Tribhuvan University
Institute of Science and Technology

A Project Report on
News Classification: Comparing PLSA and LSA as Feature Extraction Techniques

for the partial fulfillment of requirement for the degree of
Bachelor of Science in Computer Science and Information Technology
(BSc.CSIT)

SUBMITTED BY:
Badri Thapa
Dipesh Neupane
Nirmal Gelal

SUBMITTED TO:
Patan Multiple Campus
Department of CSIT
Patandhoka, Lalitpur

October 7ᵗʰ 2021

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by Badri Thapa [1508/074], Dipesh Neupane [15016/074] and Nirmal Gelal [15032/074] entitled "News Classification and Comparing PLSA and LSA as Feature Extraction Techniques" in partial fulfillment of this requirement for the degree of **Bachelor in Science in Computer Science and Information Technology (BSc. CSIT)** be processed for evaluation.

. . . .
Bikash Balami
Supervisor

LETTER OF APPROVAL

This is to certify that this project prepared by Badri Thapa, Dipesh Neupane and Nirmal Gelal entitled "News Classification: Comparing PLSA and LSA as Feature Extraction Techniques" is in partilal fulfillment of the requirement for **Bachelor in Science in Computer Science and Information Technology (BSc. CSIT)** has been well studied. In our opinion, it is satisfactory in the scope and quality for the required degree.

ACKNOWLEDGEMENT

ABSTRACT

Classification has been an important challenge in the ongoing research in the context of Nepali Texts. With the abundancy of the Nepali language texts in recent years - mainly due to the availability of internet and popularity of it, there has been a number of literatures been written to tackle it.

Although, in languages like English this is not a new domain of research, it is in early-stages in Nepali Language. And, with a large number of unstructured texts available in the internet, classifications tasks are difficult to achieve. A main problem to achieve a proper classification is to convert the unstructured documents into well-structured relevant format.

The purpose of this project is to find a automatic feature extraction techinque that can represent the document well enough so that the classificaiton can be done easily and with a relatively small data sample. Latent Semantic Analysis and Probabilistic Latent Semantic Analysis are two of the old-school feature extractor which are compared in this project and want to gain insights on which can be an optimum techinque for extracting features from the Nepali texts.

List of Abbreviations

k-NN .... K Nearest Neighbors
LSA .... Latent Semantic Analysis
PLSA .... Probabilistic Latent Semantic Analysis
SLSA .... Supplemented Latent Semantic Analysis
SVD .... Singular Value Decomposition
SVM .... Support Vector Machines
TF-IDF .... Term-frequency Inverse-Document Frequency

List of Figures

Contents

Chapter 1

Introduction

1.1   Introduction

Human beings have used text as a form of communication of their thoughts and knowledge from ancient times. All over the world, most civilizations have developed their some form of textual symbols (alphabets), and their own grammers. And Nepal is no exception to that. Nepali language evolved from Sanskrit, and the 36 consonants, 12 vowels and 10 numerals also known as Devanagari found in Nepali writings is also found in the Sanskrit language. These characters have their own variants, along with some special characters. All in all, Nepali language is a rich and complicated language with interesting grammer and symbols to write it with.

Nepali writings has had a significant rise in internet in recent times. With that, there is need of a automatic processing of such writings. So, in this project, the news articles in Nepali language will be classified into different topics (categories) and also identify keywords associated with the topic. It is also a problem of document classification which is the task of assigning a document to one or more categories. It is a supervised learning task.

The idea of document classification problem is to assign the document to one or more class. The class maybe the abstract topic a collection of document represents. The topic may be philosophical, or, scientifically represetative of the collection of the documents. There are many techinques for autoamtic classification of documents, such as, Expectation Maximization, Naive Bayes Classifier, TF-IDF, LSA, Support Vector Machines, neural networks, etc, and are used in spam filtering, email routing, language identification[1].

1.2   Problem Statement

Document Classification and topic modeling is a classic problem in machine learning. It has had massive research and there have been many procedures described and constructed to discover solutions to such problems. with a significant rise in Nepali text/articles in the internet, a lot of problems may come while sorting the articles and texts. And Latent Semantic Analysis and Probabilistic Latent Semantic Analysis is one of those techniques that attempts to cluster the documents in an unsupervised way. And it can also be a powerful feature extraction techinque which can then be used for document classification using a supervised learning method. Although, they are quite similar they incorporate different ideas and have their own advantages. So, by applying these techniques for document classification, the advantages and disadvantages of these techniques can be realized.

1.3   Objectives

The aim of this project is find out if LSA or PLSA can be good tool for extracting features the Nepali news articles topic-wise.

- Learn about Latent Semantic Analysis and Probabilistic Latent Semantic Analysis application on Nepali texts.

- Performance of LSA and PLSA on Nepali texts.

- Find the better feature extracting tool between LSA and PLSA.

1.4 Scope and Limitations

1.4.1 Scope

The project was developed to find a computationally fast way to automatically extract features from the text corpus. The features must be representative of the text corpus and should enable for better precision for classification tasks. Also, the project should establish a proper preprocessing steps for better feature extraction.

1.4.2 Limitations

The project contains some limitation such as a single document cannot be used for the feature extraction using these techniques. Stemming of the Nepali texts is not reliable. Due to typos, many words with significance are lost. Some descrepancies in Nepali Unicode values causes some punctuations hard to remove. Large number of documents is very hard to parse through and process due to limited computational devices.

1.5 Development Methodology

Extracting features from the text corpus so that the computer can understand and learn from it consists of well-described series of steps which are:

Dataset Collection

A scrapper was built and used to collect the news articles from various websites like Ekantipur, Ratopati, Setopati, ReportersNepal and OnlineKhabar. A total of 12 categories were scrapped off these websites namely: Business, Education, Entertainment, Health, Interview, Literature, National, Opinion, Sports, Technology, Tourism, and World. Although, only 10 categories were only used and Opinion and Interview were discarded.

Preprocessing

To preprocess the text means bringing the text into a form that is predictable and analyzable for any task. The preprocessing of the text in our case makes it ready to be used in training and testing of the machine learning model. Preprocessing is mainly done to reduce the noise the text that helps to imporve the performance of the classifier and speed up the clsssifcation process, thus aiding in real time news classification. The text preprocessing may include: lowercasing, stemming, tokenization, special symbols and number removal(noise removal), lemmatization, stopword removal, normalization, and some others. Here, following operations are carried out for preprocessing of the text:

1. **Tokenization:** It is one of the common taks in Natural Language Processing. It is a way of separating a peice of text into smaller units called tokens. It can either be words, characters, or subwords. Here, a word is taken as a single token. It is mostly separated by a white spaces, or using some special symbol.

2. **Special symbol and number removal:** Special symbols like $!, |, \backslash, \div, \%, \$, -,$ $(,), /, <, >, =, +,$ ˆ, etc. and numbers, those which do not have much importance in classification, are removed.

3. **Stop word removal:** Stop words are such words which are too frequent among the documents in the collection. Such words which occur too frequently are useless for purposes of retrieval. They are normally filtered out. Articles, prepositions, and conjunctions are natural candidates for a list of stopwords.

4. **Stemming:** It is the process to reduce a word into its stem. A stem is the portion of a word which is left after the removal of its affixes (i.e. prefixes and suffixes). The stemmer used is a developed module found in `https://pypi.org/project/nepali-stemmer/`.

1.5.1   Exploring the preprocessed Data

After the preprocessing the data was then analyzed. It was seen that there were a lot of typos in the texts.
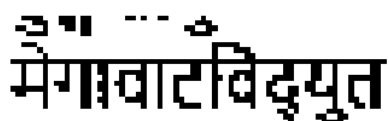
Figure 1.1: **A single word which should be three different words**

Figure 1.2: **A single word which should be two different words**
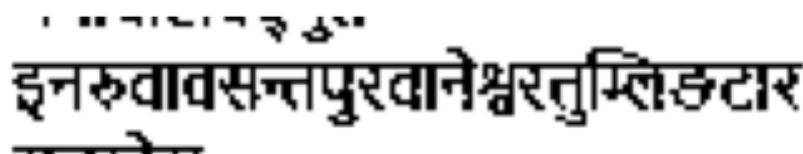
Figure 1.3: **A single which should be many different words**

Another kind of typos that was encountered - same word is written in different ways. This main happened because of people writing the foreign words in Nepali. So, the spelling for the same word was different.

This led to counting of the distinct words in the corpus. The unique word count was too high and all the words which had the word count of 1 in the corpus was deleted. Then, we randomly deleted half the corpus since the document-term matrix would not fit in the memory of eight gigabytes.

Data Preparation

After preprocessing the data was divided into several sets of different sizes for more comparative studies. There were four sets of 1600, 1200, 1000 and 500 documents. Each sets of documents were fed into the feature extraction pipeline and then into the machine learning algorithms for comparison.

(a)   (b)   (c)   (d)
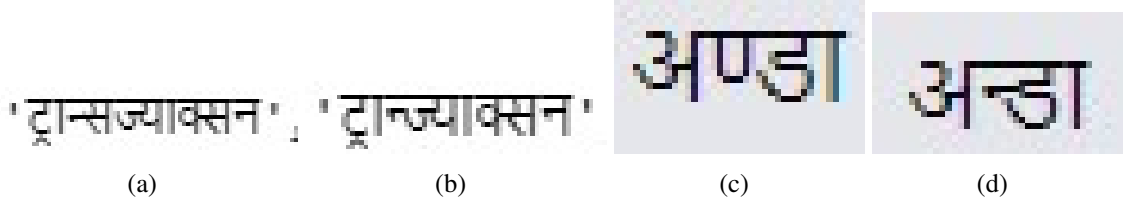
Figure 1.4: **(a) and (b) is the same word but has different spellings making them distinct and same with (c) and (d)**

Feature Extraction

After the initial text is cleaned and normalized, it is required to extract the features to be used for modeling. These words that are within our documents are assigned weights before modeling them.

In order to cluster and then classify the document, the document must be represented into a mathematical form. There are some ways to represent documents and the three classic models are, namely, Boolean, the vector, and the probabilistic models[2]. In this project, the vector space model is used[3] to convert the document into a mathematical form. These vectors are formed using the terms *t* occurring in the documents as the elements of the document vector. The simplistic approach to represent the document and the ability to do partial matching and calculating the degree of similarity between documents makes it a very optimal choice.
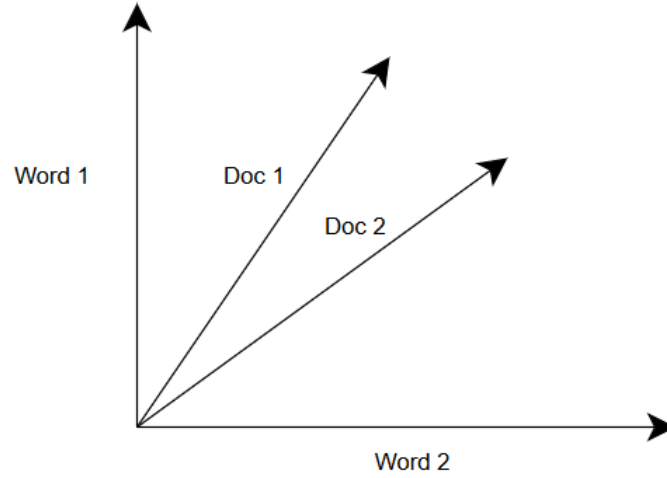
Figure 1.5: **The vector representation of document.**

Formally, if *t* different terms are present in the document and each document is represented by a t-dimensional vector, then

$$D_i = \{d_{i1}, d_{i2}, ..., d_{it}\},$$

d_{ij} representing the "weight" of the jth term[3].

And, the similarity between the two documents can be then calculated as:

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,q}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,j}^2}} \qquad (1.1)$$

where, d_j and q are the two documents as t-dimensional vectors, $w_{i,j}$ and $w_{i,q}$ are the weights of the terms, and the calculated value is known as the *cosine of the angle* between these vectors[2]. And the $|\vec{d_j}|$ and $|\vec{q}|$ are the norms of the documents.

The easiest way to represent individual words is the numerical representation as it's easy for the computer to process numbers. Some of the extractions techniques are Bag of words, TF-IDF, One-hot Encoding, and word embeddings. Here TF-IDF[4] is used as the feature extraction technique and the document is transformed into a vector form. To represent Nepali news in vector form, the TF-IDF weighting value for each word in the text is taken as a high dimensional value in a vector. It is calculated as,

$$W_{t,d,D} = tf_{t,d} * idf_{t,D} \tag{1.2}$$

where,
$tf_{t,d} = \frac{f_{t,d}}{max\{f_{t',d}:t' \in d\}}$
$idf_{t,D} = \log \frac{N}{1+|d \in D:t \in d|}$
and,
tf = term frequency
idf = Inverse document frequency
$f_{t,d}$ = Number of term t in document d
$max\{f_{t',d} : t' \in d\}$ = max occurring term t' in document d .
N = Total number of document in the corpus D .
$|d \in D : t \in d|$ = Number of documents where the term t appears.

All this can be possible to be done only after the documents are subjected to the technique of Latent Semantic Analysis (LSA)[5] which reduces the original term-document matrix **D** into a filtered term-document matrix. Here, the term-documents matrix can be the collection of vector of the documents. The vector may be a row or a column of the matrix. This method uses a mathematical technique called *Singular Value Decomposition* to identify patterns in the relationships between the terms and concepts contained in the collection of texts. It represents the meaning of a word as a kind of average of the meaning of all the passages in which it appears, and the meaning of a passage as a kind of average of the meaning of all the words it contains[6]. This method takes a large matrix (combination of vectors) of term-document data and constructs a "semantic" space wherein terms and documents that are closely associated are placed near one another.

The SVD decomposes the original matrix into three matices: a document-topic matrix, and Singular matrix, and a term-topic matrix. The SVD for a matrix(rectangular) **D** can be given as:

$$\mathbf{D} = \mathbf{U\Sigma V^T} \tag{1.3}$$

where **U** with $t \times r$ is the terms-topic matrix, Σ with $r \times r$ size is the array of Singular values and the $\mathbf{V^T}$ with $r \times d$ is the document-terms matrix. The singular values show the relative importance of our topic. **U** and $\mathbf{V^T}$ are orthonormal matrices and the Σ is a diagonal matrix which has values in descending order in the diagonal. Intuitively, in linear algebra, the three matrices rotate (**U** and $\mathbf{V^T}$) and scale (Σ) the space and is the same as doing the transformation with **D**. The process of decomposing into the three matrices will not be discussed. Although, the essence of SVD, that is, the dimensionality reduction, is to be discussed. Since the singular values describe how each latent concept explains the variance in our data, the number of values (concepts) has to be chosen so that the original matrix can be approximated with minimal error. This will be mapped into a lower-dimensional space. They will be a lower-dimensional approximation of the higher-dimensional space. The reason for this to do is that, the original matrix may be too large, sparse
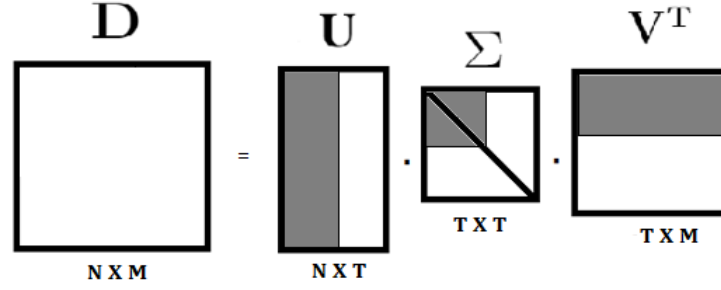
Figure 1.6: **The figure shows how the document-term matrix is decomposed into its subsequent three low rank matrices. The shaded portion shows how much values can be truncated and approximate the original matrix. Note: shaded portion is taken to approximate the original matrix D.**

and noisy. And it assumes that after rank lowering that some dimensions are combined and depend on more than one term. This will lead to merge the dimensions associated with terms that have similar meanings. And conversely, the terms that have opposite or no association with them will either cancel out, or, at least, be smaller than that with similar meanings. This process of selecting the *k* largest singular values, and their corresponding singular vectors from $\mathbf{U}$ and $\mathbf{V^T}$ is known as the dimensionality reduction step. Here, the number of singular values chosen will be directly related to the number of concepts to be separated and defined. This finaly step will lead to many things. First, the comparison between the document will occur in the low-dimensional space. The terms related with each other will be found out. The importance of the term for the concept can be seen as well. Hidden word association can be analyzed as well. Overall, the following steps can be done for low rank approximation using SVD:

1. Given $\mathbf{D}$, construct its SVD in the form as $\mathbf{D} = \mathbf{U\Sigma V^T}$.

2. Derive from Σ the matrix Σ_k formed by taking the first *k* singular values and replacing others with 0.

3. Compute and output $\mathbf{D_k} = \mathbf{U\Sigma_k V^T}$ as the rank-k approximation to $\mathbf{D}$.[7]

Although, LSA is a relatively simple technique, it sure has some downsides. Such as the relative importance of a word cannot be found out from the decomposed matrix. It is also hard to infer what the numbers in LSA is telling us and can be negative values. With regard to that, another technique for topic modeling and document classification was introduced, called as probabilistic Latent Semantic Analysis. As the name suggests, it models co-occurrence information under a probabilistic framework in order to discover the underlying semantic structure of the data. It was developed in 1999 by Thomas Hofmann[8]. Initially, it was used for text-based application, however it has slipped onto other fields: such as computer vision or audio processing. Probabilistic latent semantic analysis is an expectation maximization-based mixture modeling algorithm[9]. There exists a generative process [9] inherently for dimensional reduction and builds the document-term matrix as follows (with Z_m as a topic, d_i as document and w_j as a word token):

- Select a document d_i with probability $P(d_i)$.

- Generate the probabilities $P(Z_m|d_i)$ for a latent class (topic) and $P(w_j|Z_m)$ for a word and put it in the relevant indices of the document-word pair (d_i, w_j) . Here, document and word is assumed to be *conditionally independent*.

The conditional independence between the words and the documents is an important assumption
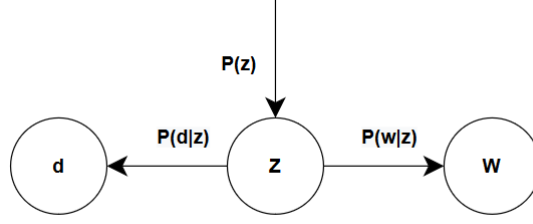
Figure 1.7: **The figures shows how the latent topic *z* can make a conditional independence between the document *d* and the words occurring in the document *w*. It is the graphical representation of the aspect model.**

in PLSA given the latent aspect in our case, the topic. Which actually means:

$$P(d, w) = P(d)P(w|d), P(w|d) = \sum_{z \in Z} P(w|z)P(z|d) \tag{1.4}$$

It means that the joint probability between the documents and words pair can be expressed in the following way:

$$P(d, w) = P(d) \sum_{z \in Z} P(z|d)P(w|z) = \sum_{z \in Z} P(z)P(d|z)P(w|z) \tag{1.5}$$

The conditional independence assumption makes it possible for the derivation of *EM algorithm*. Initially, the EM algorithm starts by initializing $P(Z_m), P(d_i|Z_m)$, and $P(w_j|Z_m)$ to 1/k, 1/n and 1/d, respectively. Here, k, n, and d denote the number of clusters (topics), number of documents, and number of words, respectively.

1. **E-step** The posterior probability is calculated using the equation:

$$P(z|d, w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z' \in Z} P(z')P(d|z')P(w|z')} \tag{1.6}$$

2. **M-step** The $P(Z_m), P(d_i|Z_m)$ and $P(w_j|Z_m)$ is calculated by:

$$P(w|z) \propto \sum_{d \in D} n(d, w)P(z|d, w), \tag{1.7}$$

$$P(d|z) \propto \sum_{w \in W} n(d, w)P(z|d, w), \tag{1.8}$$

$$P(z) \propto \sum_{d \in D} \sum_{w \in W} n(d, w)P(z|d, w). \tag{1.9}$$

These two steps are iteratively repeated until convergence. The convergence is done using the maximization of the log likelihood function:

$$\mathcal{L} = \sum_{d \in D} \sum_{w \in W} n(d, w) \log P(w, d) \tag{1.10}$$

Although the original paper [8] used a method called *Tempered Expectation Maximization* in order to avoid overfitting.

Also, the PLSA can be seen as a matrix factorization as well. The factorization of the co-occurrence of the document matrix can easily be seen using equation 1.5. It can also be seen as:

$$P = U \cdot \Sigma \cdot V \tag{1.11}$$

where each matrix means:

Figure 1.8: **The matrices can be representation of the probabilistic latent semantic analysis.**

- U contains the document probabilities $P(d|z)$.

- Σ has diagonal elements with of the prior probabilities of the topics $P(z)$.

- V contains the word probability $P(w|z)$.

- P means the $P(w, d)$.

The main advantage here of this representation over LSA is that the entries are normalized and have no negative values. And the value represents the importance of each entry for given topic.

The TF-IDF can only be used for LSA. Then, Singular Value Decomposition is applied. While for pLSA, Bag of Words is used for the creating a document-matrix form and estimate the $P(w, d), P(d|z), P(z$

Feature Extraction Details

In LSA, the number of topics had to be hand-picked and 10, 50, 100 were chosen for dimensional reduction. In pLSA, the algorithm was much more computationally expensive and only 10 was chosen for the number of topics.

Machine Learning

After the feature extraction has been done for both pLSA and LSA, supervised machine learning algorithms like Logistic Regression, Support Vector Machines and Random Forest Classifier are trained on the document-term matrix. Each set was divided into a 90-10 training-validation split. The validation for each was used to test the algorithm's performance on each set.

Logistic Regression   is a linear model for classification and is also known as logit regression, maximum-entropy classification or the log-linear classifier. In this model, the probabilisties describing the possible outcomes of a single trial are modeled using a logistic function.

A logistic function is a common S-shaped curve (sigmoid curve) with equation

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.12}$$

Support Vector Machines are a set of supervised learning methods used for classification, regression and outliers detections. A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. A good separation is achieved by the hyper-plane that has the largest distance to the nearest training datat points of any class of the classifier.

Random Forest Classifier is one of the ensemble methods that combines the predictions of several base estimators – decision trees in this case –built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

1.6 Project Roadmap

Chapter 2 details the words that has been carried out previously in the field of classification of texts in Nepali and other languages. Chapter 3 covers the different requirements like functional, non-functional and system requriement of the system. Design of the system is discussed in Chapter 4 with different diagrams. Implementation methods and system testing data and result are described in Chapter 5. The concluding Chapter 6, concludes the chapter explaining what have been done in the project and what further improvements could be done.

Chapter 2

Literature Study

2.1   Literature Review

Several notable papers have been written in classification of texts in Nepali and other languages. Only in recent times the classification of Nepali news articles have been done. All have mostly used their own datasets by scrapping the online news portals (in case of Nepali news). They are mostly supervised. The use of PLSA in Nepali language has still not been found.

Works related to other language   [10] used LSA as a data preprocessing method, and then used Independent Component Analysis (ICA) for the classification. They showed that using ICA and LSA together provides better classification for Chinese short-texts rather than using ICA alone.

[11] compared the LSA, SVM, and k-Nearest neighbor variations of the LSA models and showed that SVMs and k-NN LSA performed better than the other methods that they had compared to. They had evaluated the performance of the models using Mean Reciprocal Rank and argued that it is very well suited evaluation measure for text categorization tasks and the tests were performed on two different datasets (both in English language). They also showed that LSA performs better when it has many training examples. They also showed that the number of terms in the dataset do not influence the models' performance.

[12] used a modification of the LSA and compared with the traditional LSA. The modified LSA, called SLSA (supplemented latent semantic analysis) added extra information (the label of the document) in the original document matrix and performed SVD on it. It was shown that this added information had increased the accuracies on classification tasks by 1.14%, 1.3% and 1.63% for various term weighting schemes (tf, idf, tf-idf). The classification task was done on Hindi texts.

[13] also used another modification of LSA by providing supplementary information. They provided document category and domain information as supplementary information. They showed an improvement of 4.7% - 6.2% using this modification which shows that summary of the text can be additional informaion that can be fed into the model to make it better.

[14] used LSA for ambiguity checking in Bengali literature. They showed that the Vector Space Model and LSA gives generally reasonable outcome for Bengali language structure and sentence structure.

[8] introduced the probabilistic latent semantic analysis and used a Tempered Expectation Maximization algorithm and showed how it was significantly better than the LSA.

Works related to Nepali Language   In Nepali Language, several works have been done, such as, [15, 16, 17, 18, 19]. [15] proposed their own method where they used probability-based word embedding. And they showed that their method was the best classifier among other models.

[16] used TF-IDF method for feature extraction of Nepali news documents and classified using Naive Bayes, SVM and Neural Networks.

[17] performed a comparative study of various different document representation methods, including LSA, TF-IDF and word2vec. They showed that the accuracy of the TF-IDF with LSI model

is far superior than those other two models.

[18] used Bag of Words[20] of the Nepali news articles to train the deep learning network. [19] used TF-IDF for feature extraction of various Nepali texts and built a model using Logistic Regression, SVM, Multinomial Naive bayes, Bernoulli Naive Bayes, Nearest neighbor and some others as well.

In summary, a lot of works done in both Nepali and other languages were found and there were a lot of different methods applied. LSA had been extensively used in other languages but in Nepali texts, it was a rarity. Works in Nepali used TF-IDF and Bag of Words as a feature extraction and had a wide use of neural networks and other supervised classification techniques.

Chapter 3

System Analysis

3.1 Requirement Analysis

It defines the scope of the project in the Project Description. It describes the hardware and software components of a system required to develop and use software efficiently. The requirements may be functional and non-functional and both are essential to a successful software project. The following Functional and non-functional requirements were identified on the system.

Functional Requirement   The functional requirements defines function of the system where functions are the collective sets of input, their behaviors and their output. The application should calssify the atricle into a single topic from a group of different topics based on its contents.
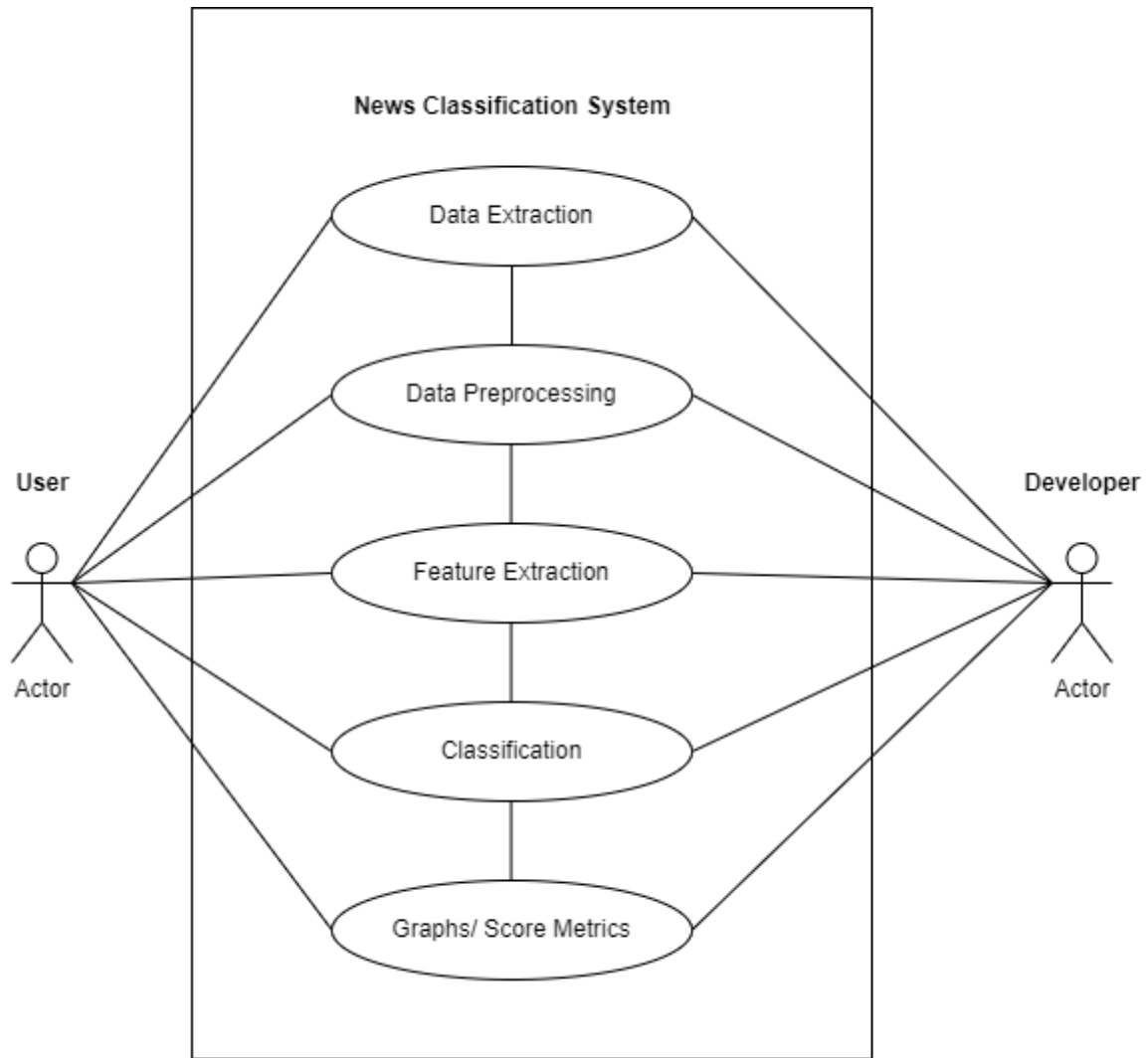
Figure 3.1: **Use-case diagram of the proposed system.**

Non-functional Requirement   Non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviour. The application should be consistent at classification whenever same article is passed into the pipeline.

Proper dataset should be built to train and extract features from for the classification.

3.2   Feasibility Study

3.2.1   Technical Feasibility

Technical analysis is concerned with determining how feasible a system is from a technical perspective. The application would require some degree of technical and coding expertise to run and understand the results.

3.2.2   Economic Feasibility

Economical analysis is the study that determines whether a system is economically acceptable or not. The application is economically feasible as there is no extra cost needed to run the program.

3.2.3 Operational Feasiblity

It is concerned with the operating capabilities of the system. It is concerned with operating capabilities of the system. It will require a decent computer with 8 GigaBytes of RAM and a modern CPU.

3.2.4 Project Timeline

Typically Schedule feasibility means estimating how long the system has taken to develop, and if it can be completed in a given time period using some methods like payback period.
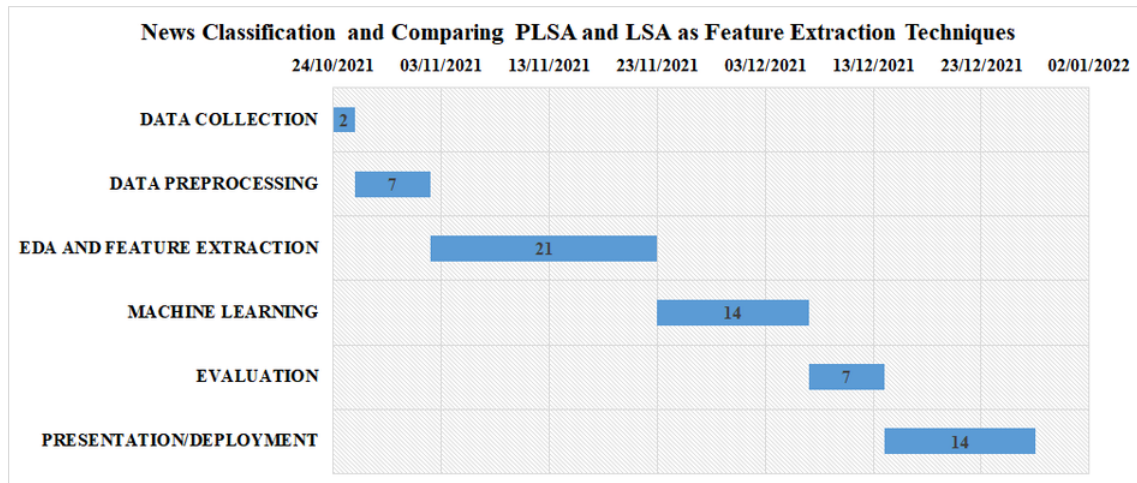

Figure 3.2: **The gantt-chart for building the proposed system.**

Chapter 4

System Design

4.1 System Architecture

This project takes the nepali texts as input and these inputs are preprocessed to generate appropritae condition to extract the feature from the texts. Data was cleaned by removing unnecessary punctuations, emojis and symbols. After the preprocessing, words with count one were removed as well. And the given data were vectorized using TF-IDF for LSA. While for PLSA the document-term matrix was used to get the better representation of texts.
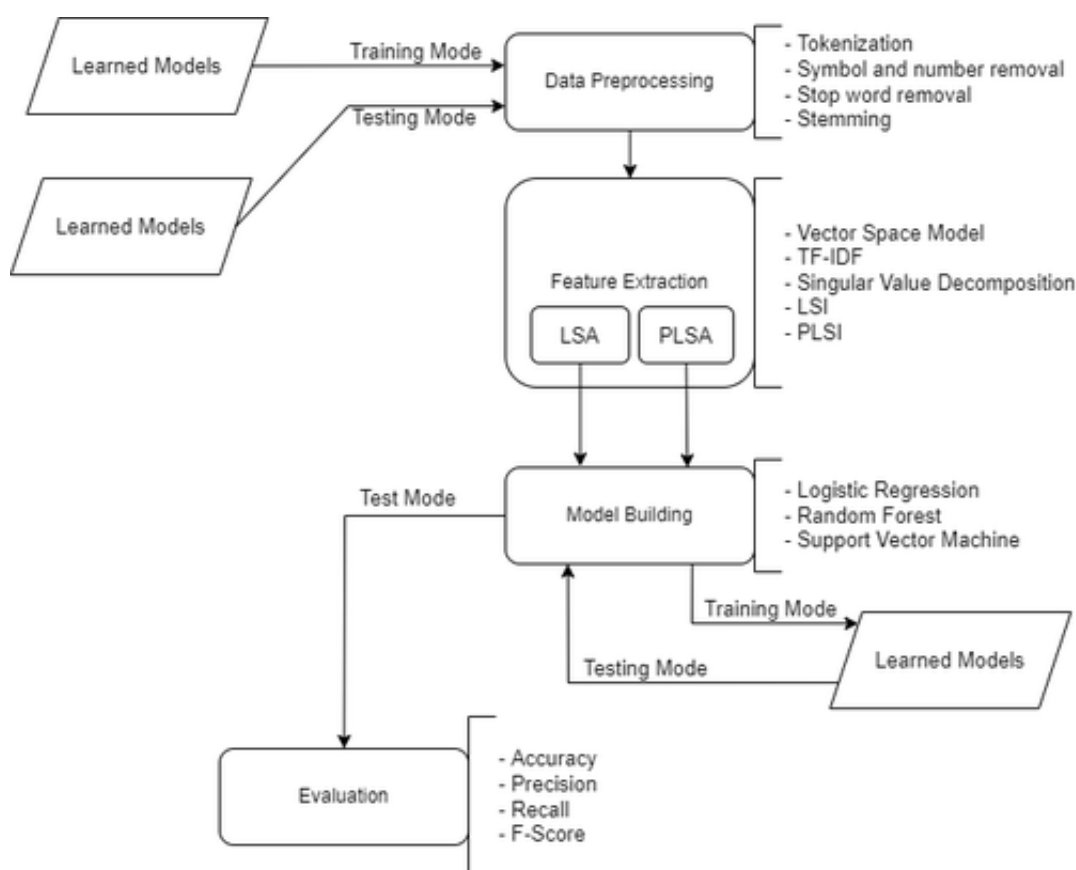

Figure 4.1: **The architecture of the Text Classification**
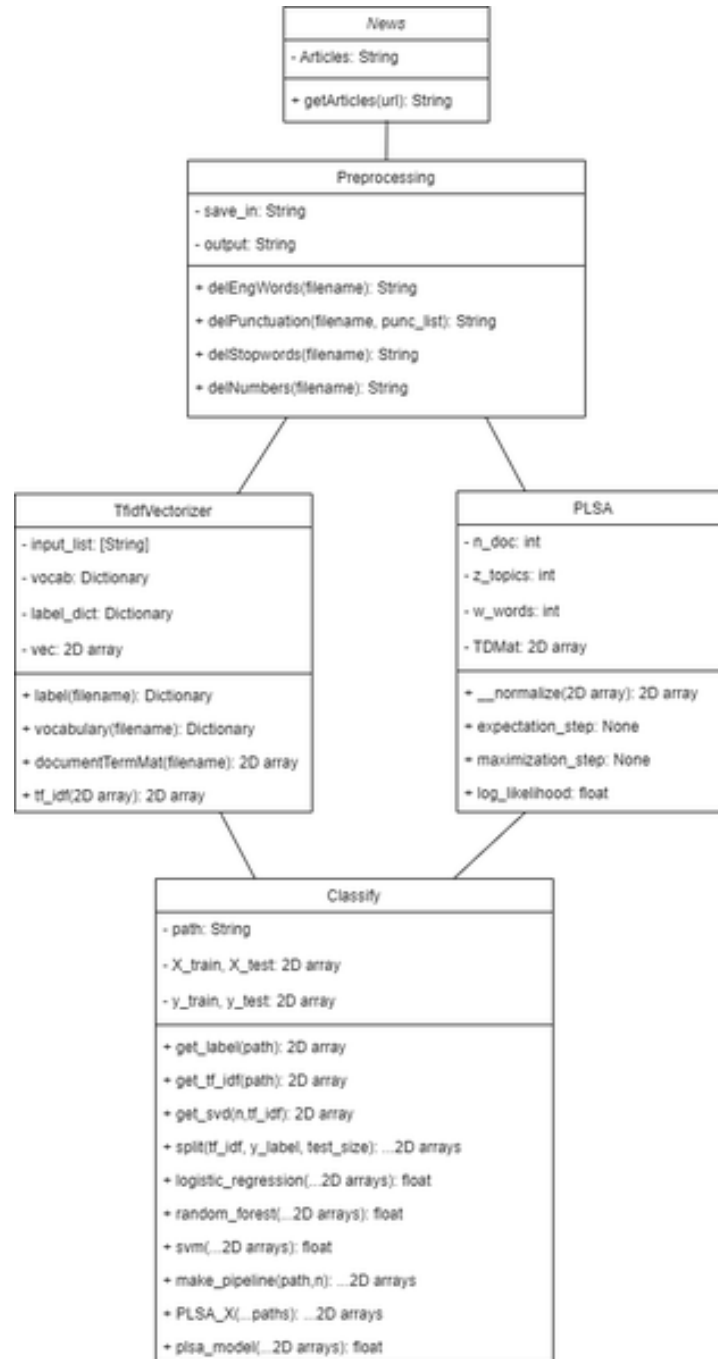
4.2 Process Design

4.2.1 Data Design

The Nepali texts used data structure rather than relational database. In this project we used program componenet level data design technique. We can use different data structure like Array, List,

Dictionary.

4.2.2 Class Diagram

The process design is as below:

News

- Articles: String

+ getArticles(url): String

Preprocessing

- save_in: String
- output: String

+ delEngWords(filename): String
+ delPunctuation(filename, punc_list): String
+ delStopwords(filename): String
+ delNumbers(filename): String

TfidfVectorizer

- input_list: [String]
- vocab: Dictionary
- label_dict: Dictionary
- vec: 2D array

+ label(filename): Dictionary
+ vocabulary(filename): Dictionary
+ documentTermMat(filename): 2D array
+ tf_idf(2D array): 2D array

PLSA

- n_doc: int
- z_topics: int
- w_words: int
- TDMat: 2D array

+ __normalize(2D array): 2D array
+ expectation_step: None
+ maximization_step: None
+ log_likelihood: float

Classify

- path: String
- X_train, X_test: 2D array
- y_train, y_test: 2D array

+ get_label(path): 2D array
+ get_tf_idf(path): 2D array
+ get_svd(n,tf_idf): 2D array
+ split(tf_idf, y_label, test_size): ...2D arrays
+ logistic_regression(...2D arrays): float
+ random_forest(...2D arrays): float
+ svm(...2D arrays): float
+ make_pipeline(path,n): ...2D arrays
+ PLSA_X(...paths): ...2D arrays
+ plsa_model(...2D arrays): float

Chapter 5

Implementation and Testing

5.1 Implementation

5.1.1 Tools used

Python, scikit-learn, numpy, matplotlib

5.2 Testing

The testing was done to find the possible flaws and the potential inefficiency of the system.

5.2.1 Unit Testing

Unit testing refers to the testing of every small modular components of the system, keeping them isolated from other modules. Here we mention testing result of the various part of the system. In unit testing, we design the whole system in modularized pattern and each module was tested. Test cases of system during feature extraction,

E.g.,

For document-term matrix we can see that:

अवसर पाउनेछन् प्रतियोगिता हुने नगद हजार रुपैयाँ पुरस्कार प्राप्त गर्नेछन् यस्तै हुने हजार रुपैयाँ हजार रुपैयाँ प्राप्त सक्ने कलेज जनाए

Figure 5.1: **Text for testing for document term matrix**

```
[[1. 1. 1. 2. 1. 3. 3. 1. 2. 1. 1. 1. 1. 1.]]
```

Figure 5.2: **Document term matrix for the given test text**

5.2.2 System Testing

In this testing phase our system as a whole was tested. Every individual component was integrated and tested.

5.2.3 Evaluation of the Model

Three different evaluation technique like: precision, recall and f-score were used. They are defined as follows:

Actual Values

	Negative	Positive
Negative	True Negative (TN)	False Positive (FP)
Positive	False Negative (FN)	True Positive (TP)

CONFUSION MATRIX

Figure 5.3: **Confusion matrix**

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F - measure = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{5.4}$$

In the first experiment, we lowered the latent-concept to 100. And then, different sized dataset of 1600, 1200, 1000, and 600 were trained and evaluated. The figure 5.4 to 5.7 explains the evaluation metrics for it. It showed random forest algorithm was more consistent which made sense as it is an ensemble techinque.

In the second experiment, we lowered the latent-concept to 50 which was evaluated on the same sets of data as befores. The figure 5.8 to 5.11 explains the evaluation metrics for it. Random forest had slight improvement in all evaluation scores. We can attribute this to lowerered sparsity of matrix.

Like in the previous experiment, SVM and Logisitic Regression fall behind significantly compared to random forest and scores for Random Forest was highest. 30 might be the optimal choice for feature extraction.

12 is clearly not suitable choice for feature extraction as shown in the figure 5.16 to 5.19. Almost all scores drop down to the bottom except random forest which showed robustness in different scenarios.
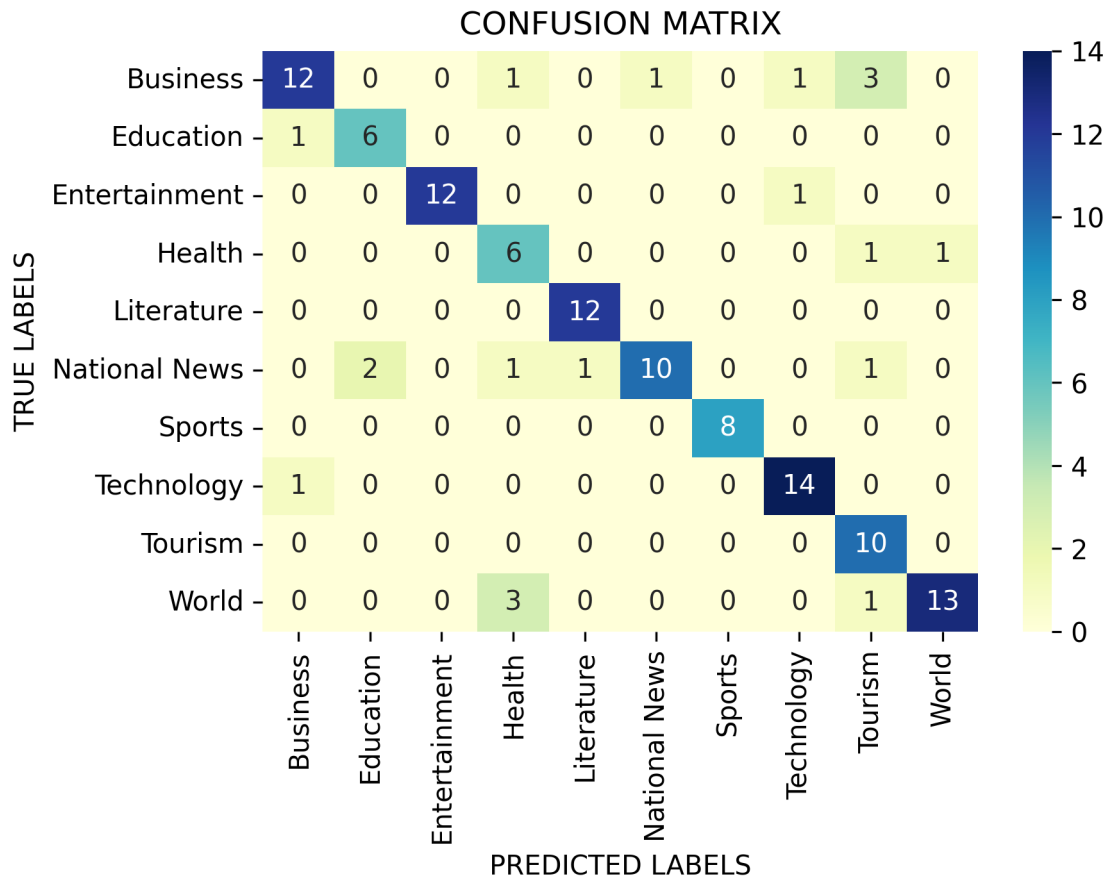
Figure 5.4: **Confusion matrix for LSA of set 3 and n = 30**

As PLSA is much more demanding in terms of computation operations and is much more time-consuming considering its iterative EM steps, calculating probability for each topic given document and word is rigorous on both memory and time complexity. So, PLSA performed worse compared to LSA for all machine learning algorithms, as the number of EM steps was not sufficient to maximize the expectation of the probablity for feature extraction. Also, the constraints on resources, more data could not be processed to utilize the full potential of PLSA. In most datasets, evaluation scores were not upto par with LSA. Even with the ensemble techinque, accuracy did not improve further than 50%.
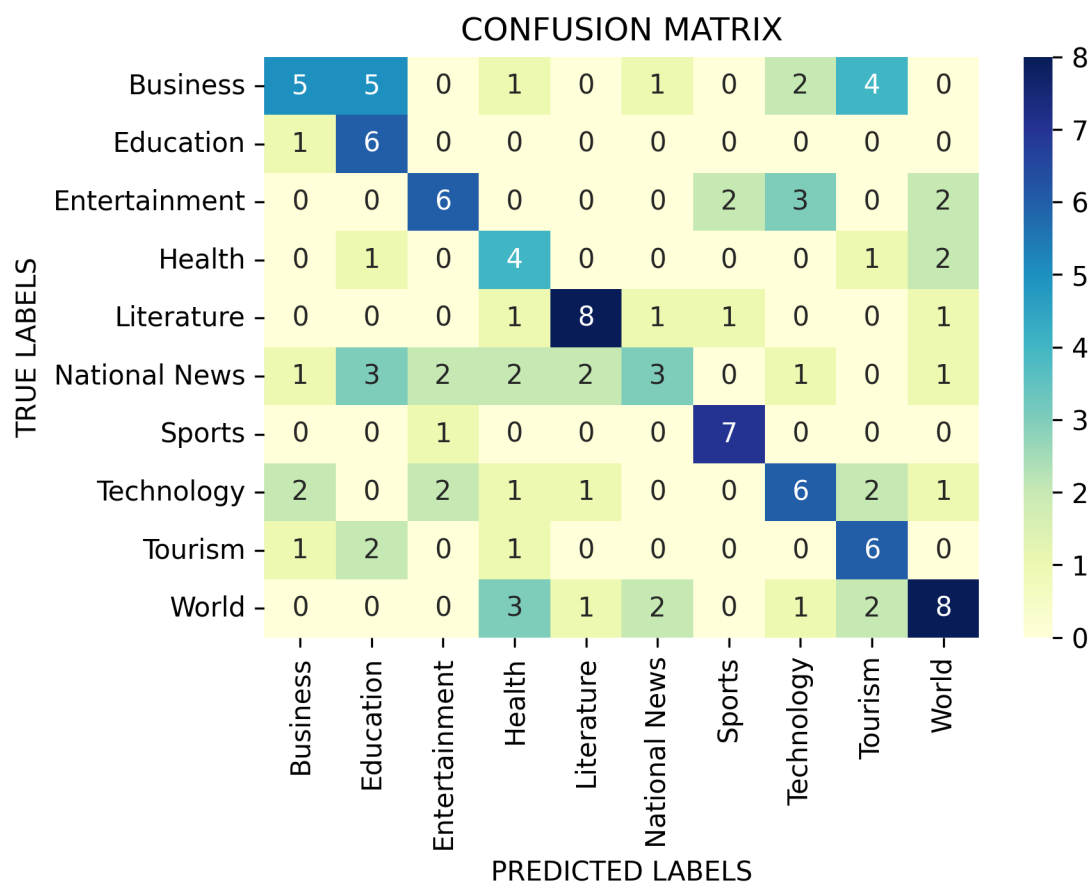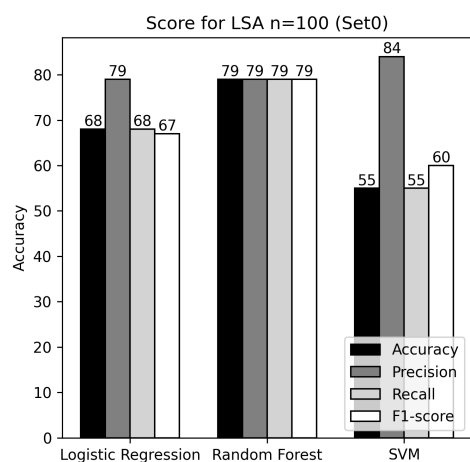
Figure 5.5: **Confusion matrix for PLSA of set 3**
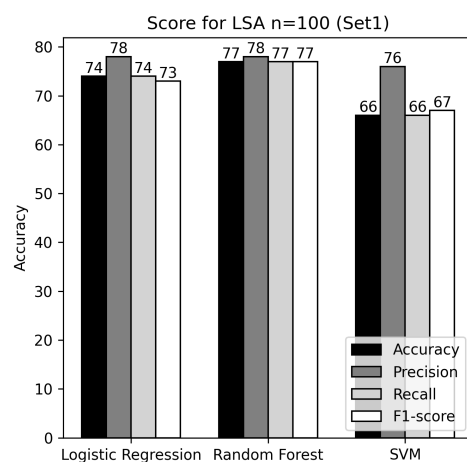

Figure 5.6: **Score for LSA n=100 (Set0)**

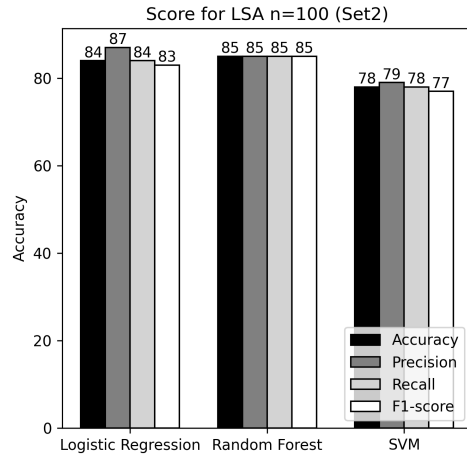Figure 5.7: **Score for LSA n=100 (Set1)**
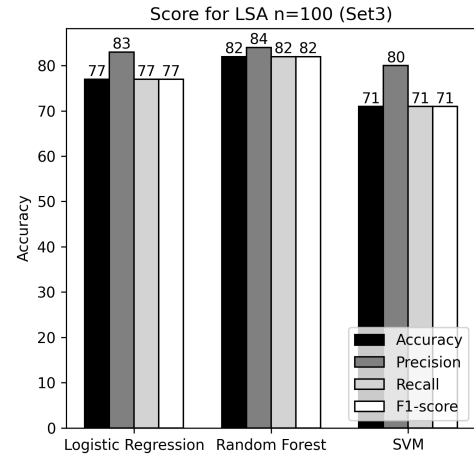
Figure 5.8: **Score for LSA n=100 (Set2)**

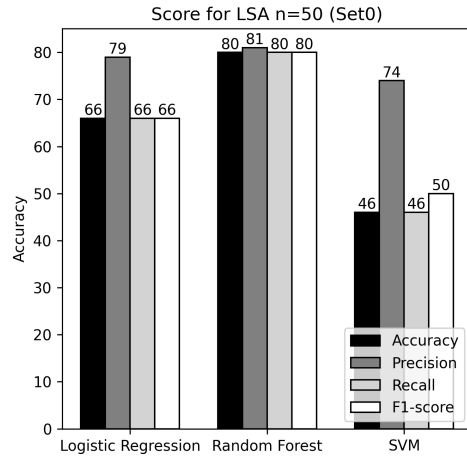Figure 5.9: **Score for LSA n=100 (Set3)**

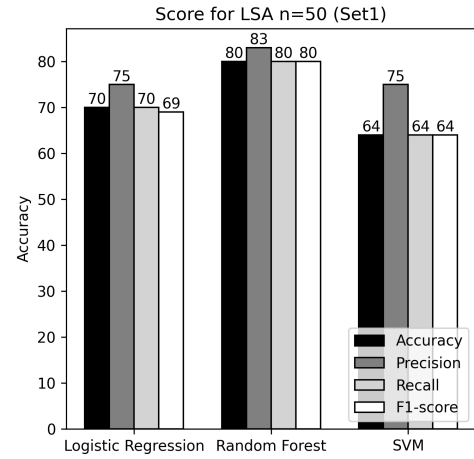Figure 5.10: **Score for LSA n=50 (Set0)**

Figure 5.11: **Score for LSA n=50 (Set1)**

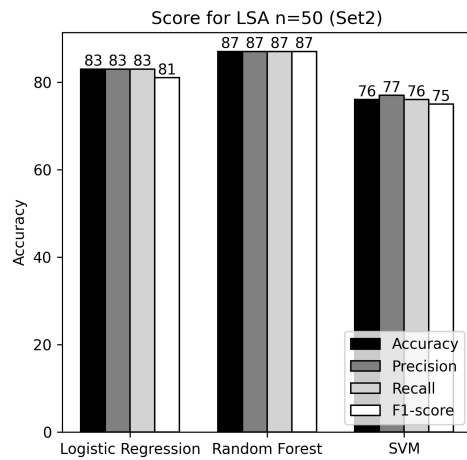Figure 5.12: **Score for LSA n=50 (Set2)**

Figure 5.13: **Score for LSA n=50 (Set3)**

21

Figure 5.14: **Score for LSA n=30 (Set0)**

Figure 5.15: **Score for LSA n=30 (Set1)**

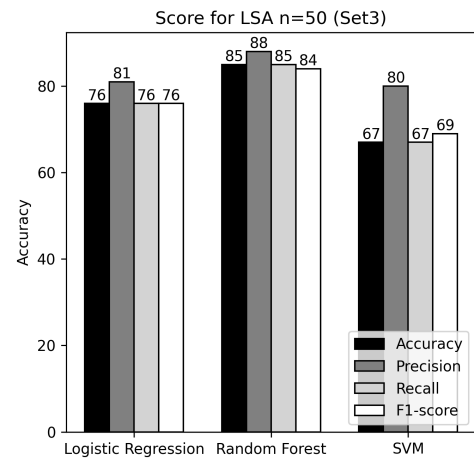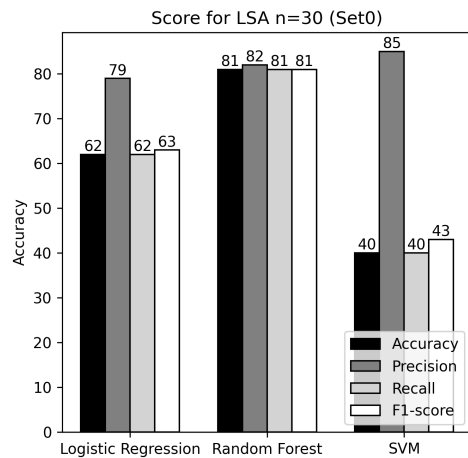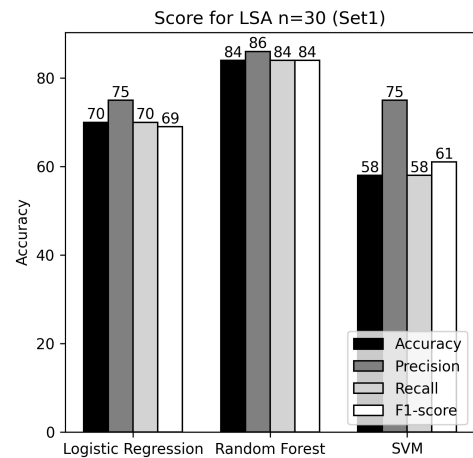Figure 5.16: **Score for LSA n=30 (Set2)**
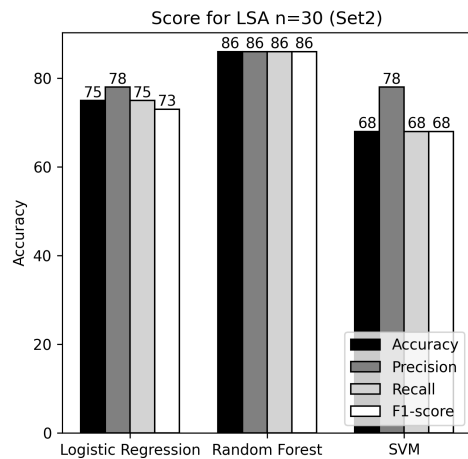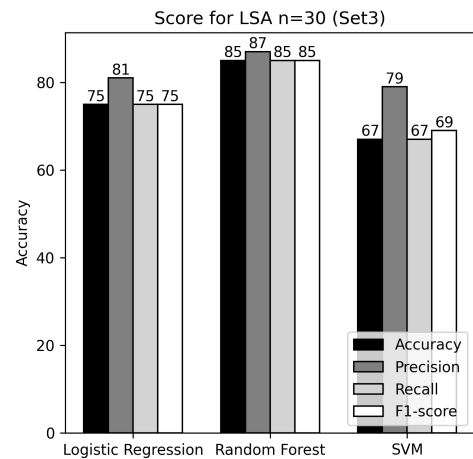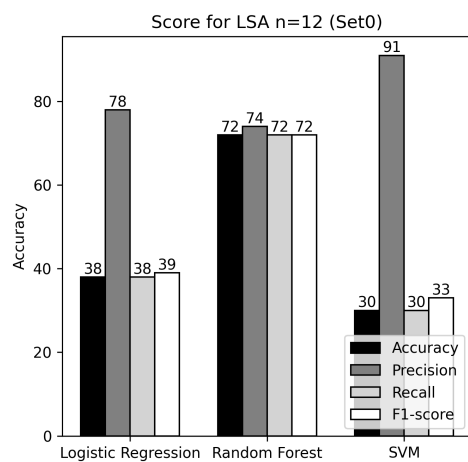

Figure 5.17: **Score for LSA n=30 (Set3)**

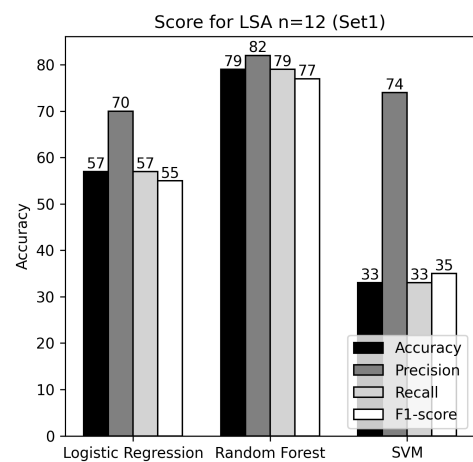Figure 5.18: **Score for LSA n=12 (Set0)**
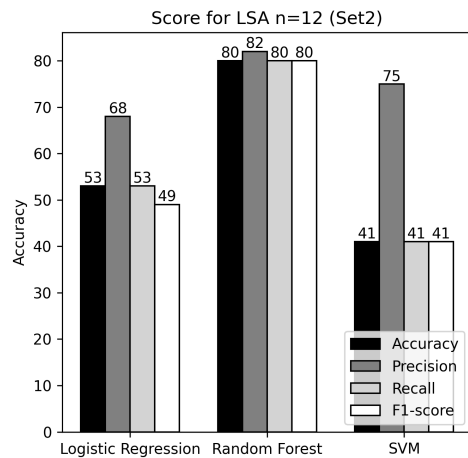

Figure 5.19: **Score for LSA n=12 (Set1)**

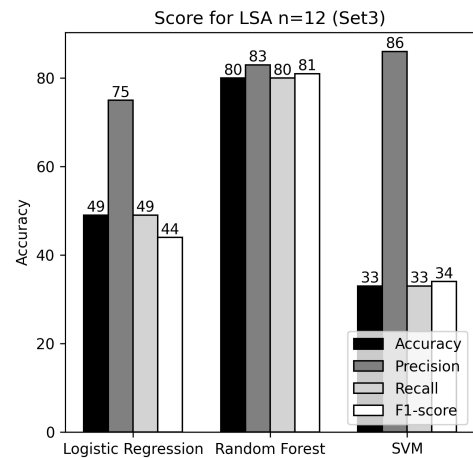Figure 5.20: **Score for LSA n=12 (Set2)**
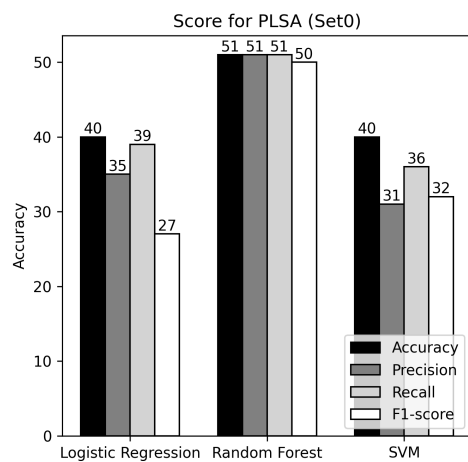

Figure 5.21: **Score for LSA n=12 (Set3)**

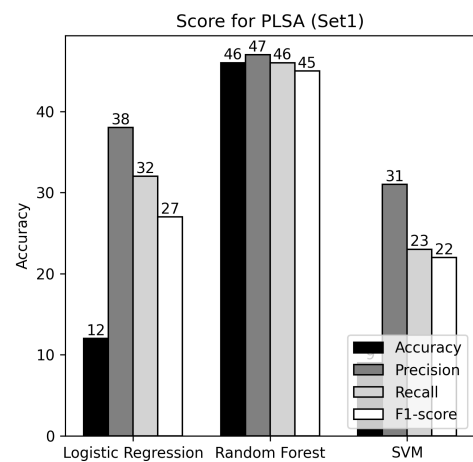Figure 5.22: **Score for PLSA (Set0)**
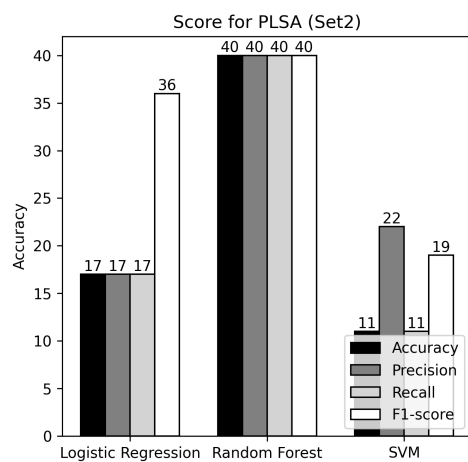

Figure 5.23: **Score for PLSA (Set1)**

Figure 5.24: **Score for PLSA (Set1)**
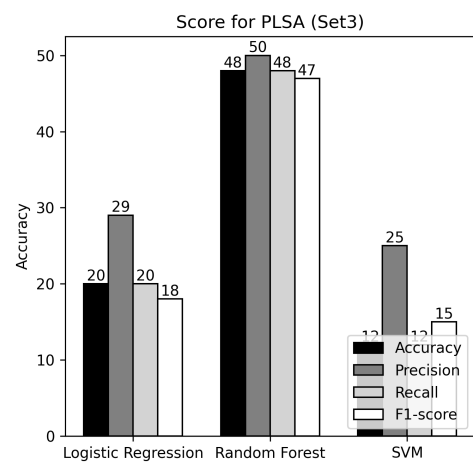
Figure 5.25: **Score for PLSA (Set3)**

Chapter 6

Conclusion and Future Recommendations

6.1   Conclusion

Hence, we found out that LSA performed significantly better than PLSA as feature extraction techinque. This might have been because of the lack of resource that we had to face during this project. We also found that preprocessing techinques that are available to us today has a lot of room for improvement. Tools like stemmer and lemmatizer could persuade the conclusions of projects such as ours to a different extent. Despite the limited amount of data and tools available for training the model.

The ensemble techinque seems to most benefit of this feature extraction techinque. Instead of manually extracting the features from the huge amount of unstructured documents, we can use PLSA and LSA to increase efficieny in classification. These kinds of algorithms can be utilized for Nepali texts and make better performing machine learning models.

6.2   Future Recommendations

Despite being a relatively simple algorithms, a lot of improvements can be made to further increase the performance of the feature extraction. Things like multi processing plsa can be made to make the algorithm run faster and more memory efficient. We can also use the ratio of the sum of tf-idf of a given word to curate the data and filter out the unneeded features that are less significant in the learning process of the model.

Also, using neural networks could make the process of feature extraction more easier and automatic. Although they also suffer from the need of large amount of resources, they are significantly more powerful than simple machine learning algorithms.

Bibliography

[1] "Document classification." `https://en.wikipedia.org/wiki/Document_classification`, Jun 2021.

[2] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[3] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[4] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press Cambridge, 2008.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[6] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[7] U. Cambridge, *Online edition (c) 2009 Cambridge UP An Introduction to Information Retrieval Christopher D*, ch. Matrix Decomposition and Latent Semantic Indexing, pp. 403–419. Manning Prabhakar Raghavan Hinrich Schütze Cambridge University Press, 2009.

[8] T. Hofmann, "Probabilistic latent semantic analysis," *arXiv preprint arXiv:1301.6705*, 2013.

[9] C. C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[10] Q. Pu and G.-W. Yang, "Short-text classification based on ica and lsa," in *International Symposium on Neural Networks*, pp. 265–270, Springer, 2006.

[11] A. Cardoso-Cachopo and A. L. Oliveira, "An empirical comparison of text categorization methods," in *International Symposium on String Processing and Information Retrieval*, pp. 183–196, Springer, 2003.

[12] K. Krishnamurthi, V. R. Panuganti, and V. V. Bulusu, "Including category information as supplements in latent semantic analysis of hindi documents," *International Journal of Computational Science and Engineering*, vol. 15, no. 1-2, pp. 138–145, 2017.

[13] K. Krishnamurthi, V. R. Panuganti, and V. V. Bulusu, "Understanding document semantics from summaries: a case study on hindi texts," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 16, no. 1, pp. 1–20, 2016.

[14] A. S. Nipu and U. Pal, "A machine learning approach on latent semantic analysis for ambiguity checking on bengali literature," in *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pp. 1–4, IEEE, 2017.

[15] C. Sitaula, A. Basnet, and S. Aryal, "Vector representation based on a supervised codebook for nepali documents classification," *PeerJ Computer Science*, vol. 7, p. e412, 2021.

[16] T. B. Shahi and A. K. Pant, "Nepali news classification using naïve bayes, support vector machines and neural networks," in *2018 International Conference on Communication Information and Computing Technology (ICCICT)*, pp. 1–5, IEEE, 2018.

[17] K. Kafle, D. Sharma, A. Subedi, and A. K. Timalsina, "Improving nepali document classification by neural network," in *Proceedings of IOE Graduate Conference*, pp. 317–322, 2016.

[18] S. Subba, N. Paudel, and T. B. Shahi, "Nepali text document classification using deep neural network," *Tribhuvan University Journal*, vol. 33, no. 1, pp. 11–22, 2019.

[19] O. Singh, "Nepali multi-class text classification," 2018.

[20] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. mcgraw-hill, 1983.

Chapter 7

Appendices

```python
from string import punctuation
You, now | 4 authors (Dpes Neupane and others)
class Preprocessing():          Dpes Neupane, 5 months ago via PR #22 • preprocessing files were made …
    '''...
    def __init__(self, save_in="", output=False):
        '''...
        self.save_in = save_in
        self.output = output

    def delEngWords(self, filename="", text="", next=False):
        '''...
        processed_text = ""
        # print(filename)
        if not text :
            with open(filename, encoding="utf-8") as fp:
                raw_text = fp.read()
        else: raw_text = text
        for letters in raw_text:
            if not ((letters <= 'z' and letters >= 'a') or ( letters >= 'A' and letters <= 'Z')):
                processed_text += letters
        if self.save_in:
            with open(self.save_in, 'w', encoding='utf-8') as wp:

                wp.write(processed_text)
        if self.output: print(processed_text)
        if next:
            return processed_text

    def delNumbers(self, filename="", text="", next = False):
        nepali_no = ['०', '१', '२', '३', '४', '५', '६', '७', '८', '९']
        processed_text = ""
        if not text :
            with open(filename, encoding="utf-8") as fp:
                raw_text = fp.read()
        else: raw_text = text
        for letters in raw_text:
            if letters not in nepali_no and not (letters >= '0' and letters <= '9'):
                processed_text += letters
        if self.save_in:

            with open(self.save_in, 'w', encoding= "utf-8") as wp:
                wp.write(processed_text)

        if self.output:
            print(processed_text)
        if next:
            return processed_text
```

Figure 7.1: **Preprocessing Steps**

```python
def tf_idf(self,vec=None):
    '''
        term-frequency (tf) =   (Number of repeatition of words in document)
                                ------------------------------------------
                                        (Number of words in a document)

        inverse-document-frequency (idf) =             (Number of document)
                                            ------------------------------------------
                                            (Number of document containing words)

        tf-idf = tf * idf
    '''
    if vec is not None:
        vec = vec/vec.sum(axis=1)[:,np.newaxis]
        idf = np.log(vec.shape[0]/np.count_nonzero(vec,axis=0))
        idf = idf.astype(np.float32, copy=False)
        vec = vec*idf
        return vec
    else:
        self.vec= (self.vec/self.vec.sum(axis=1)[:,np.newaxis])

        idf=np.log(self.vec.shape[0]/np.count_nonzero(self.vec,axis=0))

        idf=idf.astype(np.float32, copy=False)

        self.vec=self.vec*idf
```

Figure 7.2: **TF-IDF**

28

```python
def logistic_regression(X_train, X_test, y_train, y_test):
    from sklearn.linear_model import LogisticRegression
    logreg = LogisticRegression()
    logreg.fit(X_train, y_train)
    y_pred = logreg.predict(X_test)
    from sklearn.metrics import classification_report
    return classification_report(y_test, y_pred,zero_division=1)

def random_forest(X_train, X_test, y_train, y_test):
    from sklearn.ensemble import RandomForestClassifier
    rf = RandomForestClassifier(n_estimators=100)
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    from sklearn.metrics import classification_report
    return classification_report(y_test, y_pred,zero_division=1)

def svm(X_train, X_test, y_train, y_test):
    from sklearn.svm import SVC
    svc = SVC(kernel='linear')
    svc.fit(X_train, y_train)
    y_pred = svc.predict(X_test)
    from sklearn.metrics import classification_report
    return classification_report(y_test, y_pred,zero_division=1)
```

Figure 7.3: **Model**

```python
def expectation_step(self) -> None:
    '''the expectation step calculates teh posterior probability P(z | d, w)'''
    self.topic_prob_G_doc_w = np.nan_to_num(self.topic_prob_G_doc_w)

    for doc in range(self.n_doc):
        for word in range(self.w_words):
            self.topic_prob_G_doc_w[doc, :, word] = self.prob_document_G_topic[doc, :] * self.prob_words_G_topic[:, word]
            self.topic_prob_G_doc_w[doc, :, word] /= self.topic_prob_G_doc_w[doc, :, word].sum()

    self.topic_prob_G_doc_w = np.nan_to_num(self.topic_prob_G_doc_w)


def maximization_step(self) -> None:
    '''The M-step updates  P(w|z) and P(z|d)'''
    print("M-step....")
    print("updating P(z|d)")

    for doc in range(self.n_doc):
        for topic in range(self.z_topics):
            self.prob_document_G_topic[doc, topic] = self.TDmat[doc, :]  @ self.topic_prob_G_doc_w[doc, topic, :]
        self.prob_document_G_topic[doc, :] /= self.prob_document_G_topic[doc, :].sum()
    self.prob_document_G_topic = np.nan_to_num(self.prob_document_G_topic)

    print("\nupdating P(w|z)")

    for topic in range(self.z_topics):
        for word in range(self.w_words):            Dpes Neupane, 4 months ago via PR #29 • plsa algorithm coded …
            self.prob_words_G_topic[topic, word] = self.TDmat[:, word] @ self.topic_prob_G_doc_w[:, topic, :]
        self.prob_words_G_topic[:, word] /= self.prob_words_G_topic[:, word].sum()
    self.prob_words_G_topic = np.nan_to_num(self.prob_words_G_topic)


def log_likelihood(self) -> float:
    '''Calculate the current log-likelihood using the updated probability matrices'''

    return np.sum(np.log(self.prob_document_G_topic @ self.prob_words_G_topic) * self.TDmat)

def plsa(self, max_iter:int, epsilon:float):
    '''max_iter = the number of iterations the log_likelihood is calculated
    epsilon = the absolute difference between the current likelihood and last likelihood that is tolerated'''
    self.initialize()

    current_likelihood = 0.0
```

Figure 7.4: **Code of PLSA**

```python
'''
class that performs Latent Semantic Analysis on the Document-Term corpus.
'''

class LSA:
    def __init__(self):
        pass

    def get_svd(self, tf_idf, n=100):
        from numpy.linalg import svd as SVD
        import numpy as np
        u, s, vh = SVD(tf_idf, full_matrices=False)
        u_reduced = u[:, :n]
        s_reduced = s[:n]
        vh_reduced = vh[:n, :]

        usvh_temp = np.dot(u_reduced, np.diag(s_reduced))
        usvh = np.dot(usvh_temp, vh_reduced)
        return usvh
```

Figure 7.5: **Code of LSA**