure it can manage a wider variety of classes from diverse fields. We also intend to change login and registration so that they may be used by any user, allowing them, for instance, to view their previously categorized abstracts, rather than just for admin purposes. and providing the user with more information about the degree of result confidence using more measures.

As a result, by creating a system for abstract classification, our graduation project met its goals satisfactorily. Our method, in our opinion, has a great deal of promise for assisting in the effective arrangement and analysis of abstracts across several areas. It might become a useful resource for research institutions, academics, industry experts, and other people looking for effective abstract classification with more work and improvement.

References

(1) *Effectiveness of Support Vector Machine in Analyzing Medical Data*. (n.d.). Engineering Education (EngEd) Program | Section. https://www.section.io/engineering-education/effectiveness-of-svm-on-health-asessment/

(2) Gupta. (2021, June 14). *Email Spam Filtering Using Naive Bayes Classifier*. Springboard. Retrieved May 17, 2023, from https://www.springboard.com/blog/data-science/bayes-spam-filter/

(3) *NLTK :: Natural Language Toolkit*. (n.d.). NLTK :: Natural Language Toolkit. https://www.nltk.org/

(4) Czakon, J. (2022, July 21). *F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose?* neptune.ai. https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc

(5) Chakravarthy, S. (2020, July 10). *Tokenization for Natural Language Processing*. Medium. https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4

(6) *What is Tokenization in NLP? - Lexalytics*. (2021, October 6). Lexalytics. https://www.lexalytics.com/blog/tokenization/

(7) *3.1. Cross-validation: evaluating estimator performance*. (n.d.). Scikit-learn. https://scikit-learn/stable/modules/cross_validation.html

(8) *What is bag-of-words model?: AI terms explained - AI For Anyone*. (n.d.). What Is Bag-of-words Model?: AI Terms Explained - AI for Anyone. https://www.aiforanyone.org/glossary/bag-of-words-model

(9) Team, G. L. (2020, September 20). *Multinomial Naive Bayes Explained*. Great Learning Blog: Free Resources What Matters to Shape Your Career! https://www.mygreatlearning.com/blog/multinomial-naive-bayes-explained/

(10) *Machine Learning: What it is and why it matters*. (n.d.). Machine Learning: What It Is and Why It Matters | SAS. https://www.sas.com/en_us/insights/analytics/machine-learning.html

(11) *What is NLP: An Introductory Tutorial to Natural Language Processing [Updated]*. (2023.). Simplilearn.com. https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/what-is-natural-language-processing-nlp

(12) Borcan, M. (2020, June 8). *TF-IDF Explained And Python Sklearn Implementation*. Medium. https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275

(13) *3.1. Cross-validation: evaluating estimator performance*. (n.d.). Scikit-learn. https://scikit-learn/stable/modules/cross_validation.html

(14) *Category Taxonomy*. (n.d.). Category Taxonomy. https://arxiv.org/category_taxonomy