

Research Paper Classification and Distribution System



Carmen Wen

School of Computer Science
The University of Auckland

Supervisor: Dr Xinfeng Ye, Dr Sathiamoorthy Manoharan

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of
Data Science, The University of Auckland, 2022.

Abstract

Many universities require their students to undertake projects as part of the fulfilment of the students' degree. In many cases, these projects need to be assessed by independent examiners who are not involved in the project. It is time consuming for the students' advisors to find assessors for the students. In this project, we propose a system that help students' advisors to find examiners for their students' projects. The system uses machine learning techniques to identify the subject areas of the students' projects according to the text of students' reports. Based on the identified subject areas, it matches with the scholars who have research interests in the relevant areas. The system aims to provide a transparent environment that allows scholars to freely participate in the system. It is implemented on top of a blockchain. In this project, we collected a set of papers from several top journal and conference publications. Using this dataset, we have evaluated the performance of several classification algorithms, e.g., weighted multinomial Naive Bayes, weighted K-Means Clustering with Logistic Regression and Labeled-LDA. The experimental study shows that the weighted multinomial Naive Bayes model performs best among the three models in terms of accuracy, recall, f1-score and computational time.

Acknowledgements

I would like to express my gratitude to all those who helped me while doing my project and writing this thesis.

My deepest gratitude goes first and foremost to my supervisors, Dr Xinfeng Ye, and Dr Sathiamoorthy Manoharan, for their constant patient encouragement and guidance. They have been supporting me through all the stages of my project for over a whole year, including consistent weekly meetings to advise me on the right direction for my research, etc. Without their consistent and illuminating instruction, this study could not have reached its present form. Especially towards the end of the project, I had the misfortune of contracting with COVID-19 and acting in a traffic accident; their understanding and support helped me get through it.

I also owe my sincere appreciation to my friends and my classmates, who gave me their help and time by listening to me and assisting me in resolving my concerns during the difficult course of the thesis.

At last, I would like to express my thanks to my beloved family for their loving consideration and great confidence in me through these years of the university, no matter in action, monetarily or spiritually.

Contents

Abstract	1
Acknowledgements	2
1 Introduction	7
1.1 Motivation	7
1.2 Problem Statement	7
1.3 Research Objective	8
2 Literature Review	9
2.1 Logistics Regression	9
2.2 Navie Bayes	10
2.3 Label-LDA	11
2.4 K-Means	12
3 Methodologies	15
3.1 Data Structure	15
3.2 Data Collection	15
3.3 Data Preprocessing	16
3.3.1 Data Cleaning	18
3.3.2 Word Segmentation and Data Transformation	19
3.3.3 Feature Construction	20
3.3.4 Feature Dimension Reduction	21
3.3.5 Feature Extraction	21
3.3.5.1 Bag-of-Word	21
3.3.5.2 TF-IDF	22
3.3.5.3 Class-Based TF-IDF	23
3.4 Build Model	24
3.4.1 Multinomial Naïve Bayes	24
3.4.2 K-Means Clustering	25

3.4.2.1	Elbow Method	26
3.4.2.2	Multinomial Logistics Regression	27
3.4.3	Labeled-LDA	27
3.5	Verify Result	28
3.5.1	Confusion Matrix	28
3.5.2	Top-N accuracy	29
4	Distribution System	31
4.1	Preparation	32
4.2	Fabric Network	32
4.3	Application	34
4.4	Hyperledger Explorer	42
5	Experiment Result & Analyse	47
5.1	Main Result	48
5.2	Detailed Results	48
5.2.1	Result of Multinomial Naive Bayes	48
5.2.2	Result of K-Means Clustering with Logistic Regression	52
5.2.3	Result of Labeled-LDA	56
5.3	Performance Analysis	59
6	Conclusions and future work	63
6.1	Conclusion	63
6.2	Limitation	64
6.3	Future Work	64

Nomenclature

Category

<i>ai</i>	Artificial Intelligence
<i>arch</i>	Architecture
<i>dm</i>	Data Mining
<i>edu</i>	Educational Technology
<i>inter</i>	Human-Computer Interaction
<i>net</i>	Computer Networks & Wireless Communication
<i>par</i>	Parallel Distributed System
<i>secu</i>	Computer Security & Cryptography
<i>ssy</i>	Software Systems
<i>theo</i>	Theoretical Computer Science
<i>vr</i>	Computer Vision & Graphic

Model

<i>LLDA</i>	Labeled-LDA model
<i>LR</i>	Logistics Regression model
<i>MNB</i>	multinomial Naïve Bayes (MNB) model

Chapter 1

Introduction

1.1 Motivation

With the rapid development of technology, the reform of the education system, and the improvement of education standards, more and more people are receiving higher education from universities. The external inspection system is a critical component of the University's quality assurance and upgrading process. In order to ensure that the projects carried out by the student are of sufficient quality, it is necessary to implement an assessment procedure that is rigorous and fair in terms of evaluating student achievement as well as academic standards. Several external examiners frequently assess doctoral or master's theses to ensure impartiality. There are around 30,000 scientific journals, with nearly two million articles published yearly [1]. How an academic head should organize and process a large amount of academic paper data to find external reviewers with acceptable status and scholars with relevant research experience and expertise more quickly and accurately while maintaining a high standard of judgment to review is a huge challenge for the current external checking system.

1.2 Problem Statement

According to Larsen et al., the quantity of scholarly papers increases twice every five years [2]. Production of academic articles has never been halted; conversely, it has increased daily. As the long text of academic papers, it is difficult for scholars to accurately extract concise, refined and comprehensible knowledge that meets the needs of the large-scale textual information resources of this type of unstructured data source in the first place. Secondly, the current external inspection system does not allow the use of search engines or the proper classification or indexing of these research papers according to their content, which means it requires significant human resources to focus on classifying documents to maximize the availability of accurate and relevant information. It is challenging for academics to categorize articles into appropriate

categories due to the exponential expansion of paper data each year. Our research is dedicated to how to improve the efficiency of the current external inspection system by using machine learning techniques to classify academic papers properly and using blockchain techniques to build a network and create an internal communication system.

1.3 Research Objective

Text classification is a critical technology for organizing and processing large amounts of document data, which can be used in classifying papers with high quality, effectively helping researchers filter redundant information, quickly and precisely derive search results, and improve evaluation efficiency [3]. Moreover, we proposed a prototype for an improved external inspection system which call PaperSystem. The Hyperledger platform ¹, such as the Hyperledger Fabric network, will be used in this project to build a distributed network with blockchain technology; it allows different reviewers with similar interests in the research papers of students' dissertations to communicate with one another and collaborate on the project using a designed Java GUI application. Simultaneously, an enriched data visual management will be provided to the users, which is supported by Hyperledger Explorer ². Additionally, this project also determines the topics discussed in the student's report based on machine learning models such as topic modelling and topic clustering, and then generates a list of potential examiners based on the reviewer's research interests and selects one of them to be an external evaluation reviewer for the student's academic paper.

¹<https://hyperledger-fabric.readthedocs.io/en/release-1.2/network/network.html>

²<https://www.hyperledger.org/use/explorer>

Chapter 2

Literature Review

Traditionally, there are two types of machine learning, supervised learning and unsupervised learning [4]. Supervised learning is to predict labels based on given data. It learns or builds a learning model by observing the inputs and expected outputs of some labeled training dataset and relies on this learning model to predict the outcome of this function for any possible values of the inputs. Supervised learning includes logistics regression, support vector machine, decision tree, Random Forrest, KNN, Navies Bayes, etc., which can be applied to topic classification [4, 5]. Unsupervised learning is to find the hidden patterns in data. It classifies and infers conclusions by self-learning to observe the internal data structure of the unlabeled training dataset in the analysis phase. The most typical example of unsupervised learning is clustering, such as K-Means clustering. Latent Dirichlet Allocation (LDA) is another unsupervised machine learning algorithm applied to topic modeling [4, 5].

2.1 Logistics Regression

The Logistic Regression Classifier is the most widely used classification model in machine learning, and it is often used to tackle issues involving Binary Classification. Mo et al. experimented with studying the influence of text language and categorization across various health issues with 55,315 tweets [6]. To drastically decrease the data dimensionality and relieve the data sparsity issue of the terminology matrix, they tried to minimize the document terminology matrix by turning the textual information into continuous summary scores. Firstly, they created a logistic regression model using text data gathered from the Twitter social media site, including information on autism spectrum disorders [6]. A binary conclusion is made by picking significant keywords and categorizing tweets according to the frequency of terms, with tweets classed as either related to or irrelevant to the subject (0 = non-ASD, 1 = ASD). The number of the word indicates the number of Odds ratios (OR) has been calculated, where the ORs served to determine how significant the influences of topic categorization were. The use of logistic

regression estimated the aggregated influence scores, and the receiver operating characteristic (ROC) curve analysis was utilised to determine the effectiveness of the classification strategy [6]. The research results show high accuracy which around 92.9% of ASD topics and 87.9% of the non-ASD case from tweets were predicted correctly in their corresponding control group [6].

However, in the projects of work or studies, there are often required to handle the problems with Multinomial Logistic regression. According to the research by Ginting et al. [7], classification based on a Multinomial Logistic model is an excellent method for classifying text when CPU and memory use are comparable [7, 8]. Following the collection of data from Twitter, the researchers extracted the keywords present in each of the tweets. These features were then converted into binary feature vectors using the Term Frequency Inverse Document Frequency (TF-IDF) approach before fitting into the multinomial logistics model. After that, K-Fold Cross-Validation was used to evaluate the performance of the model, which data were separated into ten portions. Their results show that logistic regression performs well in the classification where the text of a phrase from Twitter had components of hate speech (503 hate speech Vs. 514 non-hated speech data) has approximately 87.7% accuracy, 80% precision on average, and 82% in recall on testing data and achieving the greatest accuracy of 84 percent in 10-folds validation [7].

2.2 Naive Bayes

In principle, Naive Bayes (NB) is a generative model. The Naive Bayes algorithm is a classification method based on Bayes' theorem and the assumption of conditional independence of features. It assumes that features are independent of each other. For a given training data set, the joint probability distribution of the input and output is first learned based on the assumption of conditional independence of the features. Then based on this model, for a given input $P(A)$, the output $P(A/B)$ with maximum posterior probability is found using Bayes' theorem. The simple equation is denote as equation 2.1.

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)} \quad (2.1)$$

Unlike other classifiers, naive Bayes is a simple probability map model suitable for multi-classification tasks and incremental training. It has low computational complexity for large-scale data, while the algorithm principle is relatively simple and easy to understand. However, it is sensitive to feature selection [9]. Kolluri et al. extracted text features according to the occurrence frequency of text features and then used the Multinomial Naive Bayes classifier to calculate the components under the word cycle to generate category results [10].

Andrews et al. compare two event models for Naive Bayesian classification [11]. They first explain the multivariate Bernoulli model and the multinomial event model. Assume there are

n documents where $d_i = \{d_1, d_2, \dots, d_n\}$. In the multinomial model, only words that appear in document d_i are involved in the computation of the posterior probabilities. However, in the multivariate Bernoulli model, not only words that appear in document d_i but also words that do not appear in document d but appear in the global word list are also involved in the calculation. To provide better evidence, they conducted comparative experiments on five different datasets, Yahoo Science, Newsgroups, Industry Sector, WebKB, and Reuters datasets. They discovered that the conclusions reached were generally consistent across all of the datasets tested. The multivariate Bernoulli model worked well with limited vocabularies but fared extremely badly when it came to managing extensive word lists. On the other hand, the multinomial model fared better when dealing with large vocabularies; in fact, its average error was 27 percent lower than that of the multivariate Bernoulli model, regardless of the size of the vocabulary. As a result, the multinomial model ought to be a more accurate predictor for a wide variety of document lengths that are included inside data sets [11].

2.3 Label-LDA

Labeled LDA (LLDA) is a probabilistic graphical model modified by LDA and combined with supervision, which describes the process of generating a collection of labeled documents [12, 13]. On the other hand, LDA is an unsupervised learning probabilistic topic generation model, which was applied to machine learning by David et al. in 2003 and has been widely used in topic modeling[12]. The input of LDA is the set of documents and the number of topics, and the output is the topics presented as a probability distribution [12]. Its core is a three-level hierarchical Bayesian network, which assumes the premise that a document is equivalent to a bag-of-words with a random mixture of potential topics and that the words in the bag are independently exchangeable, without grammatical structure or order. Each of these topics is characterized by word distribution bits [12]. Since LDA often learns complex topics to interpret, Blei et al. modified LDA and evolved it into supervised LDA (SLDA) [12, 14]. However, SLDA does not apply to multi-label corpora [13, 14]. The reason is that SLDA restricts documents to be linked with only one label, and it assumes that each document's label is produced from a heterogeneous distribution of empirical themes [13, 14].

Later, Daniel et al. proposed labeled LDA (LLDA) in 2009 [13]. LLDA models each document as a mixture of potential topics and generates each word from a topic. LLDA differs from LDA in that LLDA adds a supervision term to the document-topic distribution by simply restricting the topic model to use only topics that correspond to the observed set of labels of the document [12, 13]. Assuming there is an already known labels, (A, B, C, D). Its topic distribution corresponds to the labels, so four topics correspond to the four labels if they are four-dimensional. Then due to Gibbs Sampling [13, 15], the words will only be sampled in the topics corresponding to this article, thus getting the final topic distribution. For example, a document labeled with B & C, represent as vector (0, 1, 1, 0) might have a topic distribution

corresponding to (0, 0.1234, 0.5678, 0). After obtaining the current document topic distribution and topic vocabulary distribution, LLDA trained the new documents similarly to LDA without labeling restrictions [12, 13].

Bai et al. conducted comparative experiments on LLDA, SLDA, and SVM to classify 5000 news articles from 10 classes [16]. The classification results showed that both LLDA (90.5%) and SLDA (89.4%) outperformed SVM (88.4%) in terms of overall accuracy, particularly if training the data within a limited dataset. Although the accuracy of LLDA is slightly better than SLDA, it has a higher time complexity compared to SVM [16].

2.4 K-Means

Clustering algorithms are widely used in machine learning, and they are divided into four types: Density-based, Distribution-based, Centroid-based, and Hierarchical-based [17]. The K-Means clustering algorithm is the one that is used most often in the field of clustering. It is the simplest unsupervised learning method possible and is based on a concept known as a centroid and an iterative solution-based cluster analysis algorithm in which the data are pre-segmented into K groups, and their respective centroids are randomly initialized. The centroids are locations with the same length as each data point vector. Then the distance between each data point and its seed cluster center is calculated, each data point is assigned to the cluster center closest to it, and the centroid in each class is calculated as the new centroid. The above steps are repeated until the centers of each class do not change much after each iteration. The clustering centers and the objects assigned to them represent a cluster [17].

Since the initial centroids in the standard K-Means method are chosen randomly, it cannot provide any assurance that it will choose documents that are the most different to serve as the cluster centroids; so Li et al. have proposed an improved K-Means algorithm that enhances the performance of clustering [18]. This approach employs the Jaccard distance metric for the purpose of designating the k documents that are the most different from one another as the centroids. In addition, it employs the Word2vec model to train texts into word vectors for text vectorization which helps to determine the significant concepts across all text sentences [18, 19]. They began by gathering information utilizing a web crawler written in the programming language python [18]. The next step in the procedure is the preparation of the data, which consists of removing stop words and segmenting the data[18]. In the end, they used K-Means clustering where vectorized text as input to classify the subjects [18]. Their research shows that the clustering performance will significantly improve by implementing this adjustment [18]. An alternative approach to determine the topics was used by Liang et al [20]. Following the segmentation of words and the optimization of the content of the text via processes including text segmentation, filtering of stop words, and segmentation of words [18, 20], they create the word frequency vector matrix, extract text characteristics using TF-IDF from Scikit-learn, and conduct frequency of words, then they used K-Means clustering to build their model [20].

Dhendra et al applied Elbow method to select the optimal number of clusters for KMeans [21]. Additionally, Zahrotun et al have applied a similar approach; they implemented the range measurement method with cosine similar instead of Jaccard Similarity after calculating the TF-IDF value for the text[18, 22]. Yongli et al. present this new cluster merging strategy as a Quick Hierarchical K-Means Clustering algorithm, arguing that this approach can reduce the time complexity and the number of iterations[22, 23]. Tao et al suggested using a weighted version of the K-Means method to get a better clustering impact [24]. This technique is based on the classic K-Means algorithm, with the addition of changing the input data and their determining relevant weights [17, 24].

Chapter 3

Methodologies

3.1 Data Structure

Text classification refers to the automatic classification of a large amount of text according to certain classification criteria by humans using computers. Before text classification, we first preprocess the text to convert it into a format that the computer can recognize and then perform feature extraction [25, 26] and classification operations using traditional classifiers or some common deep learning methods [3]. This chapter focuses on the whole process of text classification architecture, such as knowledge related to text preprocessing before text classification and several different classification models.

Our project consists of the steps shown in Fig. 3.1. Before building the model, the data is processed in 3 steps, data pre-processing, creating own dictionary, and data post-cleaning. Then we built and compared three data models, Naive Bayes with term frequency-inverse document frequency (TF-IDF), Topic K-Means clustering with Logistic Regression, and labeled-LDA (Latent Dirichlet Allocation). The result will be calculated by the confusion matrix and applied with Top-N accuracy, then verified by the expert ¹.

3.2 Data Collection

We have made 11 classifications of computer science fields according to the subcategories presented by google scholar, which include Artificial Intelligence (ai), Architecture (arch), Data Mining (dm), Educational Technology (edu), Human-Computer Interaction (inter), Computer Networks & Wireless Communication (net), Parallel Distributed System (par), Computer Security & Cryptography (secu), Software Systems (ssy), and Computer Vision & Graphic (vr)

¹Data Process code:
<https://colab.research.google.com/drive/1b5LUHrpv7SdOEnwMU-XVbKvNOiRpumsc?usp=sharing>

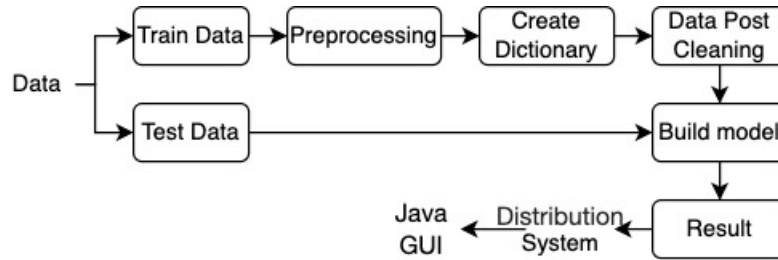


Figure 3.1: Data Process Structure

^{2 3}. Based on the Top Publications suggested by each category from google scholar, Data were gathered randomly from multiple sources manually such as ACM online Library ⁴, IEEE Xplorer ⁵, SpringerLink ⁶, ScienceDirect ⁷, and Wiley Online Library ⁸ (as Table 3.1 shown) with around 3278 papers. Those papers were mainly published within five years. The reason for collecting such data is that the articles published through the various Top Publications ⁹ have been reviewed and approved by experts in the respective industries, which are more representative of their respective fields. We have attempted to write a script and used a web scraper tool that would automatically collect the data; however, because of the copyright issue with these papers stored in various online libraries, we must repeatedly log into the university's database for students to use when conducting academic research, which inevitably makes it more challenging to collect the data. Additionally, different websites use their unique HTML styles. The keys of the articles stored in their source code are not standardized, which raises the possibility that the content obtained by writing an automatic script will not be the data we expected. Therefore, we finally chose the most primitive manual data collection method.

3.3 Data Preprocessing

The natural language processing technique known as text preprocessing is where the text classification process starts [26]. The process of data preprocessing can be broken down into roughly four steps: data cleaning, word segmentation and data transformation, feature extraction and construction, and feature dimensionality reduction [26]. In the first step, some noises in the data need to be cleaned up, which may include filtering invisible characters in ASCII codes, author citation, special symbols, etc. After cutting up all of the sentences in the text into individual words according to the natural spacing between each word, add appropriate space between the words and the words themselves. After that, characters that are not necessary are eliminated, such as stopwords, common words, and punctuation marks. Finally, some vectorization techniques will be applied to the text. There are originally 156235 terms in the dataset and 27159 terms left after cleaning. The pseudo-code for the algorithm is displayed as Algorithm 1.

²Collected Paper Database: <https://drive.google.com/drive/folders/1IQBvRnM586m1rjqy2c8b9pZQ-1O9YOkP?usp=sharing>

³Classification Dataset: <https://github.com/CarmenWen2020/Text-Classification-Dataset-Code>

⁴ACM Library: <https://www.acm.org/>

⁵IEEE Xplorer: <https://www.ieee.org/>

⁶SpringerLink Library: <https://link.springer.com/>

⁷ScienceDirect Library: <https://www.sciencedirect.com/>

⁸Wiley Online Library: <https://onlinelibrary.wiley.com/>

⁹Top Publications Recommendation: https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng

Category	Top Publications	Website / Databases
Artificial Intelligence	Neural Information Processing Systems	ACM Digital Library
	IEEE Transactions on Neural Networks and Learning Systems	IEEE Xplore
	Journal of Machine Learning Research	ACM Digital Library
	Neural Computing and Applications	SpringerLink
Architecture	International Symposium on Computer Architecture	ACM Digital Library
	ACM International Symposium on Microarchitecture	ACM Digital Library
	IEEE Transaction on Computers	IEEE Xplore
Data Mining	IEEE Transactions on Knowledge and Data Engineering	IEEE Xplore
	Knowledge and Information Systems	SpringerLink
	Journal of Big Data	SpringerLink
	ACM Transactions on Intelligent Systems and Technology	ACM Digital Library
Educational Technology	Computers & Education	ScienceDirect
	British Journal of Educational Technology	Wiley Online Library
	The Internet and Higher Education	ScienceDirect
	Education and Information Technologies	SpringerLink
Human Computer Interaction	IEEE Transactions on Affective Computing	IEEE Xplore
	International Journal of Human-Computer Studies	ScienceDirect
	IEEE Transactions on Human-Machine Systems	IEEE Xplore
	Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies	ACM Digital Library
	Proceedings of the ACM on Human-Computer Interaction	ACM Digital Library
Computer Networks & Wireless Communication	IEEE Communications Surveys & Tutorials	IEEE Xplore
	IEEE Communications Magazine	IEEE Xplore
	Journal of Network and Computer Applications	ScienceDirect
	IEEE Network	IEEE Xplore
Parallel Distributed System	IEEE Transactions On Parallel and Distributed Systems	IEEE Xplore
	The Journal of Supercomputing	SpringerLink
	IEEE Transactions on Cloud Computing	IEEE Xplore
	Journal of Parallel and Distributed Computing	ScienceDirect
Computer Security & Cryptography	ACM Symposium on Computer and Communications Security	ACM Digital Library
	IEEE Transactions on Information Forensics and Security	IEEE Xplore
	IEEE Symposium on Security and Privacy	IEEE Xplore
Software Systems	Journal of Systems and Software	ScienceDirect
	IEEE Transactions on Software Engineering	IEEE Xplore
	IEEE Software	IEEE Xplore
Theoretical Computer Science	Journal of the ACM (JACM)	ACM Digital Library
	Theoretical Computer Science	ScienceDirect
	ACM Transactions on Algorithms (TALG)	ACM Digital Library
	Algorithmica	SpringerLink
Computer Vision & Graphic	ACM Transactions on Graphics (TOG)	ACM Digital Library
	IEEE Transactions on Visualization and Computer Graphics	IEEE Xplore
	IEEE Transactions on Image Processing	IEEE Xplore
	International Journal of Computer Vision	SpringerLink

Table 3.1: Top Publications and Databases Of Each Category

Algorithm 1 Preprocess data Functions**Input:** Plain Papers Text Documents At Local**Output:** A Dataset with Tokenizer words

```

for text in each document do
    documentText  $\in$  dataset
    // load textfile to dataset
    dataset  $\leftarrow$  LoadTxt(Textfiles)
    // load textfile to dataset
    dataset  $\leftarrow$  RemoveControlCharacters(documentText)
    dataset  $\leftarrow$  RemoveAuthor(documentText)
    dataset  $\leftarrow$  RemoveSpecialSymbol(documentText)
    dataset  $\leftarrow$  preprocess9(documentText)
end for

```

3.3.1 Data Cleaning

Control Character is a special character that only appears in a specific message text, indicating a certain control function. Standardisation is very important for any program that needs to handle Unicode text in a consistent manner. Since the efficiency of information acquisition is greatly affected by the chaotic storage of the huge amount of text data, Some online document data is stored in HTML format, and some in pdf. When converting this data into a uniform format for storage, some invisible characters in ASCII are occasionally stored confusingly, which does not help much with a text representation. Therefore, it is necessary to clean up the control characters.

There is a wide variety of citation formats in academic writing, where various schools and colleges may require using a particular citation format over another. The most common citation and bibliography formats used in academic writing include APA format, MLA format, CHICAGO format, HARVARD format, and IEEE format. If an academic paper cites an article written by Smith or multiple authors in 2022, then the citation format could be (Smith, 2022), Smith (2022), (Smith & Liu, 2022), Smith and Liu (2022), (Smith et al., 2022), Smith et al. (2022), (Smith, 2022; Liu, 2022; Paul, 2022), etc. The bracket can be either a round bracket or a squared bracket. However, due to a large number of academic papers available online, a single author may publish multiple papers in a variety of fields, or there may be multiple authors with the same name working in various fields. For instance, if the article written by Smith Aja and the article written by Smith Hoka has been cited in two different fields of academic papers, the format of the citation may be the same in both cases (Smith, 2022). However, their presence does not contribute in any way to the classification of the text but may potentially influence the outcome of the classification [27]. Additionally, before the word segmentation process begins, removing author citations must be done. Because splitting will destroy this kind of citation

format, and the computer itself will not be able to recognise it.

Similarly, the presence of special symbols has a negligible effect on the classification results of the paper, despite the fact that these symbols are used extensively throughout the text. In point of fact, the computer is unable to distinguish between what constitutes a special symbol and what constitutes a word; however, the presence of special symbols produces noise for the classification. In addition to the most common non-numerical special symbols and numbers, we found many characters with no significance whatsoever, such as “Figure 1” and “Table 1”. When dealing with lengthy texts, removing references to formatting and other special symbols not only shortens the overall length of the text but also greatly reduces the detrimental effect that author names and useless characters have on classification results.

3.3.2 Word Segmentation and Data Transformation

For our study, we decided to use the NLTK Tokenize library written in Python for word segmentation. The text contents are segmented into individual words, phrases or words so that the computer can accurately determine each word in the text. This method has higher precision and a faster speed of word segmentation, which can improve the accuracy of text classification. The pseudo-code for the process9 function is shown as Algorithm 2.

Algorithm 2 Preprocess9 Function

Input: A Dataset with document text

Output: A Dataset with Tokenizer words

```

0: procedure PREPROCESS9(documentText)
    documentText  $\in$  dataset
    word  $\in$  documentText
    documentText  $\leftarrow$  tokenize(documentText)
    documentText  $\leftarrow$  checkFirstCap(word)
    documentText  $\leftarrow$  lemmtizer(word)
    documentText  $\leftarrow$  filter_stopwords(word)
    documentText  $\leftarrow$  filter_common_words(word)
    documentText  $\leftarrow$  filter_words(word)
end procedure=0

```

The lexicon is standardised through the utilisation of the process9 function. The function begins with cutting and dividing the text sentences and transforming the words with their standardised cases, such as upper case, past tense, progressive tense, etc. Then the words are filtered to exclude deactivated words and common words.

Word normalization is the task of converting words or tokens into a standard format. Case folding is one of these normalization methods. The words that appear at the beginning of a sentence and have a capital letter for their first letter can be changed to lower case. Usually,

the common way of natural language processing is converting all letters to lowercase in many tasks, i.e., to treat SCIENCE, Science, and science as the same word. However, the characters with upper case and a short length are likely to be abbreviations of a full academic name In academic papers. Sometimes case information can be particularly helpful in text classification tasks. For example, ai as a noun means A South American sloth (*Bradypus tridactylus*) with a greenish coat. However, Artificial Intelligence which the simulation of human intelligence processes by machines. Therefore, this study examined words by retaining all uppercase or lowercase words and converting the words which start with uppercase to lowercase letters with their first letter.

3.3.3 Feature Construction

Lemmatization takes the morphological analysis of the words, transforming a word's complex form into its most basic form. Instead of simply removing the prefixes and suffixes, morphological reduction transforms the words according to the dictionary. For example, words such as 'is', 'am', 'are', 'was', 'were', have the common morpheme 'be'. we can get to the root of all the original words by reducing all these different forms to their common morpheme. Thus, the morphological reduction of a sentence like "He was reading amazing stories" is "he be read amaze story" after lemmatization. The purpose of lemmatization is to unify words and reduce a linguistic word of any form to its general form, which helps for the later processed and analysis. The NLTK library in Python contains a lexical database of English words. WordNet is a large word database corpus in NLTK. WordNet contains many cognitive synonyms for each word. In WordNet, nouns, verbs, adjectives, and adverbs are each organized into a network of synonyms. In particular, we use the WordNetLemmatizer dictionary to preserve only the noun original form of words to the greatest extent possible.

Stop Word is an important tool in natural language processing, which is often used to filter noise in word separation results, improve the quality of text features, and reduce the dimensionality of text features. In the process of building a topic model, we find that stop words or common words such as 'of', 'ground', and 'got' do not help express a topic. Since there are too many such words, they occupy an important proportion of a topic's word distribution, making it very difficult to summarize the meaning of a topic. The text data of the research paper was filtered by comparing it with the stopwords and common words databases. The density of the text keywords was increased by removing words, including these prepositions and pronouns that have no meaning and may have an impact on the text classification results. At the same time, the feature dimension of keywords is reduced to a certain extent to improve the classification accuracy and efficiency.

Moreover, it is also essential to filter out some words that are too short based on their length. In lengthy texts, a significant number of the characters are not words and consist of only one or two letters. Some of them may be abbreviations of a full academic phrase, and some may be random characters made up of leftover mathematical formula cuts. Therefore, it is necessary

to eliminate these words or characters since they contributed a few values and negative effects on the classification result. We observed and retained some words with special meaning and only 2 in length. For example, the terms “3D” and “VR” can frequently be discovered in academic papers dealing with computer Vision and Graphics. In papers on computer networks and wireless communication, the terms “5G” and “4G” are frequently encountered. Selecting a number of viable approaches suited to the task at hand to achieve the desired outcome will provide a solid basis for the text classification that will follow.

3.3.4 Feature Dimension Reduction

Following the construction of the data features, it is customarily necessary to perform feature dimension reduction and feature selection on the dataset containing the features. For NLP applications, we performed an operation that is analogous to feature filtering to remove deactivated words and characters with little meaning in the previous steps. According to the count that was done in actual operation, we ended up with approximately 174295 unique characters in our build-up corpus set. However, it can easily lead to a series of problems, including lengthy training times and complicated computations; the reason is that a larger feature matrix requires more storage space. On that account, it is also critical to reduce the dimension of the feature matrix. We have selected the features by observing and applying the Filter method according to the set threshold. During the process of converting the collected academic papers into text, certain words could be concatenated together as a result of the fact that they had been stored in HTML or PDF format. i.e. “create” and “output” are concatenated together to form a feature “createOutput”, or a sentence ‘This Is Secret Encryption Key’ is concatenated together to form a feature “ThisIsSecretEncryptionKey”. There are a large number of examples of such features in our lexicon, but it is impossible to separate them into the correct words at a technical level. In addition, we find that approximately 76027 terms have only been found once throughout the entire corpus set that we are building up, whereas 20922 terms have been found twice. By observation, we can state that the frequency of such concatenated and meaningless words does not appear more than ten times in the entire corpus set, and their length is typically greater than 20. These words take up a high-dimensional space but do not contribute much to the modelling process. Therefore, we have set up the threshold value, conducted the dimensionality reduction of the data according to this threshold value, updated our own corpus set, and then operated the post-cleaning of the data according to the newest corpus set we built to filter out the words that are not in the dictionary.

3.3.5 Feature Extraction

3.3.5.1 Bag-of-Word

Following the text preparation step, we were presented with a collection of tokenized terms. Nevertheless, the system is unable to detect and analyze the text straightforwardly. Therefore,

for the computer to be able to, we will execute an operation known as text representation, including presenting the text in a certain format. The Bag of Words (BOW) model is a text vectorization model that does not consider grammar or word order. Instead, it focuses only on the frequency with which each word appears in the text. To summarise, the tokenized words were transformed into a bag of words or vectors with a defined length, and each word was treated as an independent entity. The initial vector representation of the text at the beginning is represented by the values 0 and 1; a value is given a value of 1 if it is present in the text and a value of 0 if it is absent from the text. *CountVectorizer*¹⁰ and *TfidfTransformer*¹¹ are two text representation approach used in our study while working with scikit-learn.

The BOW model can be implemented using the *CountVectorizer()* function found in the scikit-learn library. *CountVectorizer* is a method of text feature extraction that belongs to the common class of feature numerical computation. For each training text, it only considers the frequency of each word occurring in that training text document. *CountVectorizer* converts the words in the text into a word frequency matrix $a[M \times N]$, M refers to the number of text document, and N represents the number of words in corpus. It calculates the number of occurrences of each word by the *fit_transform()* function. For example, the matrix consists of an element $a[i][j]$, which represents the word frequency of word j that appears in text document i .

However, the BOW model has several weaknesses. Firstly it does not consider the order between words. Secondly, it does not reflect the keywords of a sentence. For example, the sentence “The weather in Auckland is sunny but windy in Wellington.” in the BoW model, it has a word list of ['the', 'weather', 'in', 'auckland', 'is', 'sunny', 'but', 'windy', 'wellington'], and its Count Vectorizer is denoted as [1, 1, 2, 1, 1, 1, 1, 1, 1]. The BOW model suggests that the keyword for this sentence is 'in', but the keyword here should be 'weather', where the BOW model has a wrong suggestion. TF-IDF can provide a solution to this kind of issue.

3.3.5.2 TF-IDF

The Term Frequency-inverse Document Frequency (TF-IDF) method is a statistical analysis method for keywords that can be used to evaluate the significance of a word in relation to a document set or a corpus [28]. The importance of a word is directly proportional to the number of times it appears in the article and inversely proportional to the number of times it appears in the corpus. In other words, the importance of a word increases positively with its number of occurrences in a document but decreases inversely with its frequency in a corpus. This calculation has the potential to reduce the impact that common words have on keywords effectively and to improve the relevance that exists between keywords and articles [28, 26].

In a given document, the term frequency (TF) refers to how often a given word appears in the document. However, due to the different lengths of documents, in order to prevent the phenomenon that the same words appear more frequently in longer documents than in shorter

¹⁰https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

¹¹https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

documents, the word frequency is improved by normalization. For the word t_i in a particular file, its term frequency expression is:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (3.1)$$

Where $n_{i,j}$ is the number of times that word t_i appear in document d_j , and $\sum_k n_{k,j}$ represents the sum of the occurrences of all the words in document d_j .

The inverse document frequency (IDF) reflects how often a word appears in all texts in the entire dataset. If a term is used in a large number of different text documents, the IDF value for that word should be relatively low. Conversely, if a word only occurs in a small portion of the whole text documents, its IDF value should be high. For example, some academic terms such as “Machine Learning” should have a high IDF value in our collected computer-science-based dataset. In a worse case, if a word appears in all texts, its IDF value should be 0. In other words, if a feature term t_i appears more frequently in one text document but less frequently in others, it indicates that this feature item can well distinguish this kind of text from other texts, and it has strong discrimination and representativeness [26]. The expression is:

$$IDF_i = \lg \frac{N}{n_i} \quad (3.2)$$

However, a certain word may not appear in a certain text in the calculation process. In order to prevent the denominator from being zero, the most common method is to use Laplacian smoothing to deal with the above formula [26]. The formula after smoothing should be:

$$IDF_i = \lg \frac{N}{n_i + 1} \quad (3.3)$$

In the calculation process of TF-IDF, the frequency of feature words is used as the judgment basis for the weight of feature words. The weight is calculated by the number of feature words and the proportion of feature term documents. The expression is $TF \times IDF$:

$$w_{dt} = tf_{dt} \times \lg \frac{N}{n_i + 1} \quad (3.4)$$

Where w_{dt} is the calculated weight of feature term t_i in text document d , tf_{dt} is the frequency of feature term t_i in text document d , N is The total number of document, n_i is the number of documents which contain word t_i .

3.3.5.3 Class-Based TF-IDF

Generally speaking, the larger the TF-IDF of a word in an article, the higher the importance of that word in that article will be. However, it relies heavily on the selection of corpus,

which tends to magnify the importance of rare words. Moreover, it cannot take into account the impact of the difference in word location. The same word may occur in more than one class category, but the traditional TFIDF does not consider the weight of the word in different classes. For instance, the word 'network' may frequently appear in Class Computer Networks & Wireless Communication or Artificial Intelligence as a 'neutral Network', but rarely in class Architecture or Education documents. Therefore, we propose a class-Based TFIDF [29].

$$c - \text{TFIDF}_i = \log\left(\frac{W_{ct}}{T_t + 1}\right) \times \log\left(\frac{N}{M_t} \times \frac{A_{ct}}{B_c} + 1\right) \quad (3.5)$$

Where W is the number of times that word term t_i appear in class c , T is the number of times that word term t_i appear in the entire corpus set, N is the total number of documents in dataset, M is the number of documents contain the word term t_i in dataset, A is the number of documents in that class c contain the word term t_i , B is the total number of documents is that class c . To ensure the value from non-negative, add 1 for smoothing.

3.4 Build Model

Text classification is a natural language processing task in which a computer automatically categorizes a document into one or more categories based on certain classification criteria. This section focuses on three topic analysis techniques: multinomial Naive Bayes (MNB), a supervised learning method based on topic classification; K-Means clustering, a supervised learning method based on topic clustering; Labeled-LDA (LLDA), a supervised topic model based on LDA.

3.4.1 Multinomial Naïve Bayes

The Naïve Bayes classification algorithm is simple in principle and relatively easy to implement classification algorithm. The assumption of feature independence is the core foundation of the Bayesian algorithm, which assumes that there is no correlation between features and that they exist independently of each other [10]. On the other hand, this circumstance does not arise during real operations. This assumption enhances the computing process's efficacy and decreases the computational task's complexity. However, although Naïve Bayes does not consider the correlation between the characteristics, it has shown to be quite successful in real-world applications. As a result, it has been widely used in many situations. The Naïve Bayes algorithm is based on the training text base and learns the probability distribution of the corresponding categories of features [10]. Following the training, the category corresponding to the input text characteristics is chosen based on the highest posterior probability for that category. Text classification calculations in Naive Bayes are performed in log space to overcome the underflow issue and boost processing perform.

Nevertheless, the Naïve Bayes method is still subject to certain limitations. The traditional naive Bayes classification algorithm does not consider that the same feature words have different feature weights in different categories during classification. Under the assumption of independence, the importance of the same feature words is the same, which will reduce the accuracy of the classifier. Therefore, it is necessary to give different feature weights to the same feature words according to their different importance levels in different categories. This section will improve the naive Bayes algorithm from the aspect of feature weight and integrate the feature weight calculation method above into the Naïve Bayes classification model to get a more accurate classification effect.

The multinomial Naïve Bayes (MNB) model treats the document as a bag-of-words model. It considers that the frequency of words occurring in a document has an impact on the prediction of document categories [10]. There are $j \in \{1, J\}$ categories and the document categories $C = \{C_1, C_2, \dots, C_j\}$. Assume document D_i has m feature terms $D_i = \{t_1, t_2, \dots, t_m\}$. Its corresponding maximum posterior probability is the category to which the document D_i belongs. The posterior probability formula can be expressed as [26]:

$$\mathbb{P}(C_j | D_i) = \frac{\mathbb{P}(D_i | C_j) \mathbb{P}(C_j)}{\mathbb{P}(D_i)} \quad (3.6)$$

Where $\mathbb{P}(C_j)$ is the prior probability of category C_j , $\mathbb{P}(D_i | C_j)$ is the conditional probability that document D_i in category C_j , $\mathbb{P}(D_i)$ is the joint probability of all feature terms (t_1, t_2, \dots, t_m) .

According to Naive Bayes' conditional independence assumption, the classification results for the document D_i is:

$$C_{\text{map}} = \arg \max_{C_j \in C} P(C_j) \prod_{n=1}^m P(t_n | C_j) \quad (3.7)$$

As we discuss TF-IDF above, the weighted-Naïve Bayes for the feature weights of feature term t_k in text document d_i represented as:

$$C_{\text{map}} = \arg \max_{C_j \in C} P(C_j) \prod_{k=1}^m P(t_k | C_j) \times W(d_i, t_k) \quad (3.8)$$

The process for this model will be evaluated with 10-fold cross-validation to prevent overfitting.

3.4.2 K-Means Clustering

K-Means is a clustering algorithm whose main purpose is to group similar elements or data points into a single cluster. The "K" in K-Means stands for the number of clusters. The distance measure will determine the similarity between two elements and affect the clusters' shape. Typically, the Euclidean distance will be used for K-Means clustering. Euclidean distance is

the “ordinary” straight line. It is the distance between two points in Euclidean space. Assume there are two points in location (x,y) and (a,b), and the distance between two points based on Euclidean distance should be:

$$dist(d) = \sqrt{(x - a)^2 + (y - b)^2} \quad (3.9)$$

In order to use the K-Means algorithm, it is necessary to provide an estimate of the total number of clusters. The algorithm begins by picking k records at random to serve as centroids, then it iterates through the entire data set, placing each record in the cluster that contains the centroid closest to it. After that, the previous centroid is replaced by the mean of the centroid values of the records in each cluster, and the process is repeated until convergence is reached. However, the clustering process of Kmeans belongs to unsupervised machine learning, and the need to specify the number of clusters is one of the disadvantages of this clustering algorithm. An incorrect selection of cluster numbers could result in a decline in clustering performance. The Elbow Method can be used to assess the level of quality present in the clustering process. We will first use Class-Based TF-IDF for feature weighting. Then Elbow Method will be applied to select the best clustering value k according to our dataset, and then use Kmeans clustering to reduce the dimensionality of the train Dataset according to this K value. After grouping the feature words into clusters, the classification algorithm logistic regression was used to predict the text of test data.

3.4.2.1 Elbow Method

The Elbow Method is essentially an indirect method of observation [21]. Based on the data visualization, the data points will be observed with the naked eye in several piles. The core index of the elbow method is sum of the squared errors (SSE).

$$SSE = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2 \quad (3.10)$$

where C_i is the i^{th} cluster, P is the sample point in C_i , and m_i is the center of mass in C_i , which is the mean of all samples in C_i .

SSE is the clustering error of all samples, representing the clustering effect's quality. The essential assumption of the elbow approach is that as the number of clusters k rises, the sample division will get more precise, and the degree of aggregation of each cluster will steadily increase, resulting in a decrease in the sum of squared errors SSE. In addition, when k is fewer than the actual number of clusters, an increase in K will substantially raise the aggregation degree of each cluster, and the descending range of SSE will become substantial. However, as k exceeds the number of clusters, the return to the aggregation degree attained by raising K will quickly decrease, reducing the steepness of the SSE fall. Eventually, it will become flat as the K value continues to rise. The graph of SSE and K resembles an elbow, and the K value corresponding to this elbow represents the actual number of data clusters [21].

3.4.2.2 Multinomial Logistics Regression

Logistic Regression is one of the most classic machine learning methods, and it is a supervised learning method. In our study, we will use multinomial logistic regression as the classifier for topic clustering. The output of the multinomial logistic regression model is the probability distribution of all categories. It uses the Softmax function to map the vector product of the feature vector and the weight vector into the probability distribution [7].

3.4.3 Labeled-LDA

Labeled-LDA (Labeled-Latent Dirichlet Allocation) has been used as the third text classifier in this study. Labeled-LDA (LLDA) is essentially a supervised generative model which is able to establish a mapping between each topic and each label. It is very similar to the original LDA, except that Labeled-LDA takes one more step to generate the set of labels of document d through the Bernoulli distribution ψ^d [16, 13]. The process for LLDA as follow:

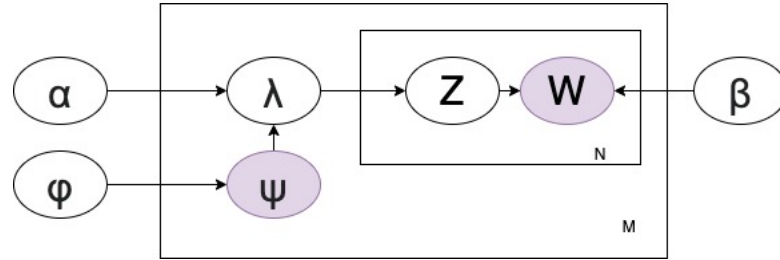


Figure 3.2: Process of Labeled-LDA

- for each topic $j \in \{1, J\}$:
Generate $\beta_j = (\beta_{j,1}, \beta_{j,2}, \dots, \beta_{j,V})^T \sim \text{Dir}(\cdot \mid \beta)$
- for each document $d \in \{1, n\}$:
Generate $\lambda^d = (\lambda_k^d, \lambda_k^d, \dots, \lambda_k^d) \sim \text{Dir}(\cdot \mid \alpha)$
Generate $\psi_k^{(d)} \in \{0, 1\} \sim \text{Bernoulli}(\cdot \mid \phi)$
 $L^d = \psi^d \times \phi^d$
- Iterated each word i in document d , where $i \in \{1, \dots, N_d\}$
Generate Topic $Z_i \sim \text{Mult}(\cdot \mid L^d)$
Generate $W_i \sim \text{Mult}(\cdot \mid \beta_{Z_i})$

Our Labeled LDA model ¹² uses the Gibbs sampling method to extract the topic word distribution iteratively, and its final accuracy is affected by the number of Gibbs sampling iterations

¹²Labeled-LDA model: <https://github.com/JoeZJH/Labeled-LDA-Python>

[15, 13]. When using the Labeled LDA model, the Labeled LDA model is trained in a step-wise manner with a certain number of intervals. Therefore, we manually evaluate the training results of the model to find the most appropriate number of iterations to ensure that the Gibbs sampling process has converged.

3.5 Verify Result

3.5.1 Confusion Matrix

A component of the evaluation of a model is known as the confusion matrix. This matrix is a metric used to evaluate a model's results. A confusion matrix will represent the result from our predicted model in the experiment. In the confusion matrix, each row of the matrix expresses the true category to which the sample belongs, and each column expresses the category prediction that the classifier had for the sample [26]. The confusion matrix represent as:

Confusion Matrix		True Value	
		Positive	Negative
Predicted Value	Positive	TP	FP
	Negative	TN	FN

Where:

- TP: the predicted value is positive, and the true value is also positive, i.e. the number of positive samples correctly predicted.
- TN: the predicted value is negative, and the true value is also negative, i.e. the number of negative samples correctly predicted.
- FP: the predicted value is positive, and the true value is negative, i.e., the number of samples that were originally negative but considered positive.
- FN: the predicted value is negative, and the true value is positive, i.e., the number of samples was originally positive but considered negative.

Based on the confusion matrix, we can calculate four metrics: accuracy, precision, recall, and F1 value, to evaluate the effectiveness of the algorithm [26].

- Accuracy is concerned with the overall model effectiveness for the case of balanced data. It evaluates the proportion of all correctly judged results of the classification model to the total observations.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Precision is defined as the proportion of correct model predictions among all results for which the model prediction is positive.

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

- Recall measures the proportion of actual positives was identified correctly

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

- F_1 score is a composite evaluation metric, the summed average of accuracy and recall; its maximum value is 1, and its minimum value is 0.

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.5.2 Top-N accuracy

In most cases, the classifier will choose the category with the highest probability as the final prediction based on the probability of the sample prediction; this is frequently referred to as Top-1 accuracy. However, in real-world situations, an academic paper can cover multiple domains and have multiple topics. As an illustration, the dataset we collected consisted of 11 different categories. Although artificial intelligence and computer vision & graphics are two distinct fields, it is not unreasonable for a research paper initially written about artificial intelligence to end up focusing on computer vision and graphics. As a result, we concluded that the best way to evaluate the accuracy of the final prediction was to consider its Top-2 accuracy; this indicates that the Top-2 accuracy with the largest probability vector at the end was correctly predicted. If this is not the case, the prediction is incorrect.

Chapter 4

Distribution System

Blockchain is a distributed and immutable ledger that records transactions and keeps track of assets in a corporate network, which is only accessible by licensed network members[30]. It enables a number of different parties to directly record transactions, eliminating the need for a reliable centralised authority to validate transactions. These transactions are intended to be unchangeable and may be validated using cryptography. Generally speaking, blockchain consists of three important components, a distributed ledger, unchangeable record, and smart contracts [30, 31].

- *Distributed ledger*: All network participants have access to the distributed ledger and its tamper-evident transaction records, and every participant is responsible for keeping a copy in the blockchain. In order to eliminate the duplication of effort typical of traditional business networks, transactions are recorded only once by using this shared ledger.
- *Unchangeable record*: Once a transaction has been entered into the shared ledger, none of the participants can alter it in any way or tamper with it. If there is a mistake in the record of the transaction, a second transaction will need to be entered to undo the error, and both transactions will be visible to the user.
- *Smart contracts*: The codes run on the blockchain network, also known as Chaincode. They are used to describe the terms of a trade transaction and are put into action automatically after the preconditions of the contract have been satisfied.

Hyperledger Fabric is one of the open sources and permitted blockchain technologies created by the Linux Foundation [31, 32]. It is made up of several different components, including ledgers, client applications, Chaincode, which is used to run smart contracts for specific businesses, an ordering service, organisations, peer nodes, identities, and membership [33]. Every component fulfils a unique function that serves a certain goal. Endorsement, ordering, validation, and committing are the four primary steps that make up the flow of the transaction [34].

In order for us to be able to start their individualised Fabric network, there are multiply parameters of each component's settings that need to be altered and specified on our own. It is not easy to deal with those parameters because they interact with others. This chapter will discuss the need to build our PaperSystem and how the system work.

4.1 Preparation

The Hyperledger Fabric Application SDK supports multiple languages, including node.js, Go and Java. The study will use the java language to design our system, and this designed system will be implemented under the local host of macOS Monterey 12.3.1, m1 chip.

4.2 Fabric Network

The Hyperledger Fabric network in this study has been formed by one orderer service, three organizations with two peers each ¹. In order to better understand how data works in a fabric network, we will introduce some basic concepts ² and the transaction flow.

- *Channel*: A dedicated “subnet” for communication between multiple members of a particular network that allows participating organizations to join and communicate with each other, but only the members participating in the relevant channel can access any transactions or information associated with that channel. An organization can participate in multiple channels at the same time.
- *Certificate Authority (CA)*: Used for user management and certificate issuance tasks, i.e. user registration, user authentication, etc. Hyperledger Fabric is a blockchain network that requires permission to authenticate, and only authorized users can query or invoke a transaction on the authorized channel.
- *Orderer*: A node that provides consensus services that take over transactions containing endorsement signatures, orders unpacked transactions, creates a new ordered block of transactions, and distributes the newly created block to all Peers nodes on the relevant channel.
- *Peer*: A node maintains the ledger and stores all transactions on the joined channels. Each peer can join one or more channels. Organizations ensure that confidential information is shared only on specific channels to allow participants since the same peer node is stored separately under a different channel. Each peer node can take on multiple roles, i.e. endorser peer, leader peer, committer peer and anchor peer.

¹Network Code: <https://github.com/CarmenWen2020/CarmenNetwork-3Org>

²Key Concepts: https://hyperledger-fabric.readthedocs.io/en/release-2.1/key_concepts.html

- *Endorser Peer*: checks the transaction proposal, executes (read or write) and endorses the transaction execution result of the smart contract.
- *Leader peer*: responsible for communicating with the sorted orderer service node, getting the latest blocks from the orderer node and synchronizing them within the organization.
- *Committer Peer*: responsible for maintaining the ledger, verifying the transaction's legitimacy results in the blocks taken over from the sorted service node and updated to the local ledger. All Peers are currently Committer Peer by default.
- *Anchor peer*: Used for communication between organizations, making peer nodes between different organizations aware of each other's existence.

The architecture of the Fabric network is shown in fig 4.1.

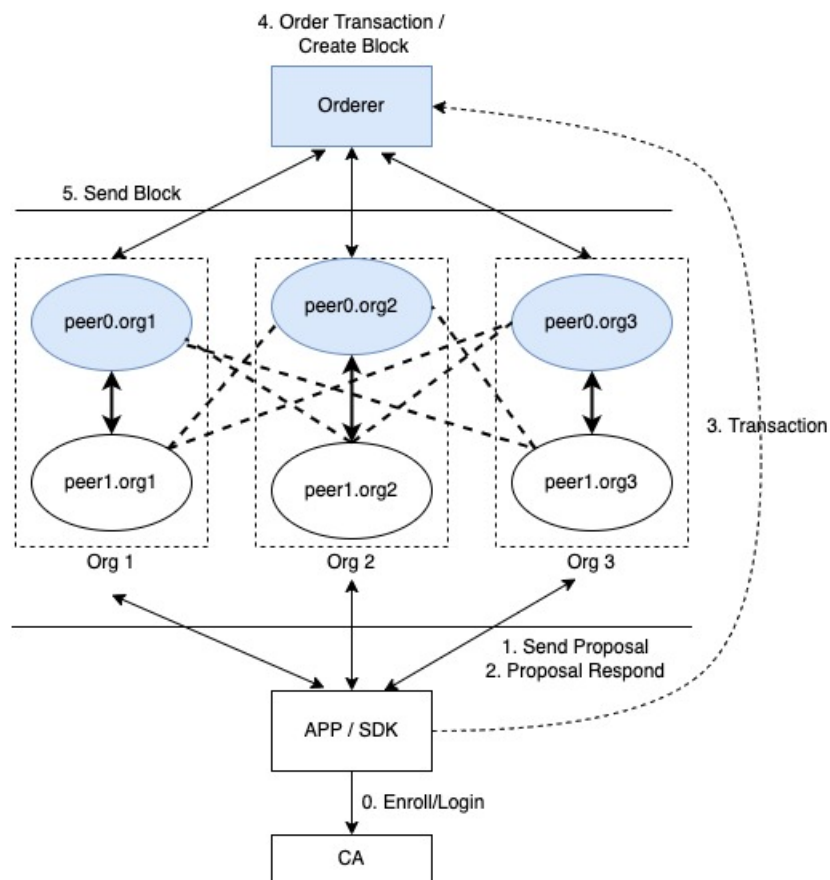


Figure 4.1: The Architecture of Hyperledger Fabric Network

The transaction flow in hyperledger fabric is as follow [35]:

- Step 1: Client SDK obtains the legal identity certificate from CA to join the application channel within the network, then initiates a transaction request and sends the transaction proposal to the endorser peers specified in the configuration file.
- Step 2: The endorser peer verifies the received transaction proposal request, such as ensuring that the transaction proposal is in the correct format, whether the transaction has not been submitted before, and whether the client MSP signature of the transaction proposal is valid, etc. Then it simulates the execution of the transactions and returns the completed endorsement signatures and simulated transaction results to the application.
- Step 3: The application collects a certain number of transaction endorsements. After constructing the transaction request, the transaction proposal and results are sent to the orderer node.
- Step 4: The orderer node globally sorts all legitimate transactions in the network and generates a block structure from the combination of the sorted transactions.
- Step 5: The orderer node sends the generated blocks to the leader peers of different organizations on the channel. Then the committer peer synchronizes the blocks from the leader peer, verifies the block contents, adds the new blocks to the blockchain, and submits the set of write operations to be updated in the state database for all valid transactions. The data are only accessible between the access privileges organization, which built up an interactive connection within the channel.

4.3 Application

With the Chaincode^{1 3} installed in the channel by docker-compose, we are able to invoke the functions defined in the smart contract through the terminal using the corresponding identity certificates, which include createData(), queryData(), queryAllByKey(), richQuery() updateData(), deleteData(). The information is transmitted using JSON format and stored in the CouchDB database⁴. However, Using the terminal to access the test network does have one drawback, which is that the user must write a lengthy and laborious piece of shell code that must contain the identity certificate address and corresponding port of the orderer node and each organization's leader peer, as well as the data in JSON format for the transaction proposal. For the average user, such code conveyed in a command line interface is not only obscure but also prone to errors that could result in an unsuccessful transaction. Therefore, we developed an

³Chaincode: <https://github.com/CarmenWen2020/CarmenNetwork-3Org/tree/master/chaincodes>

⁴CouchDB Database: http://localhost:5984/_utils

application, PaperSystem that interacts with the test network and smart contracts by establishing a connection with the fabric gateway ^{5 6}. Graphical User Interface (GUI) implements the application in Java, and the GUI application’s workflow is shown in Figures 4.2.

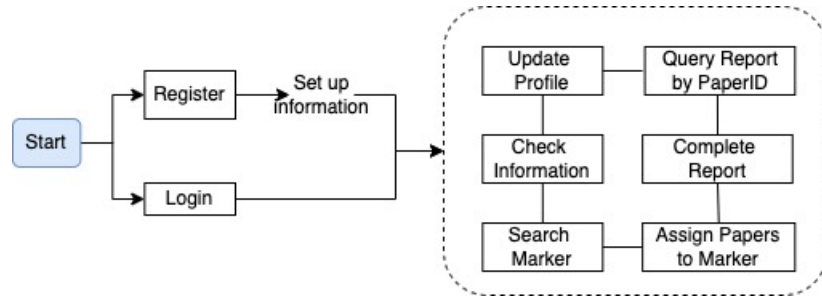


Figure 4.2: workflow for Java GUI Application

From chapter 3, we already know the methodologies of how text classification works. When an academic thesis submit to the postgraduate office of the university, the academic leader can use our constructed models to predict the category of the paper and then pass that identified paper to an expert who is interested in the relevant field for review and scoring through PaperSystem, a Java GUI application we developed ⁷. The following steps show the guidelines for using the application with an example:

Step 1: The user can enter the system by login into their account (as Figure 4.3). The queryData() function will be invoked to verify whether the user exists by their matching username and password, which are stored in the CouchDB database. An encryption technique will be applied to the password to avoid betraying confidential matters. A new user can register the account and then set up their information, such as express their area of interest or experience and their willingness to review relevant papers (as Figure 4.4). The data will be generated in a JSON format (as shown in Figure 4.5), and the request will be submit to the channel using the createData() function. The GUI application will jump to the menu page when the user successfully logs into the system (as shown in Figure 4.6).

Step 2: The option “Update Profile” will allow users to express a different area of interest or change their minds about paper marking (as shown in Figure 4.7).

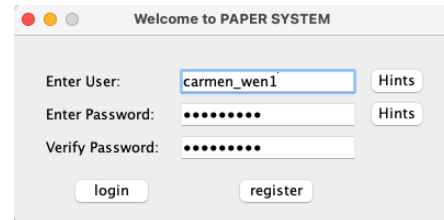


Figure 4.3: Java GUI Application - Login / Register

⁵The official Hyperledger fabric-gateway github: <https://github.com/hyperledger/fabric-gateway>

⁶The official Hyperledger fabric-gateway-java github: <https://github.com/hyperledger/fabric-gateway-java>

⁷Java GUI Code:<https://github.com/CarmenWen2020/GeneralChainCode-App-carmen/tree/master/src/test/java>



Figure 4.4: Java GUI Application - Set up Information

Step 3: The user will be able to check their profile by selecting the “Check Information” option, which will display information such as their area of interest, the amount of credit they have, whether or not they are willing to mark a thesis, and whether or not a paper has been assigned to their account. The credit represents the number of papers that users successfully reviewed. All users are initialised with one credit, and the credit will be accumulated by adding one credit if they have finished a thesis scoring successfully. (as shown in Figure 4.8).

Step 4: Assuming a paper has been predicted from our build models, the student’s advisor can select the “Search Marker” option shown in Figure 4.9 to generate a list of candidates according to the predicted paper’s categories and the candidates’ interest area. The request will be sent to our fabric network using the `richQuery()` function and send a response to the GUI application; the application will show a table with a list of potential examiners based on the matching reviewer’s research interests with their related credit. Only the users willing to score will be displayed in the generated result (as shown in Figure 4.10). The `richQuery()` function supports searching multiple items from the value of the JSON object, which means multiple categories can be selected to search marker. (as shown in Figure 4.11 & 4.12).

Step 5: After the academic leader has searched a marker from Step 4, the predicted paper can be referred to the corresponding reviewer from the search result by entering the reviewer’s name and paper ID under the option “Assign Papers to Marker” (Figure 4.13). The paper must be assigned to someone else, which means self-referred is not allowed (Figure 4.14). Once this operation has been successfully invoked, the user will receive the message for the assigned paper when they log in to the system (Figure 4.15); otherwise, they can check their

```

1- {
2-   "_id": "carmen_wen1",
3-   "_rev": "3-16a0bb6c67ea10e5b148d8d500879e76",
4-   "Credit": 1,
5-   "PW": "44245644ab1fa0aebf3472ebd2e6b0629502f093b0afa153",
6-   "PaperID": "none",
7-   "ReviewState": true,
8-   "ai": true,
9-   "arch": false,
10-  "dm": true,
11-  "edu": false,
12-  "inter": false,
13-  "net": false,
14-  "par": false,
15-  "secu": false,
16-  "ssy": false,
17-  "theo": false,
18-  "vr": false,
19-  "-version": "CgMBJwA="
20- }

```

Figure 4.5: An Example stored in CouchDB with JSON form

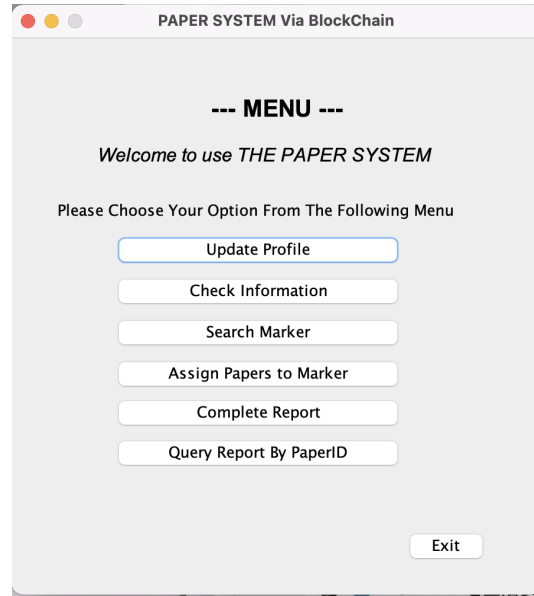


Figure 4.6: Java GUI Application - Menu

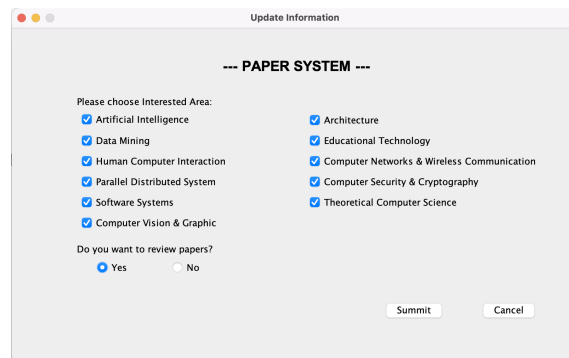


Figure 4.7: Java GUI Application - Update Profile

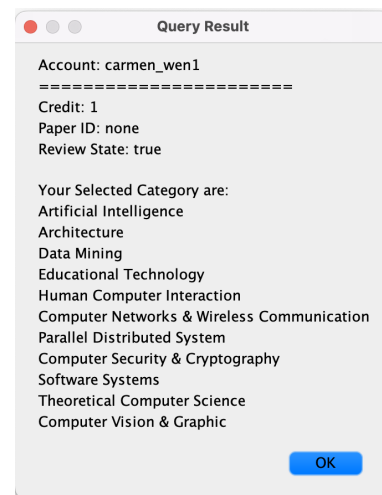


Figure 4.8: Java GUI Application - Check Information

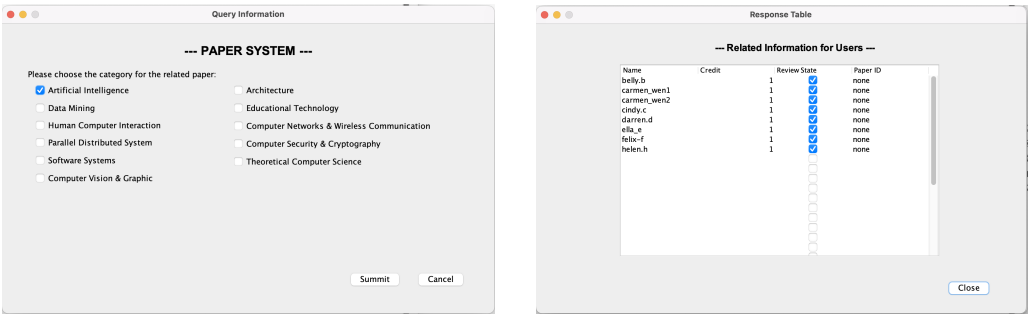


Figure 4.9: Java GUI Application - Search Figure 4.10: List of Markers who interested in AI

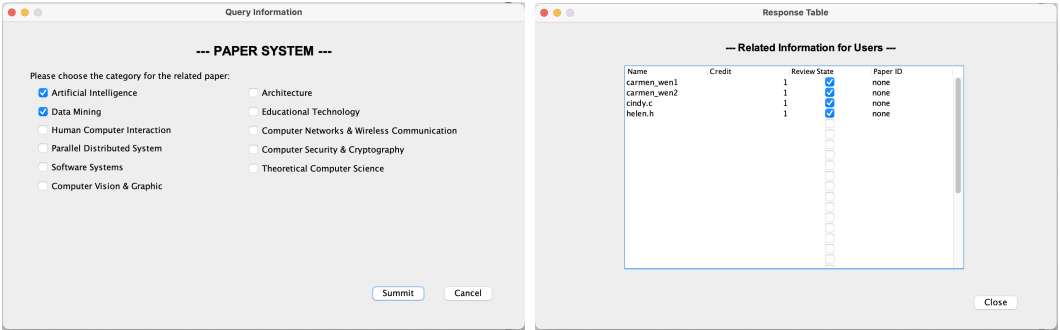


Figure 4.11: Java GUI Application - Search Figure 4.12: List of Markers who interested in AI and Data Mining

corresponding information by step 3 (Figure 4.16).

Step 6: The “Complete Report” option will allow the reviewer to post their report by entering the paper ID and publicising it widely on the world stage. (Figure 4.17). The request is only accepted when the reviewer enters the correct paper ID assigned to their account and completes the report (Figure 4.18). This request will generate the report to JSON form with “PaperID” as key and “MarkedBy” & “report” as values, then send it to our fabric network by `CreateData()` function. Users can operate Step 3 again when they finish (Figure 4.19).

Step 7: All participants in the network can query the report for the reviewed thesis by the option “Query Report By PaperID” on the menu. The report will be displayed the information of the reviewer and their comment (Figure 4.20).

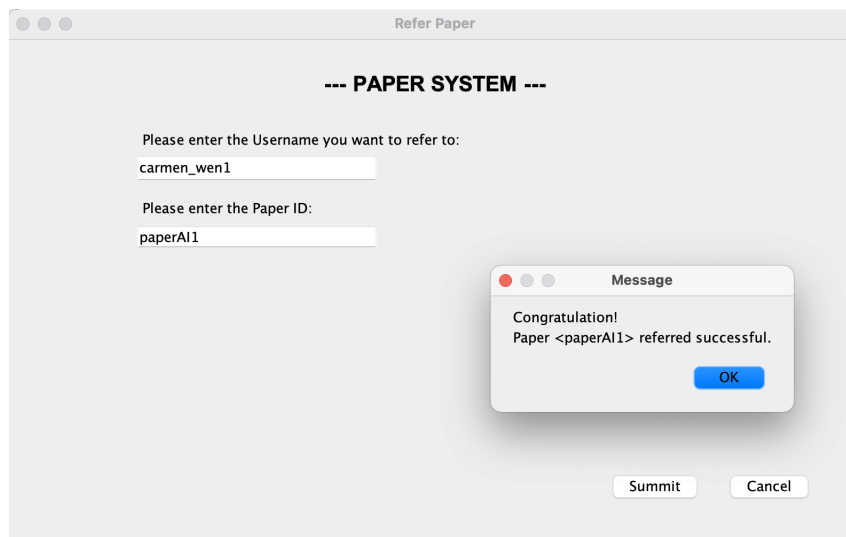


Figure 4.13: Java GUI Application - Assign Paper to Markers

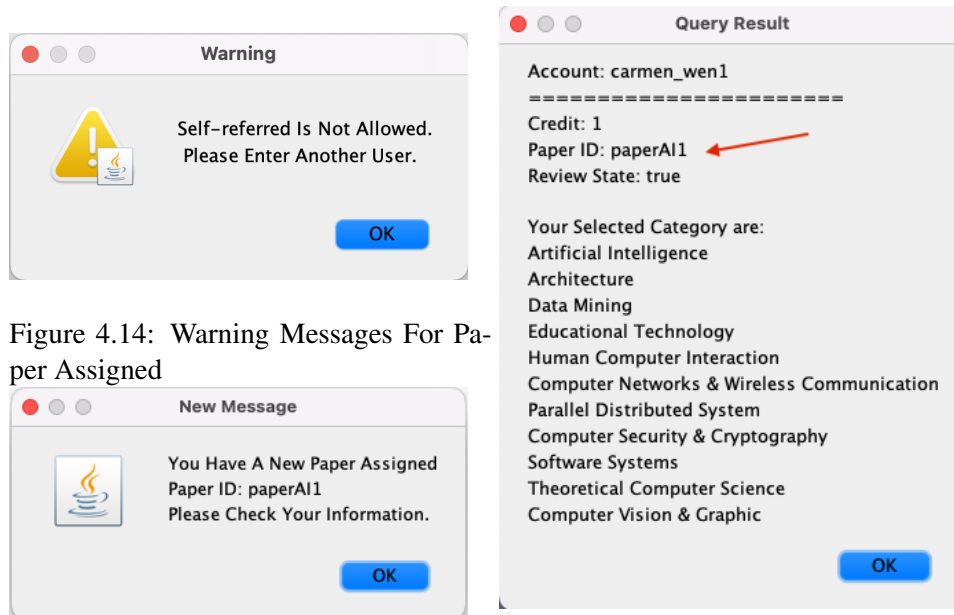


Figure 4.14: Warning Messages For Paper Assigned

Figure 4.15: New Messages For Login

Figure 4.16: Check Information For The Assigned Paper

The 'Finish Paper' window is titled '--- PAPER SYSTEM ---'. It contains the following fields and controls: a text input for 'Please enter the Paper ID:' with 'paperAI1' entered; a question 'Have you finished your marking?' with 'Yes' selected via a radio button; a text area for 'Please post your report here:' containing the text 'This is a report for the paperAI1, which mark by carmen_wen1, and here is some comment bla bla bla'; and 'Summit' and 'Cancel' buttons at the bottom right.

Figure 4.17: The Opertation For Completed Report

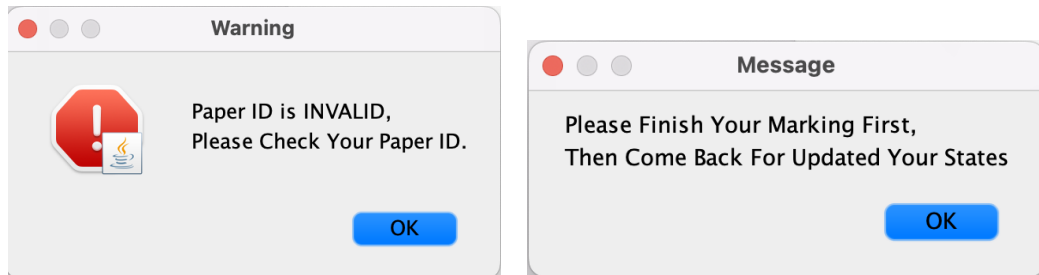


Figure 4.18: Warning Messages for Complete Report

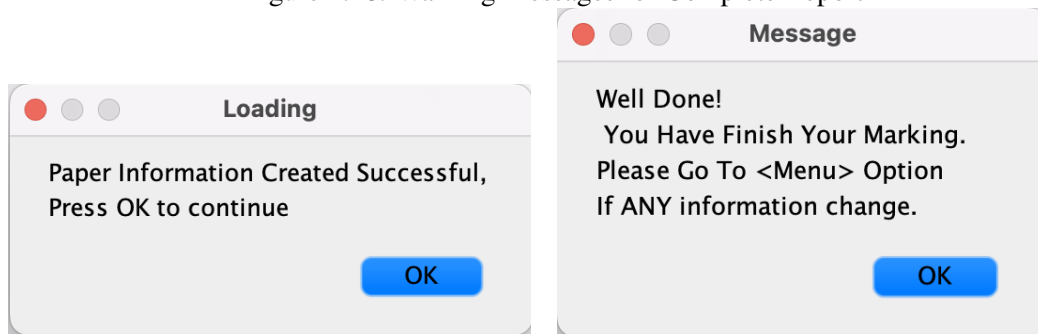


Figure 4.19: Successful Messages for Complete Report

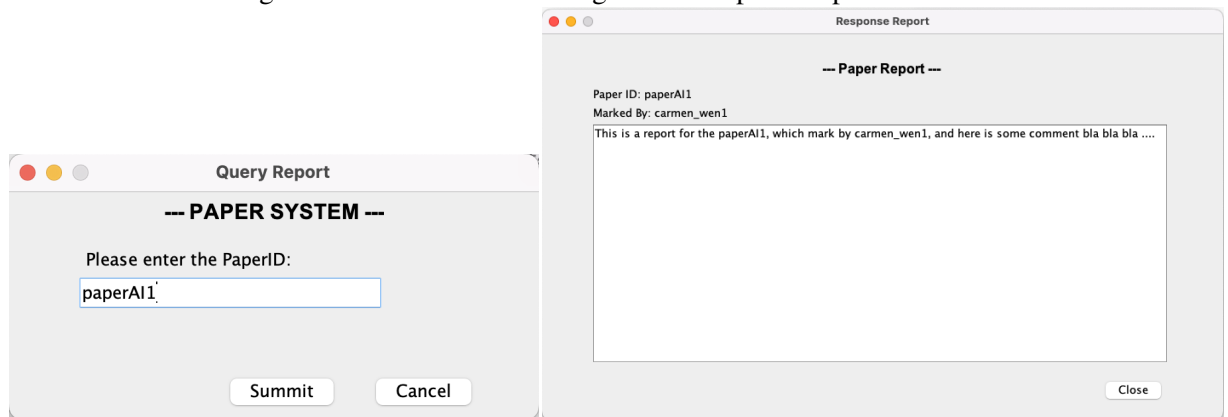


Figure 4.20: The Operation For Query Report By PaperID

4.4 Hyperledger Explorer

Hyperledger Explorer ⁸ is one of the Hyperledger open source projects, which is a highly maintainable graphical visual tool for blockchain. Explorer ^{9 10} has been implemented in our project research to view activity on the underlying blockchain network, including queries for chaincodes, blocks, transactions, and any other relevant information stored in the ledger, such as the transaction detail written from which organisation, the proportion of each organisation, the number of block of each transaction, etc. It also can show if the new organisation has been joined to the blockchain. Explorer supports automated deployment by specifying the configuration parameters, setting up the environment and then presenting through Docker by connecting the localhost port ¹¹. The Dashboard of Hyperledger Explorer is shown as Figure 4.21.

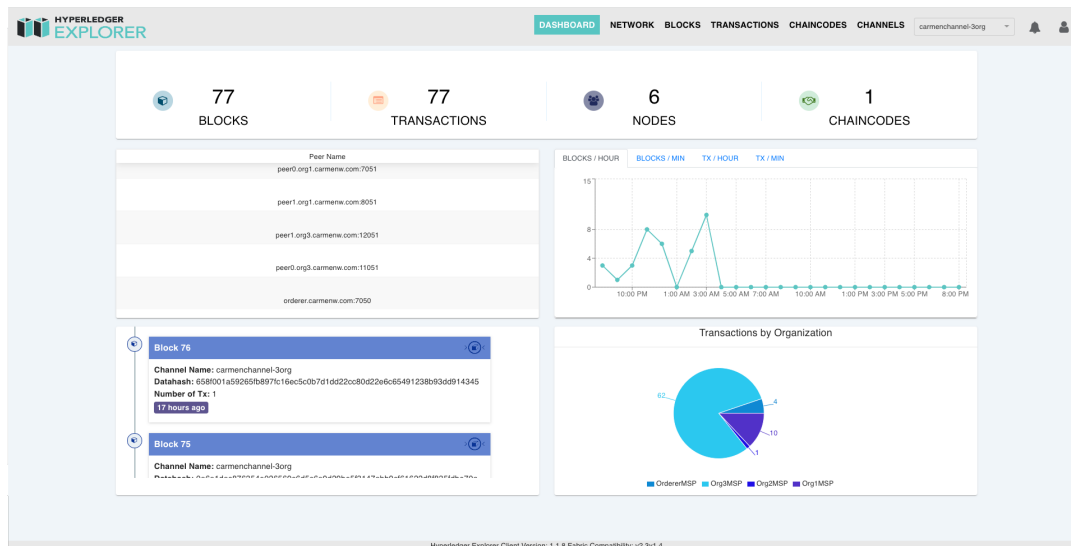


Figure 4.21: Hyperledger Explorer - Dashboard

Figure 4.22 presents the fabric network configuration as discussed in Section 4.2. Explorer can find the transaction detail for each block after the transaction proposal has been successfully submitted to the channel (Figure 4.23 & 4.24). The users need to specify their certificates in the application; this decides which user from which organization sends the transaction proposed by the defined certificates (Figure 4.26). Once the location of the certificates has been set up, the

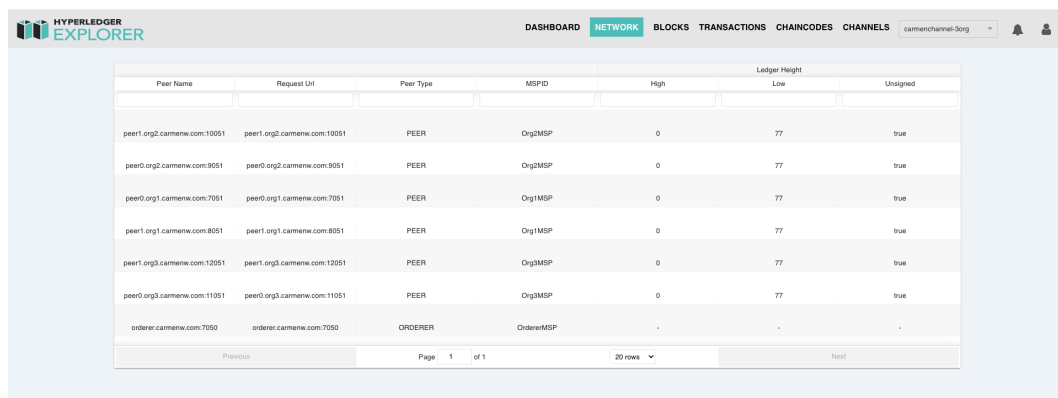
⁸Hyperledger Explorer Office Website: <https://www.hyperledger.org/use/explorer>

⁹Hyperledger Explorer Official Code: <https://github.com/hyperledger/blockchain-explorer>

¹⁰Explorer Files & Codes: <https://github.com/CarmenWen2020/CarmenNetwork-3Org/tree/master/explorer>

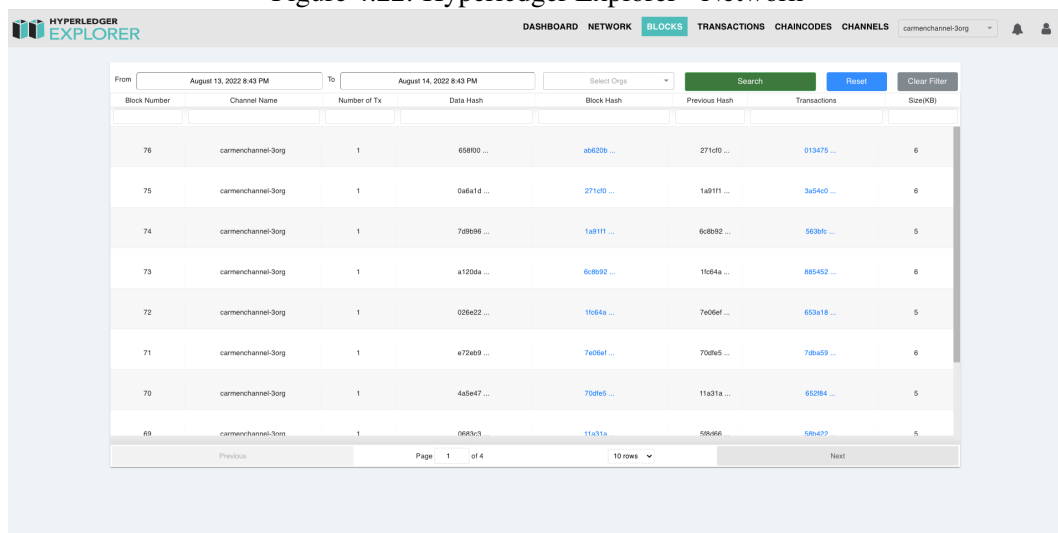
¹¹Explorer Localhost: <http://localhost:9090/#/>

GUI application will use this identity to verify through the fabric gateway for every transaction. We can think of a university postgraduate office as an organization, with the academic leader as the leader peer and the other general academic researchers as the other peers. From the explorer, we can see that *org3* does one of the transactions we explained before (Figure 4.24 & 4.25).



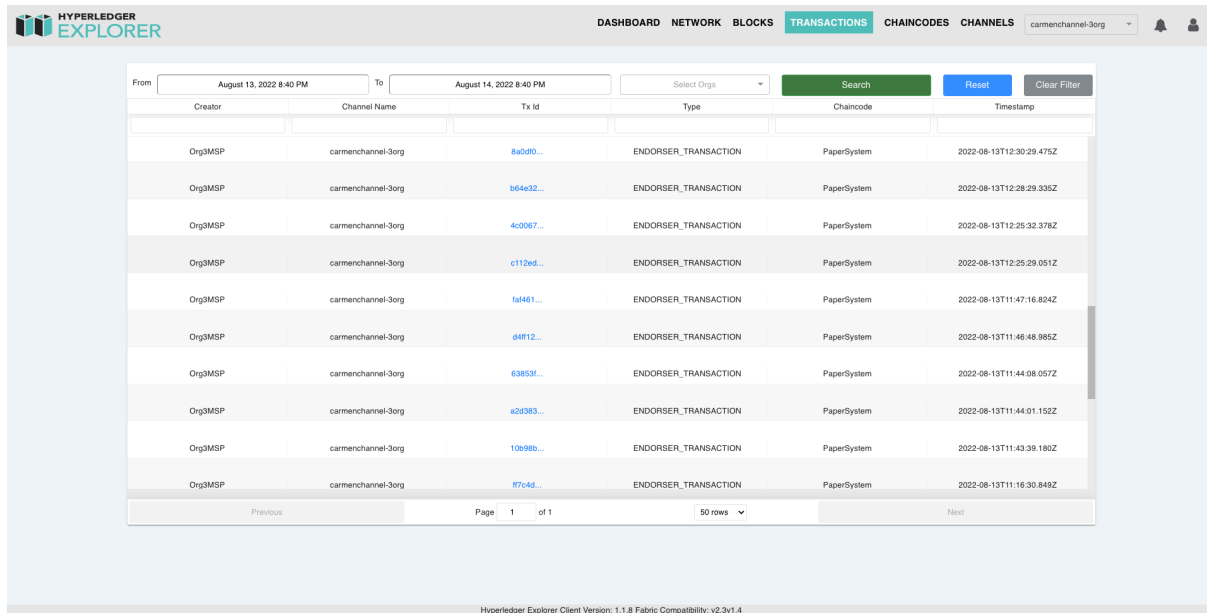
Peer Name	Request URI	Peer Type	MSPID	Ledger Height		
				High	Low	Unsigned
peer1.org2.camernw.com:10051	peer1.org2.camernw.com:10051	PEER	Org2MSP	0	77	true
peer0.org2.camernw.com:9051	peer0.org2.camernw.com:9051	PEER	Org2MSP	0	77	true
peer0.org1.camernw.com:7051	peer0.org1.camernw.com:7051	PEER	Org1MSP	0	77	true
peer1.org1.camernw.com:8051	peer1.org1.camernw.com:8051	PEER	Org1MSP	0	77	true
peer1.org3.camernw.com:12051	peer1.org3.camernw.com:12051	PEER	Org3MSP	0	77	true
peer0.org3.camernw.com:11051	peer0.org3.camernw.com:11051	PEER	Org3MSP	0	77	true
orderer.camernw.com:7050	orderer.camernw.com:7050	ORDERER	OrdererMSP	-	-	-

Figure 4.22: Hyperledger Explorer - Network



Block Number	Channel Name	Number of Tx	Data Hash	Block Hash	Previous Hash	Transactions	Size(KB)
76	camernchannel-3org	1	65800 ...	ab820b ...	271cd0 ...	013475 ...	6
75	camernchannel-3org	1	0a6a1d ...	271cd0 ...	1a9111 ...	3a54c0 ...	6
74	camernchannel-3org	1	7d9b98 ...	1a8111 ...	6c8b92 ...	563b4c ...	5
73	camernchannel-3org	1	a120da ...	6c8b92 ...	11d64a ...	885452 ...	6
72	camernchannel-3org	1	026e22 ...	1b564a ...	7e06ef ...	653a18 ...	5
71	camernchannel-3org	1	e72a69 ...	7e06ef ...	70dfe5 ...	7dbd59 ...	6
70	camernchannel-3org	1	4a5e47 ...	70dfe5 ...	11a31a ...	652984 ...	5
69	camernchannel-3org	1	0683c1	11a31a	659d66	6db479	5

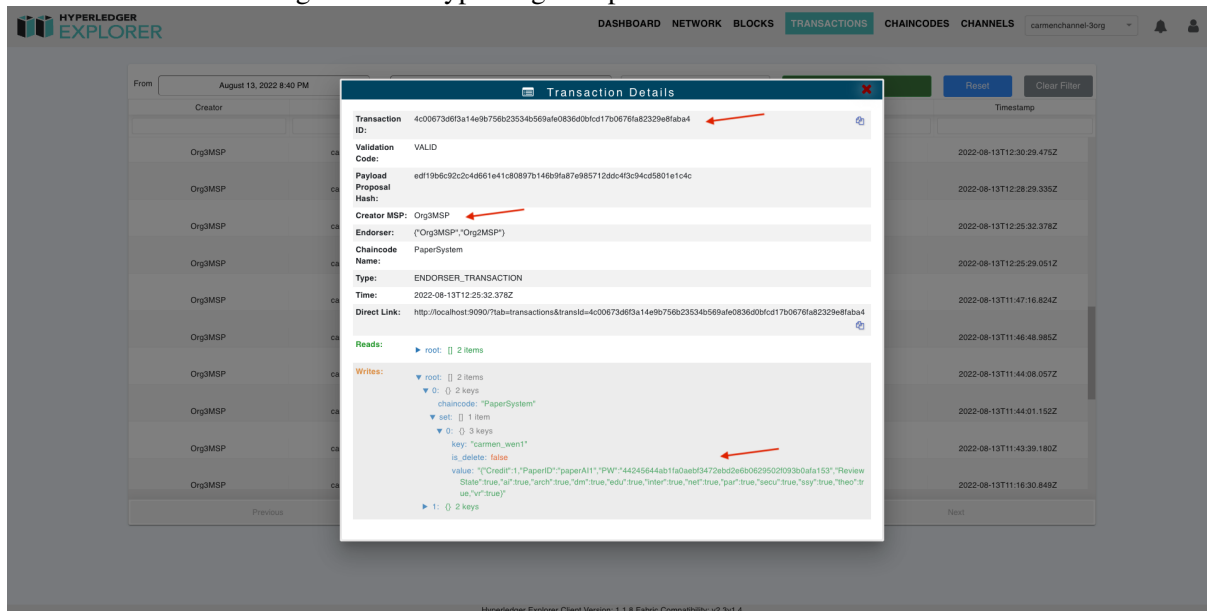
Figure 4.23: Hyperledger Explorer - Block



The screenshot shows the Hyperledger Explorer interface with the 'TRANSACTIONS' tab selected. A table lists transactions for the 'carmenchannel-3org' channel. The table has columns for Creator, Channel Name, Tx Id, Type, Chaincode, and Timestamp. The transactions are all of type 'ENDORSE_TRANSACTION' and chaincode 'PaperSystem'. The table is paginated, showing 10 rows out of 10.

Creator	Channel Name	Tx Id	Type	Chaincode	Timestamp
Org3MSP	carmenchannel-3org	8a0d9...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T12:30:29.475Z
Org3MSP	carmenchannel-3org	b64e32...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T12:28:29.335Z
Org3MSP	carmenchannel-3org	4c0067...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T12:25:32.378Z
Org3MSP	carmenchannel-3org	c112e6...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T12:25:29.051Z
Org3MSP	carmenchannel-3org	fa461...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T11:47:16.824Z
Org3MSP	carmenchannel-3org	d8f12...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T11:46:48.985Z
Org3MSP	carmenchannel-3org	63853...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T11:44:08.057Z
Org3MSP	carmenchannel-3org	a2d383...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T11:44:01.152Z
Org3MSP	carmenchannel-3org	10c985...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T11:43:39.180Z
Org3MSP	carmenchannel-3org	87c4d...	ENDORSE_TRANSACTION	PaperSystem	2022-08-13T11:16:30.849Z

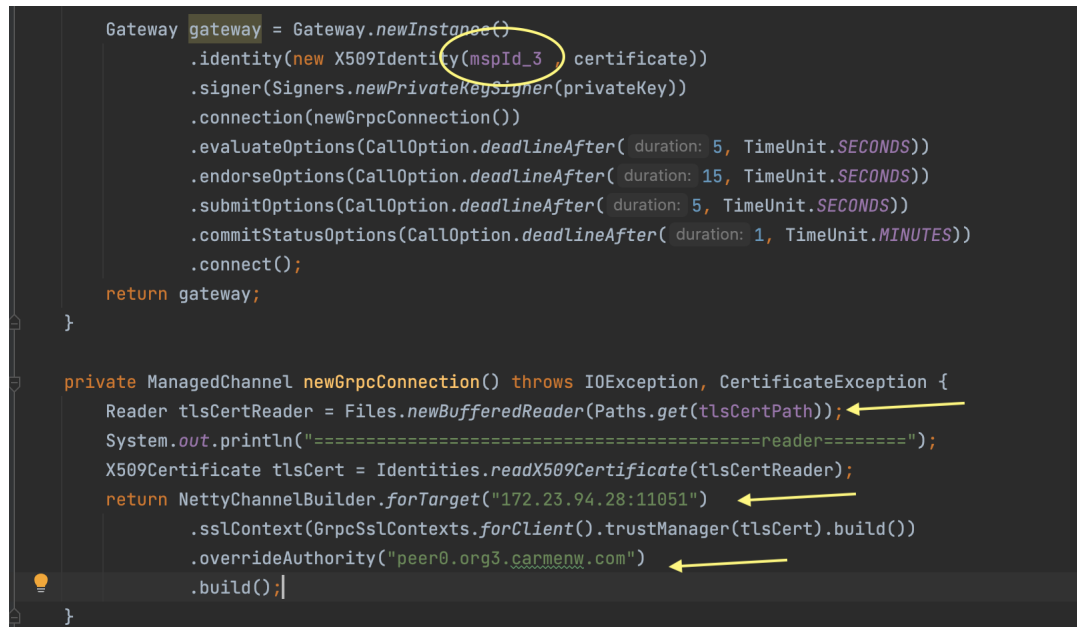
Figure 4.24: Hyperledger Explorer - Transaction



The screenshot shows the Hyperledger Explorer interface with the 'TRANSACTIONS' tab selected. A modal window titled 'Transaction Details' is open, displaying the details of a transaction. The modal includes fields for Transaction ID, Validation Code, Payload, Proposal Hash, Creator MSP, Endorser, Chaincode, Name, Type, Time, and Direct Link. The 'Writes' section shows a JSON object with keys 'chaincode' and 'set'.

Field	Value
Transaction ID	4c00673d6f3a14e9b756b23534b5689a0836d0bfc17b067fa82329e8faba4
Validation Code	VALID
Payload	ed119b6c92c2c4d661e41c80897b146b9fa87e985712ddc4f3c84cd5801e1c4c
Proposal Hash	
Creator MSP	Org3MSP
Endorser	(["Org3MSP", "Org2MSP"])
Chaincode	PaperSystem
Name	
Type	ENDORSE_TRANSACTION
Time	2022-08-13T12:25:32.378Z
Direct Link	http://localhost:9090/txns-transactions&transId=4c00673d6f3a14e9b756b23534b5689a0836d0bfc17b067fa82329e8faba4

Figure 4.25: Hyperledger Explorer - Transaction Detail



The image shows a code editor with Java code for a Hyperledger Explorer Gateway. A yellow circle highlights the `mspId_3` parameter in the `identity` method call. Three yellow arrows point to specific lines in the `newGrpcConnection` method: the first arrow points to the `Files.newBufferedReader` line, the second arrow points to the `return` statement, and the third arrow points to the `overrideAuthority` line.

```
Gateway gateway = Gateway.newInstance()
    .identity(new X509Identity(mspId_3, certificate))
    .signer(Signers.newPrivateKeySigner(privateKey))
    .connection(newGrpcConnection())
    .evaluateOptions(CallOption.deadlineAfter( duration: 5, TimeUnit.SECONDS))
    .endorseOptions(CallOption.deadlineAfter( duration: 15, TimeUnit.SECONDS))
    .submitOptions(CallOption.deadlineAfter( duration: 5, TimeUnit.SECONDS))
    .commitStatusOptions(CallOption.deadlineAfter( duration: 1, TimeUnit.MINUTES))
    .connect();
return gateway;
}

private ManagedChannel newGrpcConnection() throws IOException, CertificateException {
    Reader tlsCertReader = Files.newBufferedReader(Paths.get(tlsCertPath));
    System.out.println("=====reader=====");
    X509Certificate tlsCert = Identities.readX509Certificate(tlsCertReader);
    return NettyChannelBuilder.forTarget("172.23.94.28:11051")
        .sslContext(GrpcSslContexts.forClient().trustManager(tlsCert).build())
        .overrideAuthority("peer0.org3.carmenw.com")
        .build();
}
```

Figure 4.26: Certificates In Application

Chapter 5

Experiment Result & Analyse

Following the workflow present in Figure 5.1 and the methodologies described in chapter 3, we have built three models, multinomial Naive Bayes with TF-IDF using 10-fold cross-validation, Multinomial Logistics Regression using K-Means clustering with class-based TF-IDF, and Labeled-LDA for classifying the labels of academic papers. This chapter will evaluate the performance of those three models and analyse their results.

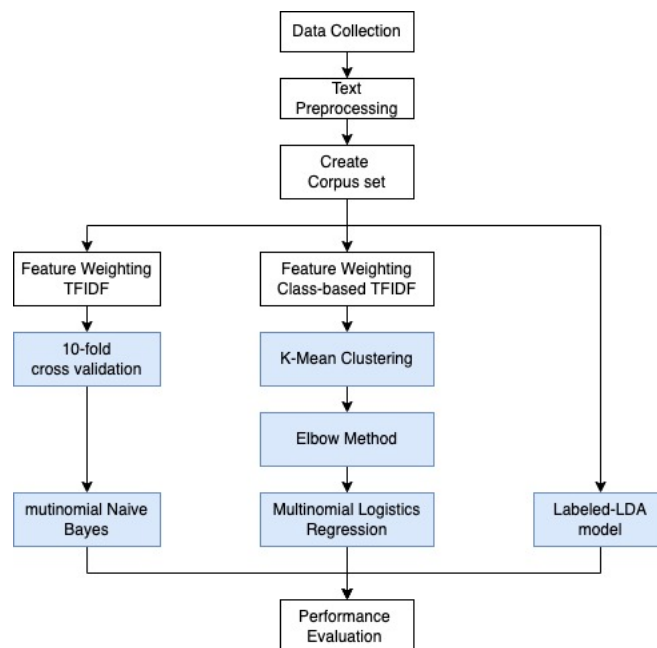


Figure 5.1: The workflow for classification

5.1 Main Result

The result (Table 5.1) shows no significant difference between the Naive Bayes model and the Labeled-LDA model regarding their term of precision, recall, and F1-score. As a matter of fact, Labeled-LDA has slightly higher accuracy than Naive Bayes at Top-1 at around 76.36%, and Naïve Bayes has approximately 76.14%. Nevertheless, K-Means Clustering with logistic regression has the lowest at around 69.32%. At top-2 accuracy, K-Means Clustering and Labeled-LDA perform similarly, with 87.7% and 88.4%, respectively, where Labeled-LDA obtains a slightly lower accuracy, but Naïve Bayes performs the best at approximately around 92.5%. Moreover, Naïve Bayes model has the faster computational time, around 40 minutes for training. In comparison, the K-Means with logistic regression model spends approximately 1 hour, and the Labeled-LDA model takes the longest training time, with more than 10 hours. Overall, the study suggests that the Naïve Bayes model has a better performance compared to others.

Model	Recall	F1-score	Precision	Top-2 Accuracy
Naïve Bayes	0.761	0.761	0.761	0.925
K-Means Logistics regression	0.693	0.693	0.693	0.877
Labeled-LDA	0.764	0.764	0.764	0.884

Table 5.1: Comparison Result for Models

5.2 Detailed Results

5.2.1 Result of Multinomial Naive Bayes

The underlying assumption of the Naive Bayesian is that features are independent of each other. When conducting the experiment for multinomial Naive Bayes, we used 10-fold cross-validation to determine the number of features needed in the training set by comparing the features' respective predicted accuracies on average. In order to find out what number of features has the highest average accuracy in k-fold, Figure 5.2 and Figure 5.3 estimated average accuracy for different number of features in training set. We first calculated accumulated 1000 terms each time; Figure 5.2 shows the highest peak around 5000 terms, but the trend starts to wavy and decreases; it tends to smooth after 20000 words. We have observed the accuracy by order and found out the other peaks are around 4000, 7000, and 6000. Then we conducted a calculation of the number of features ranging from 4000 to 7000. The trend is oscillated and

wiggly and tends to decrease. The highest peak is around 4100 features. Calculating accuracy by the different number of features and selecting the optimized number of features is to prevent the model from overfitting. We finally used around 4100 words based on their TI-IDF values to build our model, and we got a Top-1 accuracy of approximately 76.14%. We finally used around 4100 words based on their TI-IDF values to build our model, and we got a Top-1 accuracy of approximately 76.14%.

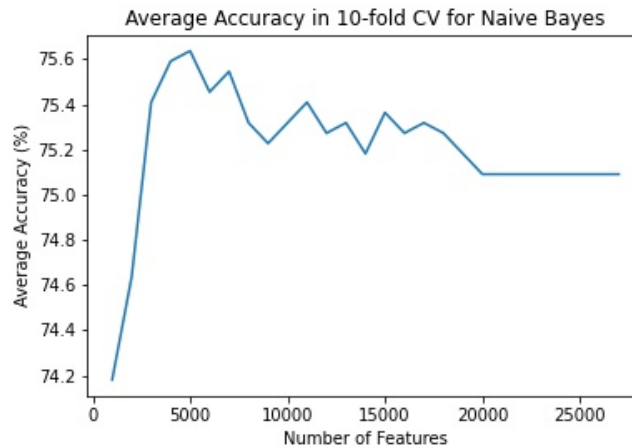


Figure 5.2: Average Accuracy under 28000 terms

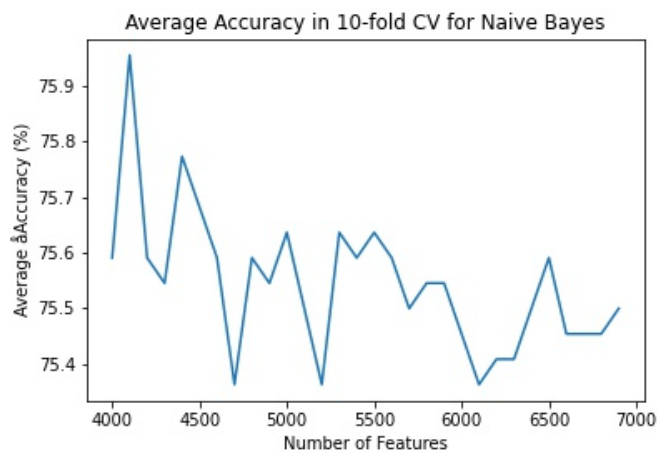


Figure 5.3: Average Accuracy between 4000-7000 terms

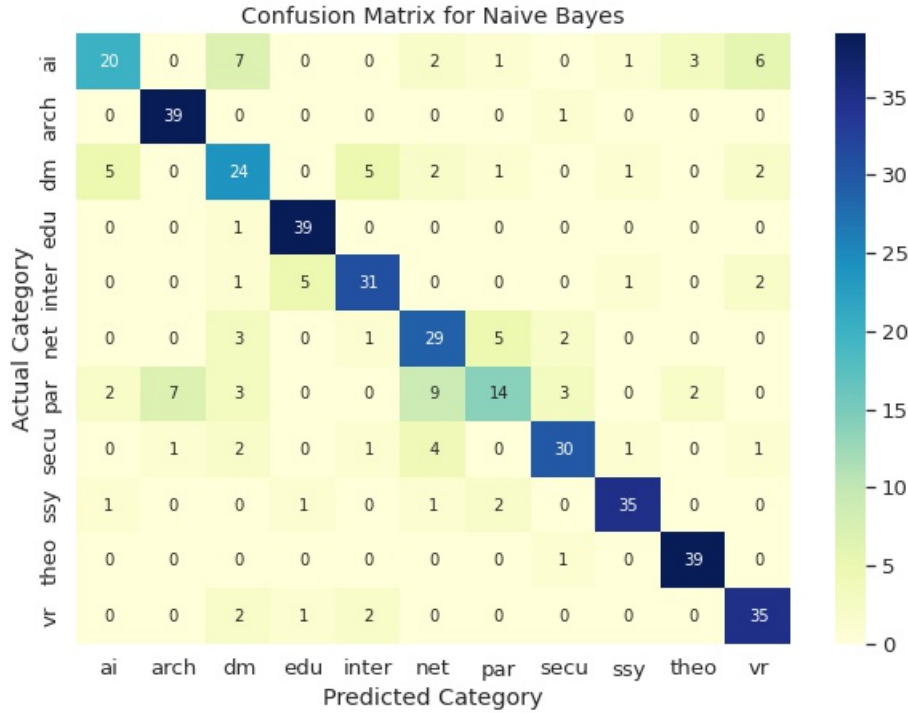


Figure 5.4: Confusion Matrix for Naive Bayes

The confusion matrix in Top-1 labels shown in Figure 5.4 suggests that the papers in the categories *Architecture*, *Educational Technology* and *Theoretical Computer Science* perform an excellent prediction, where 97.5% of them have predicted their corresponding labels correctly. Conversely, some classes can easily be confused with other classes. The most notable categories are represented by *Artificial Intelligence* (*ai*), *Data Mining* (*dm*), and *Parallel Distributed systems* (*par*); only 50%, 60% and 35% of them were predicted correctly, respectively. *Artificial Intelligence* papers have a high probability of predicting as *Data Mining* and *Computer Vision & Graphic*, where seven of them labelled as *Data Mining* and six labelled as *Computer Vision & Graphic*. Moreover, there are seven *Parallel Distributed System* papers predicted as *Architecture*, while there are nine of them were predicted as *Computer Networks & Wireless Communication*. The classification report for each category calculated by the Top-1 confusion matrix was shown in Table 5.2.

Category	Precision	Recall	F1-Score	Sample Size
Artificial Intelligence	0.714	0.500	0.588	40
Architecture	0.830	0.975	0.897	40
Data Mining	0.558	0.600	0.578	40
Educational Technology	0.848	0.975	0.907	40
Human-Computer Interaction	0.775	0.775	0.775	40
Computer Networks & Wireless Communication	0.617	0.725	0.667	40
Parallel Distributed System	0.609	0.350	0.444	40
Computer Security & Cryptography	0.811	0.750	0.779	40
Software Systems	0.897	0.875	0.886	40
Theoretical Computer Science	0.886	0.975	0.929	40
Computer Vision & Graphic	0.761	0.875	0.814	40
accuracy	0.761	0.761	0.761	440
macro avg	0.755	0.761	0.751	440
weighted avg	0.755	0.761	0.751	440

Table 5.2: Classification Report for Naive Bayes

Sometimes it is difficult to precisely determine the class of a paper by the method of human thinking; for example, in the following two articles, one of them is originally *Artificial Intelligence* and predicted to *Data Mining* - "A new hybrid deep learning-based phishing detection system using MCS-DNN classifier" and the other one is originally *Data Mining* but predicted as *Artificial Intelligence* - "Neural networks for model-free and scale-free automated planning". We might think both articles are affiliated with the category *Artificial Intelligence* from the title alone. According to the probabilities of *Artificial Intelligence* and *Data Mining* calculated by the model, the first article has 14% and 23.7%, respectively, while the second article has 26.5% and 16.97%, respectively. Based on the predicted wrong category articles from the Top-1 confusion matrix, we observed the probabilities predicted for each category and found some commonalities. Many of them have a similar probability of the predicted and actual categories; otherwise, the second highest probability could be their true labels. The Top-2 confusion matrix in Figure 5.5 has verified our assumption. Overall, our multinomial Naive Bayes model has 92.5% in Top-2 accuracy.

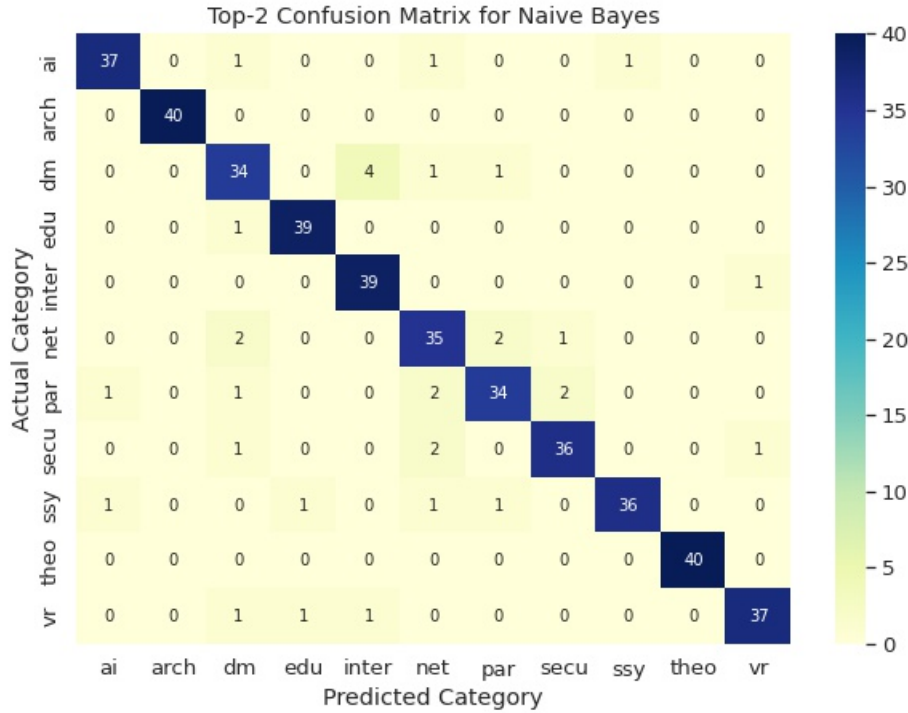


Figure 5.5: Top2 - Confusion Matrix for Naive Bayes

5.2.2 Result of K-Means Clustering with Logistic Regression

Before we built the model, we used K-Means to vectorise features after calculating their TF-IDF values based on their classes, and the Elbow method was used to determine the number of clusters in a data set. We use `KElbowVisualizer()`¹ function in the `yellowbrick.cluster` package to find out the optimal value of `k` in K-Means and plot it as Figure 5.6. The graph shows that the trend starts sharply downward and then gradually slows down after 65; this suggests that 65 clusters should be used for our dataset to train with the logistic regression model. The model's Top-1 (Figure 5.7) and Top-2 (Figure 5.8) accuracies are 69.32% and 87.2%, respectively.

¹<https://www.scikit-yb.org/en/latest/api/cluster/elbow.html>

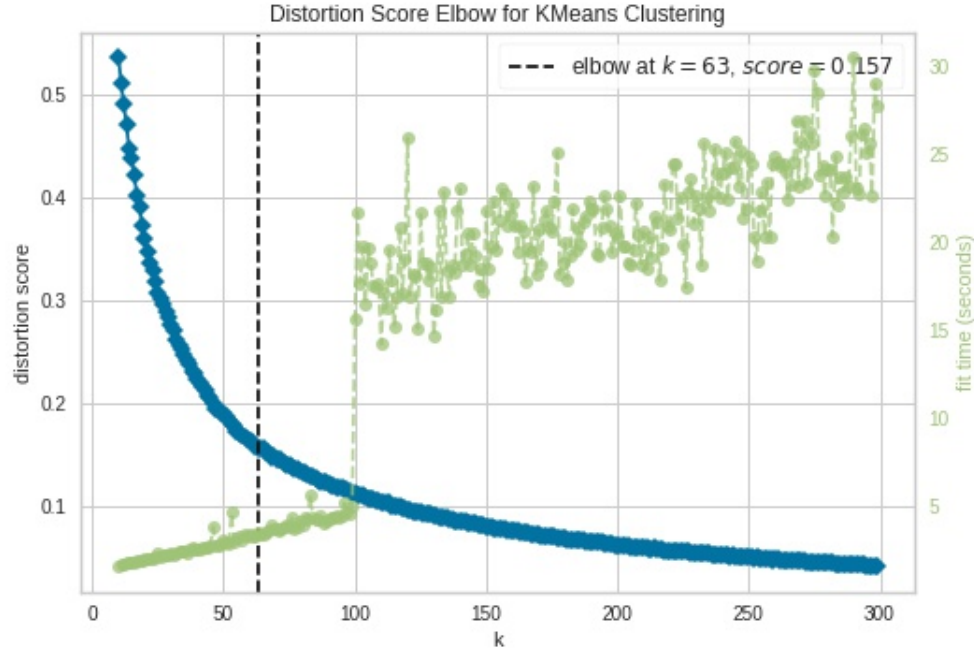


Figure 5.6: The Elbow Method

A similar result is that *Artificial Intelligence* and *Data Mining* papers confuse each other, as shown in the classification report in Table 5.3. Almost half of them are wrongly predicted to other categories; the probabilities are 45% and 50%, respectively, even though they work better when Top-2 accuracy is evaluated. The confusion matrix for Top-2 shows that there are still three papers that were originally *Artificial Intelligence* (*ai*) but predicted as *Computer Networks & Wireless Communication* (*net*), and five papers in true label in *Computer Security & Cryptography* (*secu*) but predicted as *Computer Networks & Wireless Communication* (*net*). We randomly choose a misclassified *Artificial Intelligence* paper² that was predicted as *net*, where the probability predicted to *Artificial Intelligence* is around 1.71% and 53.8% in *Computer Networks & Wireless Communication*. The paper has 3742 words and 752 unique words. We screen-read the paper content rapidly and countered the words by sorting their frequency; then, we found out there are some keywords that may affect the result, such as "clients", "communication", and "server". They contributed a high portion, where "clients" appears 159 times, "communication" appears 90 times and "server" appears 34 times. Another misclassified paper

²Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data:
<https://ieeexplore.ieee.org/document/8889996>

³ was *Computer Security & Cryptography* paper which has 4265 words with 916 unique. The proportion of predicting to *Computer Networks & Wireless Communication* is approximately 87.5%, which is a high probability. We repeated our steps for counting the frequency of the terms. The top-15 words are [”attack”, ”attacker”, ”defense”, ”strategy”, ”server”, ”propose”, ”DDoS”, ”honeypot”, ”IoT”, ”signal”, ”network”, ”framework”, ”defender”, ”payoff”, ”belief”], where many of them seem to occur frequently in both classes, and misleading the result. Overall, the model is predicted reasonably.

Category	Precision	Recall	F1-Score	Sample Size
Artificial Intelligence	0.500	0.450	0.474	40
Architecture	0.825	0.825	0.825	40
Data Mining	0.476	0.500	0.488	40
Educational Technology	0.841	0.925	0.881	40
Human-Computer Interaction	0.730	0.675	0.701	40
Computer Networks & Wireless Communication	0.596	0.700	0.644	40
Parallel Distributed System	0.553	0.525	0.538	40
Computer Security & Cryptography	0.629	0.550	0.587	40
Software Systems	0.897	0.875	0.886	40
Theoretical Computer Science	0.854	0.875	0.864	40
Computer Vision & Graphic	0.707	0.725	0.716	40

accuracy	0.691	0.691	0.691	440
macro avg	0.692	0.693	0.691	440
weighted avg	0.692	0.693	0.691	440

Table 5.3: Classification Report for Multinomial Logistics Regression Using K-Means Clustering

³Towards Fine-grained Network Security Forensics and Diagnosis in the SDN Era:
<https://dl.acm.org/doi/10.1145/3243734.3243749>

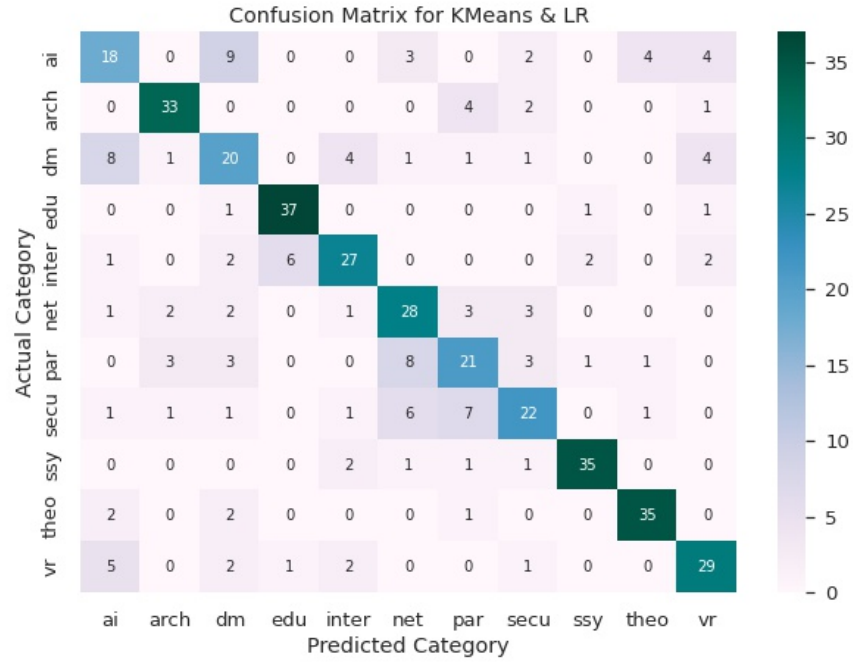


Figure 5.7: Confusion Matrix for KMeans & LR

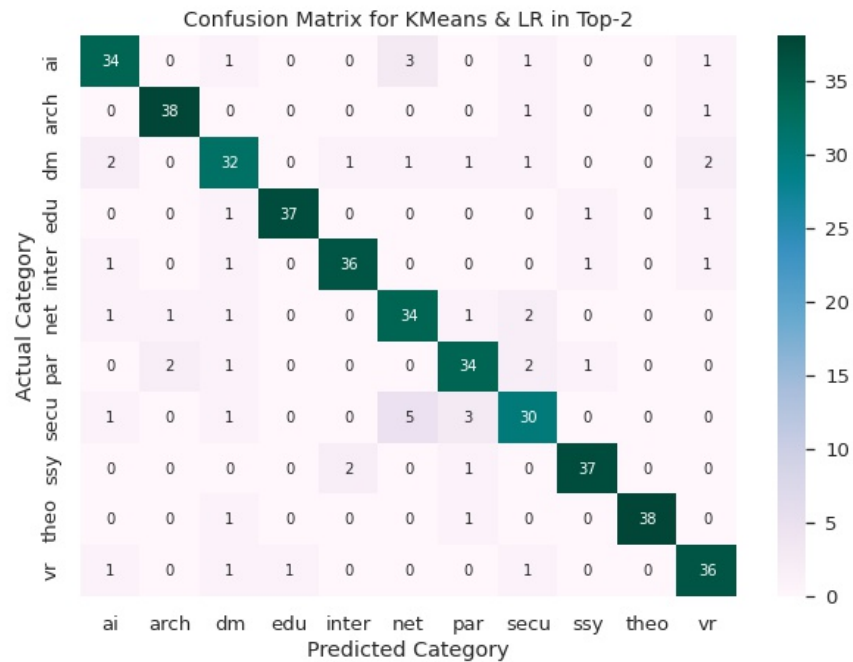


Figure 5.8: Confusion Matrix for KMeans & LR in Top-2

5.2.3 Result of Labeled-LDA

In order to guarantee that the Gibbs sampling procedure has reached a state of convergence, we manually assess the outcomes of the model's training, such as observing the value of delta beta and perplexity, to determine the optimal number of iterations. According to what is displayed in Figure 5.9, the level of complexity decreases as the number of iterations increases. The initial value of perplexity started around 3101; it decreased dramatically between 0 and 20 iterations, then gradually slowed down noticeably for the remaining iterations. The perplexity seems to converge at around 60 iterations, and the trend tends to be smooth.

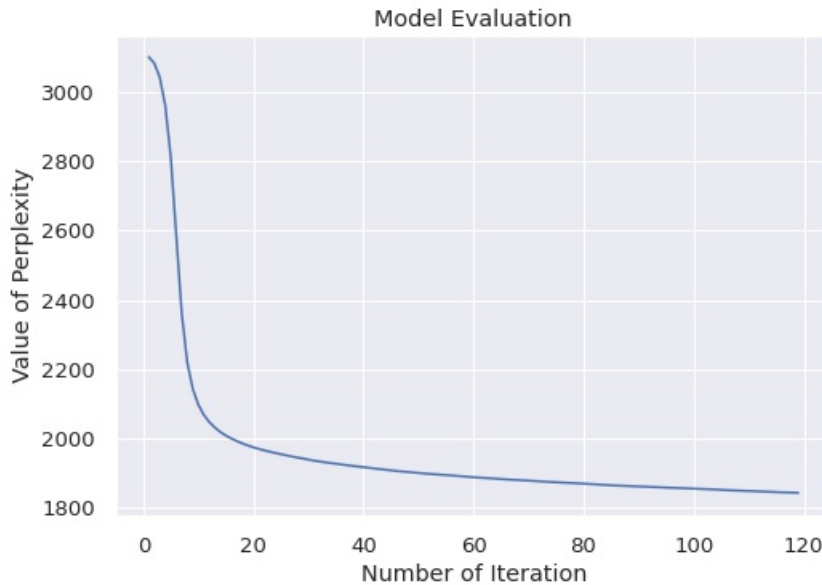


Figure 5.9: The convergency of Labeled-LDA

After 119 iterations, we got a Labeled-LDA model with Top-1 and Top-2 accuracies corresponding to 76.36% and 88.41%. From the classification report in Table 5.4 we could see that the papers with their true labels *Artificial Intelligence* have a high probability of being predicted as *Data Mining* and *Computer Vision & Graphic Model*; this happened in the prediction on the previous two models, and it seems reasonable. However, the category in par has a worse prediction on Labeled-LDA, which is only 37.5% predicted correctly on its true label; even though the Top-2 true label has been calculated, it still performs poorly. The confusion matrix calculated by Top-2 shows that around six par papers were labelled as *Computer Networks & Wireless Communication*; four were labelled as *Architecture*, and three were labelled as *Artificial Intelligence* (Figure 5.10 & 5.11).

One example is that we look through the papers with the highest error rate of category, which was predicted to *Computer Networks & Wireless Communication* by following the steps for counting the frequency of the terms for papers and the Top-50 terms provided by the model for both topics, *Computer Networks & Wireless Communication* and *Parallel Distributed System*. For example, one paper⁴ has 4062 words, with 793 in unique. Its Top-20 terms are ['service', 'data', 'user', 'migration', 'virtual', 'machine', 'router', 'identifier', 'network', 'locator', 'tunnel', 'migrate', 'protocol', 'mobile', 'separation', 'equipment', 'decision', 'controller', 'optimal', 'IP'], with the amount of terms are [176, 154, 104, 92, 88, 82, 76, 72, 70, 59, 52, 49, 44, 42, 40, 40, 39, 39, 34, 33], respectively, where the terms ['protocol', 'service', 'controller', 'mobile', 'data', 'user', 'network'] has interacted with Top-50 terms in topic *Computer Networks & Wireless Communication* and terms ['machine', 'data', 'network'] has interacted with Top-50 terms in topic *Parallel Distributed System*. The terms "data" and "network" are the common words for those two topics; this suggests that *Computer Networks & Wireless Communication* has a high proportion in this paper, but the paper is a particular *Parallel Distributed System* paper.

Category	Precision	Recall	F1-Score	Sample Size
Artificial Intelligence	0.667	0.550	0.603	40
Architecture	0.826	0.950	0.884	40
Data Mining	0.600	0.600	0.600	40
Educational Technology	0.848	0.975	0.907	40
Human-Computer Interaction	0.732	0.750	0.741	40
Computer Networks & Wireless Communication	0.652	0.750	0.698	40
Parallel Distributed System	0.600	0.375	0.462	40
Computer Security & Cryptography	0.816	0.775	0.795	40
Software Systems	0.897	0.875	0.886	40
Theoretical Computer Science	0.950	0.950	0.950	40
Computer Vision & Graphic	0.739	0.850	0.791	40

accuracy	0.764	0.764	0.764	440
macro avg	0.757	0.764	0.756	440
weighted avg	0.757	0.764	0.756	440

Table 5.4: Classification Report for Labeled-LDA

⁴Follow-Me Cloud: When Cloud Services Follow Mobile Users: <https://ieeexplore.ieee.org/abstract/document/7399400>

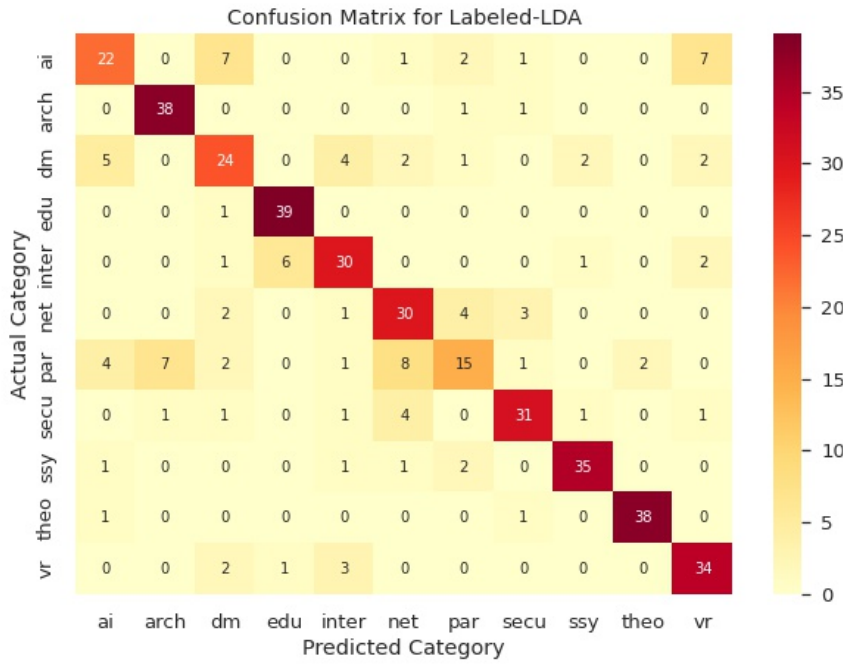


Figure 5.10: Confusion Matrix for Labeled-LDA & LR

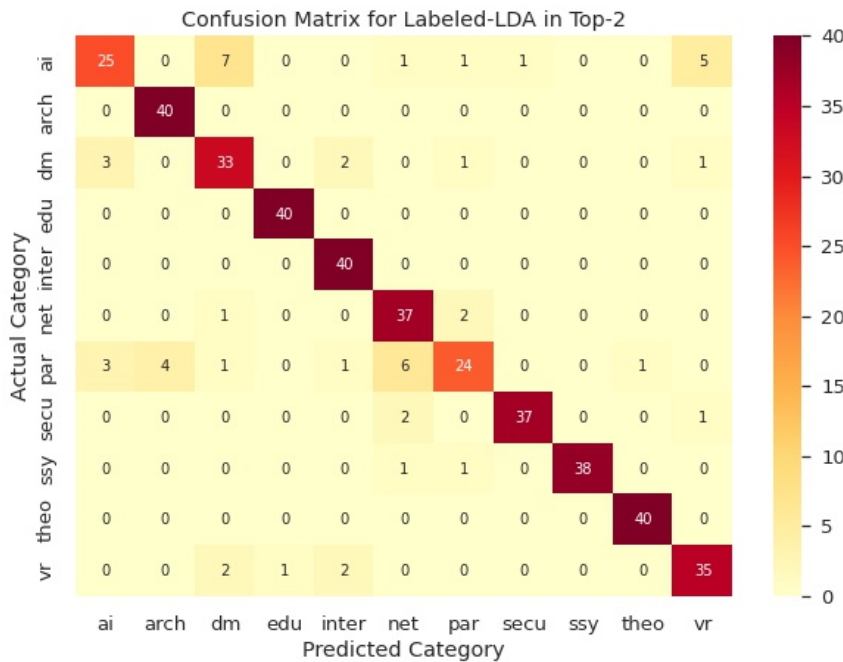


Figure 5.11: Confusion Matrix for Labeled-LDA in Top-2

5.3 Performance Analysis

After analysing the outcomes for each model, we discovered that the articles that corresponded to the topic that had been incorrectly predicted were distinct for each model. The vast majority of them did not interact with one another. For instance, an *Artificial Intelligence* paper⁵ has different Top-1 labels among three models, where it has been labelled as *Computer Vision & Graphic Model* in the multinomial Naive Bayes model (MNB), contributing 53.1% of probability. It also has been predicted as *Data Mining* papers with the probability of approximately 83% in the Logistics Regression model (LR) and predicted as *Computer Vision & Graphic Model* in the Labeled-LDA model (LLDA) with the highest probability of around 99%. At the same time, the second label in multinomial Naive Bayes model is its true topic *Artificial Intelligence* and *Data Mining* in LR. Hence, we compared the Recall for evaluating the performance of predicting the topic between different models.

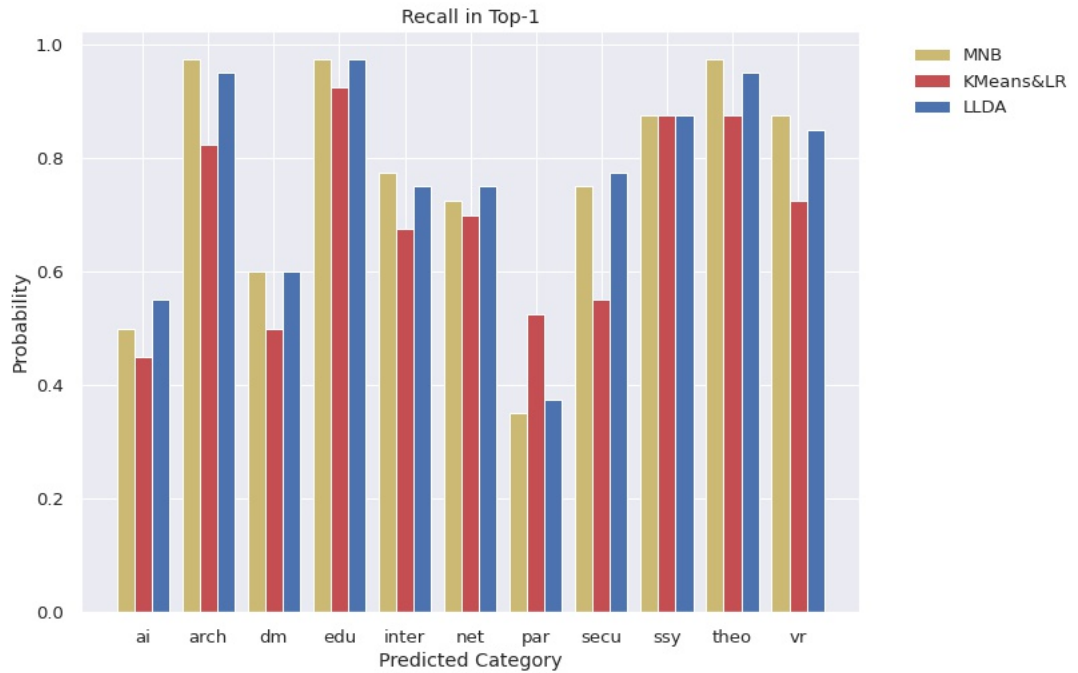


Figure 5.12: Comparison of Recall between models

⁵Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review:
<https://ieeexplore.ieee.org/document/9173706>

The plot presented in Figure 5.12 implies that the Multinomial Naive Bayes and Labeled-LDA models performed significantly better than the KMeans & LR model in most instances when we predicted the primary category, except for the topics *Parallel Distributed System* and *Software Systems*. When compared to the other two models, *Parallel Distributed System* seems to achieve a higher probability in KMeans & LR, while *Software Systems* did not appear to have any differences between the models.

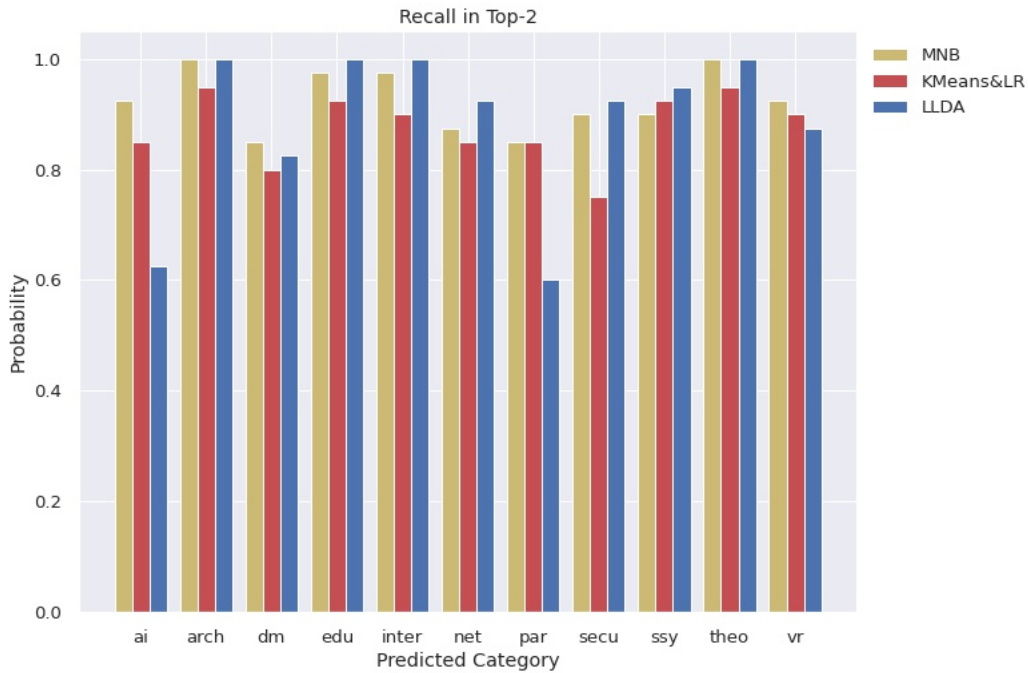


Figure 5.13: Comparison of Recall between models in Top-2

In most cases, Labeled-LDA and Multinomial Naive Bayes are indistinguishable in terms of their accuracy rates. As a consequence, we can consider the outcome of the Top-2 category. When the Top-2 category is taken into account, Labeled-LDA model was the best model in *Artificial Intelligence* originally, but now it has become the worst. In contrast, Multinomial Naive Bayes achieves possibly the best performance in this category. In addition, the Labeled-LDA model still acts poorly in *Parallel Distributed System*, but Multinomial Naive Bayes seems to obtain a similar probability to KMeans & LR. The Multinomial Naive Bayes model seems to be very stabilised; it acquires a high probability in almost all categories. The distribution of *Software Systems* has slightly changed, where it seems to be the optimal model among the three

models.

Chapter 6

Conclusions and future work

6.1 Conclusion

The main objective of the research is to construct system that helps academics to find examiners for students' reports. Several models using machine learning techniques are implemented to discover the most appropriate thesis topics automatically. The experiments were applied with the Multinomial Naive Bayes model, K-Means Clustering with Multinomial Logistic Regression model, and Labeled-LDA model on our manually collected academic papers dataset. The result of this study suggests that Multinomial Naive Bayes and Labeled-LDA have better precision than KMeans & LR in general, where they have a similar probability distribution of the classification among all topics when predicting the primary category. Furthermore, Multinomial Naive Bayes still performs well when the Top2 labels have been considered. In contrast, KMeans & LR seems to have the lowest level of stability and the most inaccurate prediction among the three models, although its accuracy improves significantly when considering the top2 category. Labeled-LDA also have a high level of accuracy, but it takes the longest computational time. Hence, this research indicates that the Multinomial Naive Bayes model has an excellent performance in classifying academic papers.

The application, PaperSystem, has been designed for this project, where it develops by Java GUI through our customized Hyperledger Fabric Network, which is friendly to the user. The data are visible and composed by the Hyperledger Explorer. One of the strengths of current work is that PaperSystem allows authorized organizations such as university postgraduate offices or student's supervisor to rapidly seek out a group of users with relevant experience and interests and communicate with each other. In addition, this not only reduces the time required to find potential academic reviewers but also saves the effort required for human resources for labelled thesis manually, where the most appropriate classification result could be provided from the Multinomial Naive Bayes model with a high level of prediction rate.

6.2 Limitation

A number of limitations need to be noted regarding the present study. PaperSystem is a prototype launched by the fabric network, whereas Hyperledger Fabric is a platform that is not yet fully developed. The configuration of each component needs to be identified and set up before activating the network, such as the number of organisations and their relevant number of peers. However, it is a tedious and difficult task to specify those parameters because they interact with each other; any parameter not set up probably may lead to a failed network establishment. It usually takes lots of time for the researcher to figure out the relationships with their corresponding version configurations. We have tried to design a test network with five organisations. Although the network architecture has been created and the Chaincode has been installed successfully, we could not mobilise the functions of the Chaincode; one possible reason could be that our computer was not configured with enough memory for the network to function. Furthermore, PaperSystem has been established through the private local host, which means only the user under the same IP address can access and use the system. In the meanwhile, if the test network has been successfully started, but the IP address of the same computer device has been changing during the development of the programme, our application will not be able to call the Chaincode function, which means that it cannot connect to the test network. Although the study has successfully demonstrated the feasibility of PaperSystem, It has some limitations in terms of model and system integration, while the predicted model and the design communicated system are currently separated in our study.

6.3 Future Work

This research has thrown up many questions in need of further investigation. Although the study has successfully demonstrated the feasibility of PaperSystem, It has some limitations in terms of model and system integration, while the predicted model and the design communicated system are currently operating separately in our study. Considerably more work could be done for the model and the designed system in a series of well-developed pipelines. Another question raised by this study regarding the configuration of the network could be to develop a simpler system based on the current PaperSystem, which can simplify the fixed configuration and make it compatible between operating systems and test networks in further research. Moreover, this simpler system could be constructed on the public host and run on a virtual machine in terms of the IP address problem.

Bibliography

- [1] “Too much academic research is being published. [online].” [https://www.universityworldnews.com/post.php?story=20180905095203579#:~:text=No%20one%20knows%20how%20many,million%20articles%20published%20each%20year.,07 September 2018](https://www.universityworldnews.com/post.php?story=20180905095203579#:~:text=No%20one%20knows%20how%20many,million%20articles%20published%20each%20year.,07%20September%202018).
- [2] P. Larsen and M. Von Ins, “The rate of growth in scientific publication and the decline in coverage provided by science citation index,” *Scientometrics*, vol. 84, no. 3, pp. 575–603, 2010.
- [3] M. Thangaraj and M. Sivakami, “Text classification techniques: A literature review,” *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 13, p. 117, 2018.
- [4] P. Cunningham, M. Cord, and S. J. Delany, “Supervised learning,” in *Machine learning techniques for multimedia*, pp. 21–49, Springer, 2008.
- [5] “Topic modeling: An introduction.” <https://monkeylearn.com/blog/introduction-to-topic-modeling/>, Sep 2019.
- [6] C. Mo, J. Yin, I. C.-H. Fung, and Z. T. Ho Tse, “Aggregating twitter text through generalized linear regression models for tweet popularity prediction and automatic topic classification,” *European Journal of Investigation in Health, Psychology and Education*, vol. 11, no. 4, pp. 1537–1554, 2021.
- [7] P. S. B. Ginting, B. Irawan, and C. Setianingsih, “Hate speech detection on twitter using multinomial logistic regression classification method,” in *2019 IEEE International Conference on Internet of Things and Intelligence System (IoTIS)*, pp. 105–111, IEEE, 2019.
- [8] A. M. Ramadhani and H. S. Goo, “Twitter sentiment analysis using deep learning methods,” in *2017 7th International annual engineering seminar (InAES)*, pp. 1–4, IEEE, 2017.

- [9] J. Chen, H. Huang, S. Tian, and Y. Qu, "Feature selection for text classification with naïve bayes," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5432–5435, 2009.
- [10] J. Kolluri and S. Razia, "Text classification using naïve bayes classifier," *Materials Today: Proceedings*, 2020.
- [11] A. McCallum, K. Nigam, *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, Citeseer, 1998.
- [12] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
- [13] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 conference on empirical methods in natural language processing*, pp. 248–256, 2009.
- [14] J. Mcauliffe and D. Blei, "Supervised topic models," *Advances in neural information processing systems*, vol. 20, 2007.
- [15] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.
- [16] W. Li, L. Sun, and D.-K. Zhang, "Text classification based on labeled-lda model," *CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION*-, vol. 31, no. 4, p. 620, 2008.
- [17] D. Forsyth, *Probability and statistics for computer science*. Springer, 2018.
- [18] M. Li, K. Wu, and L. Chen, "An analysis on the weibo topic detection based on k-means algorithm," in *2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, pp. 1328–1331, IEEE, 2022.
- [19] M. M. Haider, M. A. Hossin, H. R. Mahi, and H. Arif, "Automatic text summarization using gensim word2vec and k-means clustering algorithm," in *2020 IEEE Region 10 Symposium (TENSYP)*, pp. 283–286, IEEE, 2020.
- [20] R. Liang, Y. Li, X. Chen, and J. Chen, "Patent trend analysis through text clustering based on k-means algorithm," in *2020 International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, pp. 115–118, IEEE, 2020.
- [21] D. Marutho, S. H. Handaka, E. Wijaya, *et al.*, "The determination of cluster number at k-mean using elbow method and purity evaluation on headline news," in *2018 international seminar on application for technology of information and communication*, pp. 533–538, IEEE, 2018.

- [22] L. Zahrotun, N. H. Putri, and A. N. Khusna, "The implementation of k-means clustering method in classifying undergraduate thesis titles," in *2018 12th international conference on telecommunication systems, services, and applications (TSSA)*, pp. 1–4, IEEE, 2018.
- [23] W. Yongli, Y. Huanhuan, G. Xiaoze, S. Shurong, *et al.*, "Qh-k algorithm for news text topic extraction," in *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 610–614, IEEE, 2018.
- [24] H. Tao, J. Li, T. Luo, and C. Wang, "Research on topics trends based on weighted k-means," in *2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 457–460, IEEE, 2017.
- [25] H. Liang, X. Sun, Y. Sun, and Y. Gao, "Text feature extraction based on deep learning: a review," *EURASIP journal on wireless communications and networking*, vol. 2017, no. 1, pp. 1–12, 2017.
- [26] A. I. Kadhim, "Survey on supervised machine learning techniques for automatic text classification," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 273–292, 2019.
- [27] S. M. Mohammad, "Examining citations of natural language processing literature," *arXiv preprint arXiv:2005.00912*, 2020.
- [28] A. Guo and T. Yang, "Research and improvement of feature words weight based on tfidf algorithm," in *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pp. 415–419, IEEE, 2016.
- [29] M. P. Grootendorst, "ctfidf - bertopic." <https://maartengr.github.io/BERTopic/api/ctfidf.html>.
- [30] "what is blockchain technology." <https://www.ibm.com/topics/what-is-blockchain>. [Online; accessed 2022-08-26].
- [31] "what is hyperledger fabric." <https://aws.amazon.com/blockchain/what-is-hyperledger-fabric/>. [Online; accessed 2022-08-26].
- [32] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, (New York, NY, USA), Association for Computing Machinery, 2018.
- [33] "Chaincode tutorials, [online]." <https://hyperledger-fabric.readthedocs.io/en/release-1.4/chaincode.html>, 2019.

- [34] “Hyperledger-fabric.readthedocs.io. 2022. building your first network — hyperledger-fabricdocs main documentation. [online].” https://hyperledger-fabric.readthedocs.io/en/latest/build_network.html, 2019.
- [35] “Blockchain network [online].” <https://hyperledger-fabric.readthedocs.io/en/release-2.1/network/network.html>, 2019.