Chapter 1

INTRODUCTION

1.1 Introduction

1.1.1 Document Classification

Modern information age produces vast amount of textual data, which can be termed in other words as unstructured data. Internet and corporate spread across the globe produces textual data in exponential growth, which needs to be shared, on need basis by individuals. If the data generated is properly organized, classified then retrieving the needed data can be made easily with least efforts. Hence the need of automatic methods to organize, classify the documents becomes inevitable due to such exponential growth in documents, very especially after the increase usage of internet by individuals. Automatic classification refers to assigning the documents to a set of pre-defined classes based on the textual content of the document or feature vector hat represent the document in which it belongs to. There are many research work had done for the text classification based on machine learning approaches and rule-based approaches. Paper [25] introduces many learning approaches like Naive Bayesian, Riccho Method, Neural Network, Decision tree, KNN method and so on for text classification. But we analyze the concept of new classification model which classifies Self-created database of Nepali collected documents to predefined classes such as education, business, sport, health and crime etc. The task of Nepali document classification is unique challenges in term of time and efficiency. The classification can be flat or hierarchical. As show in the figure 1.1 flat classifications, categories are considered parallel, i.e., one category does not supersede another. If the class categories grow significantly large in number say, in thousands then searching with such a large number of categories becomes very difficult. This difficulty leads to have hierarchical classification in which the thematic relationship between the classifications is also used, in searching of documents. Figure 1.2 gives hierarchical classification.

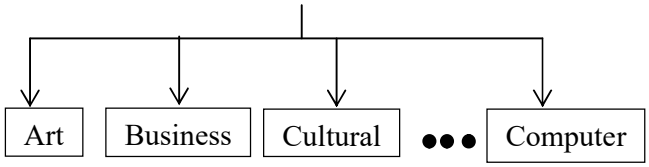


Figure 1.1: Flat classification

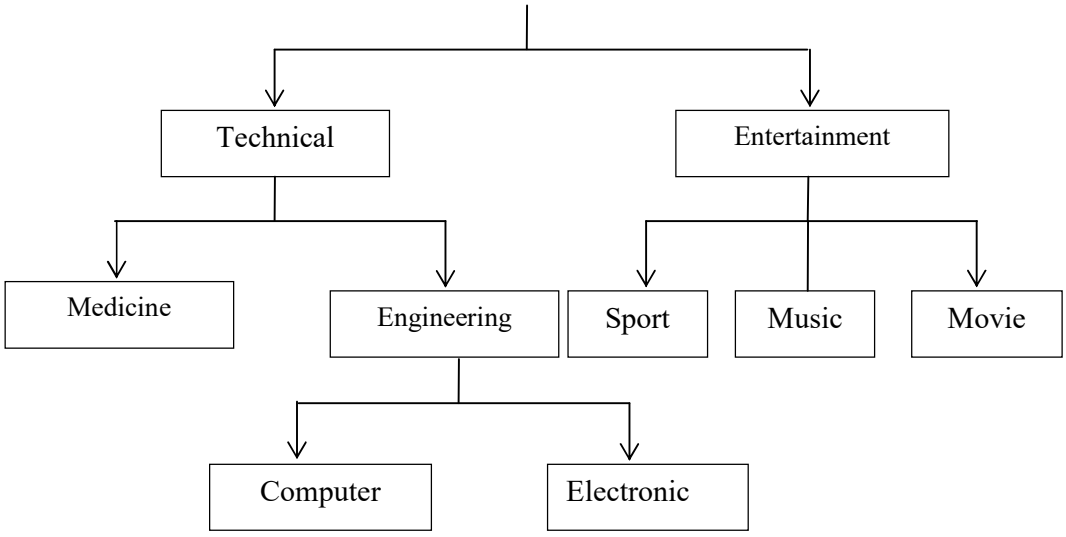


Figure 1.2: Hierarchical classification

1.1.2 Text Classification

Text classification [30], also known as text categorization or topic spotting, is a supervised learning task, where pre-defined category labels are assigned to documents based on the likelihood suggested by a training set of labeled documents. In knowledge engineering, expert knowledge is used to define manually a set of rules on how to classify documents under the pre-defined categories. It is discussed in [25] that the machine learning approach to document classification leads to time and cost savings in terms of expert manpower without loss in accuracy. In the problem of text classification we have a set D of documents and a set *S* of pre-defined categories. The aim is to assign a Boolean value to each (d_i, c_j) pair, where $d_i \in D$ and $c_j \in S$. A value of *true* assigned to (d_i, c_j) stands for the decision of assigning document d_i to category c_j. Analogously, value of false assigned to (d_i, c_j) stands for the decision of not assigning document d_i to category c_j. To state more formally, the task is to

approximate the unknown target function f: D×S→ {true, false}, that describes the way the documents should actually be classified, by the classifier function f': $D \times S \rightarrow \{true, false\}$ such that number of decisions of f and f' that do not coincide is minimized.

Initially machine learning was applied for a binary classification, where the classifier determines whether the document belongs to some pre-defined category or not. A special case of single-label classification is binary classification, in which, given a category ck, each dj ∈ D must be assigned either to ck or to its complement ck. An example of binary classification is distinguishing an e-mail message between spams and legitimate [27].
Figure 1.3 gives binary classification. In hard classifier a document at any instance of time belongs to a single category, it cannot be a member of two classes.

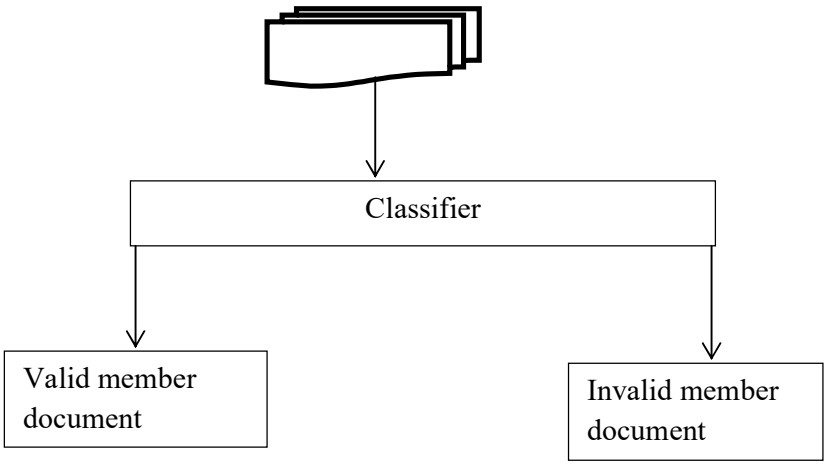


Figure 1.3: Binary classification

If the classification is made in such a way that one document belongs to utmost only one class at a time, then we refer to such a classification as single-label multi-class classification task. Figure1.4 shows abstract multi-class single label classifier.

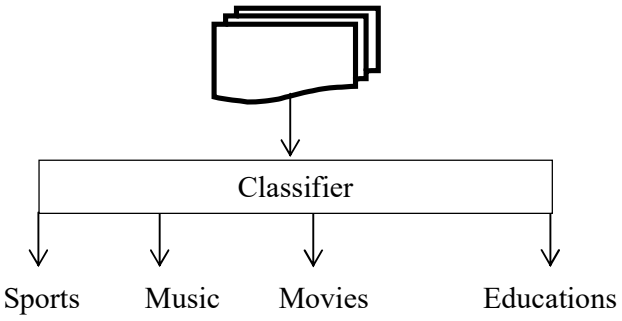


Figure 1.4: Multiple class single level classification

1.1.3 Principles of Text Classification

The two main principles that are widely accepted are as follows:

- **Content based classification**: In this classification, the weight given to particular subjects in a document determines the class to which the document is belongs to. In automatic classification, it could be the number of times given words appears in a document.
- **Request oriented classification**: In this classification, the anticipated request from users is influencing how documents are being classified.

1.1.4 Methods of Text classification

Text classification includes two main methods: topic based text classification and text genre- based classification.

- **Topic-based text categorization**: classifies documents according to their topics.
- **Texts genre- based classification**: classifies document based on the genre which can be defined as the way a text is created, the way it is edited, the register of language it uses, and the kind of audience to whom it is addressed.

As considering the human interference, text classification can be categorized into manual or automated.

1.1.4.1 Manual text classification

Text classification can be done manually which is very accurate when job is done by experts and is consistent when the problem size and team is small. Since, it is difficult and expensive to scale. It has several disadvantages

- It needs domain experts in the areas of predefined categories.
- It is time-consuming, leads to frustration.
- It is error-prone and could be employee biased (subject biased).
- Human decision among two experts may disagree.
- Need to repeat the process for new documents (possibly of another domain).

1.1.4.2 Automatic text classification

Manual text classification has several disadvantages so need to employee machine learning to automate the classification is needed. In machine learning generally two types of learning algorithms are found in the literature. Supervised learning algorithms and unsupervised learning algorithms. We have used supervised earning algorithms in this research. In the

process of TC construction of a generalize model(classifier) is considered, which is constructed using pre-classified training documents for each category so as to classify correctly the unseen documents

Automatic Text Classification [30] which automatically involves assigning a text document to a set of pre-defined classes, using a machine learning technique. The classification is usually done on the basis of significant words or features vectors extracted from the text document.

1.1.5 Nepali Text Classification

Nepali Text classification is the act of dividing a set of input Nepali documents into two or more classes where each document can be said to belong to one or multiple classes. We classified the collected Nepali textual document into five different categories namely Business, Crime, Sport, Health and Education. It is one of the challenging problems in the field of artificial intelligence and machine learning. Today huge amount of information are being associated with the web technology and the internet which are unstructured, unordered and congested. To gather useful information from it these texts have to be categorized [25]. Since, huge growth of information flows and especially the explosive growth of Internet promoted the growth of automated text classification. The development of computer hardware provided enough computing power to allow automated text classification to be used in practical applications. Text classification is commonly used to handle spam emails, classify large text collections into topical categories and manage knowledge and also to help Internet search engines.

1.2 Applications of Document Classification

The motivation behind developing text classification systems is inspired by its wide range of applications.

1. **Spam Filtering**: A spam filter is a program that is used to detect unsolicited and unwanted email and prevent those messages from getting to a user's inbox. A text classification system could, in the ideal case, categorize incoming messages into genuine and spam categories, rejecting these that it found to be spam.

2. **Document Organization**: A news or media company will typically get hundreds and thousands of submissions every day. In order to efficiently handle such vast flow of information, there is a need of an automatic text classification system, which would

categorize each document by topics so that they could be sent to the relevant recipient maintaining the Integrity of the Specification.

3. **Web page prediction**: Text classification can be used to predict web page the user is likely to click on. Each hyperlink text description is treated as a miniature document. Also a text categorization system could be used to naively predict the next page for a fast look-ahead caching system.

4. **Pornography classification**: The exponential increase of information in internet has raised the issue of information security. Pornography web content is one of the biggest harmful resources that pollute the mind of children and teenagers. Several web content classification approaches have been proposed to avoiding these illicit web contents which are accessing by the children.

5. **Automatic summary evaluation**: Text classification could be applicable to evaluate automatic summarization of text on the basis of feature vector of document.

1.3 Motivation

Nepali Text Classification is a special problem in the domain of text mining and machine intelligence. The field of Text Classification is split into two different categories: Automatic classification and Manual classification. Due to information overload, efficient classification and retrieval of relevant content has gained significant importance. The problem of classification increases when we operate it in the automatic mode. There lots of work has been done in this area in the past few years. There are lots of research work have been done for English as well as other language too. But there is no any work done for the Nepali language, since I was motivated to do this research work for Nepali language. Nepali Text Classification system can also help in automatic organizing of web content, filtering, prediction of web pages and any other. Automatic processing benefits into availability for their contents. Although, a lot of approaches have been proposed for other languages, so automated text categorization is still a major area of research.

1.4 Problem Definition

In this research work, the problem of Nepali news type classification is addressed. The problem of Nepali text classification is determined the class of input document according to

its content. The classification task will be carried out with Naive Bayesian (NB), and SVM. SVM and Naive Bayesian (NB) are techniques to classifying news type. In this thesis work, to present models based on SVM and Naive Bayesian (NB) methods will be presented for news type filtering the text written in which is used for Nepali language. The accuracy of classifier is one of the main concerns of the thesis. In order to classify documents, data sets are prepared by collecting documents from newspaper, article etc. The problem is a kind of text classification, since news type is in a language is a special case of texts in the language.

There are many sub-problems in the domain of document classification such as stop-words removal, symbols and punctuation removal, white spaces removal, word stemming, feature extraction, term weighting etc. These are also addressed with the most suitable solutions in the literature for this research work.

1.5 Objective

The objective of this research work is to investigate various feature extraction techniques and to compare Support vector machine and Naive Bayesian probabilistic techniques, to improve accuracy of Nepali text classification. Comparative Performance matrices are analyzed.
The sub-problem is also addressed such as stop-words removal, symbol and punctuation mark removal, digit and Non-Nepali character removal, stemming, feature extraction, term weighting etc.
The main objective is given below:

1. To build Nepali news type filter based on two methods: Support vector machine and Naive Bayes
2. To analyze the accuracy of Support vector machine (SVM) and Naïve (NB) Bayes Classifiers for news type classification on Nepali text.

1.6 Contribution of this Dissertation

The main contribution of this thesis to the field of automatic Nepali news type classification can be seen in its extensive experimental work. A more detailed list of the various contributions is provided below:

- Use of Naive Bayes Classifier and Support vector machine to analysis the Nepali news type classification.
- Built a news type text classification model for Nepali language text

1.7 Outline of the Document

The remaining part of the document is organized as follows

Chapter 2 describes necessary background information and related work of document classification research on single document as well as in multi document.

Chapter 3 describes in detail the system model and the theoretical approaches for automated Nepali document classification problem. It includes document preprocessing, feature extraction and classification methods.

Chapter 4 describes the implementation details of the system. All the methods described in the **Chapter 3** are implemented for system evaluation.

Chapter 5 includes experimentation results and analysis of the systems.

Chapter 6 concludes the system performance and future directions.

Chapter 2

BACKGROUND AND LITERTURE REVIEW

2.1 Literature review

Text classification, which dates back to the beginning of the 1960 but only in 1990 did it became major principle in the information systems discipline. Text categorization is now being applied in many context based on a vocabulary to document indexing based on controlled vocabulary, to document filtering , word sense disambiguation, etc. “Knowledge engineering” [25] which is more popular approach used in late 1980 which consisting of set of rules encoding expert knowledge on how to classify document under given categories. Furthermore, machine earning approach was introduced due to the increasing popularity of classification introduces an automatic text classifier by learning. If we survey previous works relevant to this research, There exist other kinds of approaches to text categorization than machine learning based ones: heuristic and rule based approaches. Heuristic approaches were already applied to early commercial text categorization systems. However, we count out the kind of approaches in our exploration, since they are rule of thumbs. Since rule based approaches have poor recall and require a time consuming job of building rules manually as mentioned in the previous section, they are not covered in this thesis, either. Therefore, this research counts only machine learning based approaches to text categorization considered as state of the art ones.

Typical machine learning algorithms applied traditionally to text categorization are KNN (K Nearest Neighbor), NB (Naïve Bayes), SVM (Support Vector Machine), and BP (Back Propagation). The four approaches to text categorization have been used more popularly in previous literatures on text categorization than any other traditional approaches.

Among them, the simplest approach is KNN. KNN is a classification algorithm where objects are classified by voting several labeled training examples with their smallest distance from each object. KNN was initially applied to classification of news articles by Massand et al, in 1992 [19]. Yang compared 12 approaches to text categorization with each other, and judged that KNN is one of recommendable approaches, in 1999 [33]. KNN is evaluated as a simple and competitive algorithm with Support Vector Machine for implementing text

categorization systems by Sebastiani in 2002 [25]. Its disadvantage is that KNN costs very much time for classifying objects, given a large number of training examples because it should select some of them by computing the distance of each test object with all of the training examples.

Another popular and traditional approach to text categorization is NB Differently from KNN; it learns training examples in advance before given unseen examples. It classifies documents based on prior probabilities of categories and probabilities that attribute values belong to categories. The assumption that attributes are independent of each other underlies on this approach. Although this assumption violates the fact that attributes are dependent on each other, its performance is feasible .In text categorization [17]. Naïve Bayes is used popularly not only for text categorization, but also for any other classification problems, since its learning is fast and simple [5]. In 1997, Mitchell presented a case of applying NB to text categorization in his textbook [17]. He asserted that NB was a feasible approach to text categorization, although attributes of numerical vectors representing documents were dependent on each other; this fact contradicts with the assumption underlying in NB. In 1999, Mladenic and Grobellink evaluated feature selection methods within the application of Naïve Bayes to text categorization [18]. Their work implied that NB is one of standard and popular approaches to text categorization. Androutsopoulos et al adopted NB for implementing a spam mail filtering system as a real system based on text categorization in 2000 [1]. It requires encoding documents into numerical vectors for using NB to text categorization.

Another popular and traditional approach to text categorization is SVM. Recently, this machine learning algorithm becomes more popular than the two previous machine learning algorithms. Its idea is derived from a linear classifier, Perceptron, which is an early neural network. Since the neural network classifies objects by defining a hyper-plane as a boundary of classes, it is applicable to only linearly separable distribution of training examples. The idea of SVM is that if a distribution of training examples is not linearly separable; these examples are mapped into another space where their distribution is linearly separable. SVM optimizes the weights of the inner products of training examples and its input vector, called Lagrange multipliers [3], instead of those of its input vector, itself, as its learning process. It defines two hyper-planes as a boundary of two classes with a maximal margin. Refer to [12] or [3], for more detail description on SVM. The advantage of SVM is that it is tolerant to huge dimensionality of numerical vectors; it addresses the first problem. Its advantage leads

to make it very popular not only in text categorization, but also any other classification problems [3]. In 1998, it was initially applied to text categorization by Joachims [14]. He validated the classification performance of SVM in text categorization by comparing it with KNN and NB. Drucker et al adopted SVM for implementing a spam mail filtering system and compared it with NB in implementing the system in 1999 [4]. They asserted empirically that SVM was the better approach to spam mail filtering than NB. In 2000, Cristianini and Shawe-Taylor presented a case of applying SVM to text categorization in their textbook [3]. In 2002, Sebastiani asserted in his survey paper that SVM is most recommendable approach to text categorization by collecting experimental results on the comparison of SVM with other approaches from previous works [25]. In spite of the advantage of SVM, it has two demerits. One is that it is applicable to only binary classification; if a multiple classification problem is given, it should be decomposed into several binary classification problems for using SVM. The other is that it is fragile to the problem in representing documents into numerical vectors, sparse distribution, since the inner products of its input vector and training examples generates zero values very frequently.

The third popular and traditional approach to text categorization is BP. It is most popular supervised neural network and used for not only classification tasks but also nonlinear regression tasks [10][11]. It is also derived Perceptron, together with SVM. When a distribution of training examples is not linearly separable, in SVM, the given space is changed into another space where the distribution is linearly separable, whereas in back propagation, a quadratic boundary is defined by adding one more layer, called hidden layer [10][11]. More detail explanation about back propagation is included in [10] or [11]. In 1995, BP was initially applied to text categorization by Wiener in his master thesis [31]. He used Reuter 21578 as the test bed for evaluating the approach to text categorization and shown that back propagation is better than KNN in the context of classification performance. In 2002, Ruiz and Srinivasan applied continually back propagation to text categorization [23].They used a hierarchical combination of BPs, called HME (Hierarchical Mixture of Experts), to text categorization, instead of a single BP. They compared HME of BPs with a flat combination of BPs, and observed that HME is the better combination of BPs. Since BP learns training examples very slowly, it is not practical, in spite of its broad applicability and high accuracy, for implementing a text categorization system where training time is critical. Research on machine learning based approaches to text categorization has been progressed very much, and they have been surveyed and evaluated systematically. In 1999, Yang

evaluated 12 approaches to text categorization including machine learning based approaches directly or indirectly in text categorization [33]. She judged the three approaches, LLSF (Linear Least Square Fit), K Nearest Neighbor, and Perceptron, worked best for text categorization. In 2002, Sebastiani surveyed and evaluated more than ten machine learning based approaches to text categorization [25]. He asserted that Support Vector Machine is best approach to text categorization with respect to classification performance. All approaches which were surveyed and evaluated in these literatures require encoding documents into numerical vectors in spite of the two problems. We explored and presented previous cases of applying one of the four traditional machine learning algorithms to text categorization. Although the traditional approaches are feasible to text categorization, they accompany with the two main problems from representing documents into numerical vectors. In the previous works, dimension of numerical vectors should reserve, at least, several hundred for the robustness of text categorization systems. In order to mitigate the second problem, sparse distribution, a task of text categorization was decomposed into binary classification tasks in applying one of the traditional approaches. This requires classifiers as many as predefined categories, and each classifier judges whether an unseen document belongs to its corresponding category or not. There is a previous trial to solve the two problems. In 2002, Lodhi et al proposed a string kernel for applying Support Vector Machine to text categorization [16]. In their solution, documents as raw data are used directly for text categorization without representing them into numerical vectors. String kernel is a function computing an inner product between two documents given as two long strings. An additional advantage of the solution is to process documents independently of a natural language in which documents are written. However, their solution was not successful in that it took far more time for computing string kernel of two documents and the version of SVM using the string kernel was not better than the traditional version. There are many research works for text classification found based on rule and machine learning approaches. But there is no comparison of Naïve Bayes and support vector machine classification techniques are available in the literature of Nepali text is made. Since, classification of Nepali text is challenging problem. There is a large corpus of research on the application of text classification in different domains, but no system to date has achieved the goal of system acceptability for Nepali text classification. This research will be a successful attempt to solve Nepali news type classification problem by proposing Naïve Bayesand support vector machine classifier.

2.2 Machine Learning

Machine learning is programming computers to optimize a performance criterion using example data or past experience. Learning is done by the execution of a computer program to optimize the parameters of a defined model, which can be predictive to make predictions in future or descriptive to gain knowledge from data or both. There are two things that need to be achieved in the learning process. First, the training needs to be done with an efficient algorithm to solve the optimized problem, and to store and process the data. Second, the trained model needs an efficient representation and algorithm for inference.

2.2.1 Types of Machine Learning Algorithms

Machine learning algorithms are organized in a taxonomy based on their desired outcomes. The common algorithm types are:

- Supervised learning – These algorithms generate a function based on the training data set that maps inputs to predetermined labels or classes.

- Unsupervised learning – These algorithms are not provided with classification. They seek out similarities between data points to form clusters.

- Semi-supervised learning – The algorithms combine both labeled and unlabeled data to generate a classifier.

- Reinforcement learning – These algorithms learn by interacting with an environment. Instead of being trained on an existing data, these algorithms learn from their actions which are selected based on past actions (exploitation) or by new choices (exploration). The algorithm receives a numeric reward from the environment for a successful outcome of an action and it selects actions that maximize the accumulated reward over time [8].

- Transduction – These algorithms are similar to supervised algorithms but instead of explicitly specifying a function, they try to predict new outputs based on training inputs, training outputs, and testing inputs.

- Learning to learn – These algorithms learn their own inductive bias based on previous experience.

2.3 Overview of Data Mining Concept

Data Mining is the process of extracting knowledge or discovering of new information from large volumes of raw data [15]. The knowledge or information should be new and one must be able to use it. It discovers patterns and relationship using data analysis tools and techniques to build models.

There are two main kinds of models in data mining which are as follow:

- Predictive model: In this model, known data results are used to develop a model and that can be used to explicitly predict values.
- Descriptive model: In this model, patterns are described from existing data and models are abstract representation of reality which can be reflected to understand business and suggest actions.

2.4 Data Mining

Data mining was introduced in the 1990s and it is traced back along three categories i.e. classical statistics, artificial intelligence and machine learning. Data mining is the process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. It is also known as knowledge discoveries detecting something new from large–scale or information processing [15]. Its objective is to extract information from a data set and transforms it into an understandable structure for further use. It is mainly related to database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post processing of discovered structures, visualization, and online updating. The data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system.

A Decision Support System [15] is a computer-based information system that supports business or organizational decision making activities. It serves the management, operations, and planning levels of an organization and help to make management decisions, which may be rapidly changing and not easily specified in advance .Hence, the actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as groups of data records (cluster analysis i.e. grouping a set of objects in such a way that objects in the same group are more similar to each other than

to those in other groups), unusual records (anomaly detection i.e. detection of outliers, noise, deviations or exceptions in large data sets) and dependencies (association rule mining i.e. detecting interesting relations between the variables in large databases).

2.4.1 Purpose of Data mining

Data mining can automate the process of finding relationships and patterns in raw data. Thus, results can be either utilized in an automated decision support system or assessed by a human analyst. There are three main reasons to use data mining: especially in science and business areas which need to analyze large amounts of data to discover trends which they could not otherwise find.

- Too much data and too little information.
- There is a need to extract useful information from the data and to interpret the data.
- Predict outcomes of future situations.

2.4.2 Technique in Data mining

Data mining is the automated extraction of patterns representing knowledge implicitly stored in large databases, data warehouses and other massive information repositories. Some of the techniques adopted in data mining as given below:

2.4.2.1 Association rule

In this technique, interesting association between attributes that are contained in a database are discovered which are based on the frequency counts of the number of items occur in the event (i.e. a combination of items), association rule tells if item X is a part of the event, then what is the percentage of item Y is also the part of event.

2.4.2.2 Clustering

Clustering is a technique used to discover appropriate groupings of the elements for a set of data. It is undirected knowledge discovery or unsupervised learning i.e, there is no target field and relationship among the data is identified by bottom-up approach.

2.4.2.3 Decision Tree

In this technique, classification is performed by constructing a tree based training instance with leaves having class labels. The tree is traversed for each test instance to find a leaf, and the class of the leaf is predicted class. This is a directed knowledge discovery in the sense that there is a specific field whose value we want to predict.

2.4.2.4 Neural Network

It is often represented as a layered set of interconnected processors. These processor nodes are frequently referred as neurons so as to indicate a relationship with the neurons of the brain. Each node has a weight connection to several other nodes in adjacent layers, each individual nodes take the received from connected nodes and use the weights together to compute output values.

2.4.2.5 Classification and prediction

Classification is the technique in which set of documents are classified in the predefined category. Prediction is the process of predicting categorical class labels, constructing a model based on the training set and class labels in a classifying attribute.

2.5 Automatic Text Classification

Automatic text categorization has an important application and research topic since the inception of digital documents. The TC task assigns category label to new documents based on the knowledge gained in a classification system. A wide variety of supervised machine learning algorithms has been applied to this area using a training data set of categorized documents. TC can play an important role in wide range of more flexible, dynamic and personalized task. In general, it can be applied in many applications requiring document organization or selective and adaptive document dispatching. Automatic document classification tasks can be divided into three sorts:

2.5.1 Supervised Text Classification

In Supervised Learning, incorporates an external teacher, the set of possible classes is known in advance so that each output unit is told what its desired response to input signals ought to be. It may require global information during the learning process. Supervised learning includes error-correction learning, reinforcement learning and stochastic learning. An important concerning issue of supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. A paradigm of supervised learning is least mean square convergence which is known method among many paradigms.

2.5.2 Unsupervised Text Classification

In unsupervised classification, the set of possible classes is not known. After classification, we can try to assign a name to that class. It is called clustering, where the classification is done entirely without reference to external information. It uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that data are organized by itself presented to the network and detects their emergent collective properties.

2.5.3 Semi-supervised text Classification

In this classification, parts of the documents are labeled by the external mechanism. It learns with a small set of labeled examples and a large set of unlabeled examples i.e. learning with positive and unlabeled examples.

2.6 Automatic Text Classification Techniques

Dealing with unstructured text, handling large number of attributes, examining success of preprocessing techniques , dealing with missing meta data and choice of a suitable machine learning technique for training a text classifier are major concerns of automatic text classification. Since, no single method is found to be superior to all others for all types of classification. Some of widely accepted techniques due to extensive increase in digital documents are as follows:

2.6.1Support Vector Machines

Support Vector Machines (SVMs) are a generally applicable tool for machine learning. Let training examples are xi, and the target values y_i€ {-1, 1}. SVM searches for a separating hyper plane, which separates positive and negative data samples from each other with maximal margin, in other words, the distance of the decision surface and the closest example is maximal[12].

2.6.2Artificial neural networks

A neural network is a powerful data-modeling tool that is able to capture and represent complex input/output relationships [11]. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform ”intelligent” tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways:

- a neural network acquires knowledge through learning.

- a neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

Physical nervous system is highly parallel, distributed information processing system having high degree of connectivity with capability of self-learning. Human nervous system contains about 10 billion neurons with 60 trillions of interconnections. These connections are modified based on experience.

Artificial neural network is non-linear, parallel, distributed, highly connected network having capability of adaptively, self-organization, fault tolerance etc. which closely resembles with physical nervous system. Artificial neural networks are composed of interconnecting artificial neurons that mimic the properties of biological neurons which can be either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system. The real biological nervous system is highly complex. Artificial neural network algorithms attempt to abstract this complexity and focus on what may hypothetically matter most from an information processing point of view. Another incentive view is to reduce the amount of computation required to simulate artificial neural networks, so as to allow one to experiment with larger networks and train them on larger data sets.

2.6.3 K-Nearest Neighbor

K-NN is a simplest type of machine learning algorithms, proposed by Galavotti in 2000. In the learning, function is approximated locally and all computation is deferred until classification. It is well known as lazy learning algorithm or instance based learning. It is non-parametric method used for classification and regression in which input consists of k closest training set in the feature space. The output depends on whether K-NN is used for classification or regression. In K-NN classification: the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (where k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the

classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point. In this learning, Euclidean distance is commonly used for continuous variables and Hamming distance is used for discrete variables for text classification problems. A common weighting scheme consists in giving each neighbor a weight of d, where d is the distance to the neighbor.

2.6.4 Decision Tree

This learning algorithm constructs the decision tree with a divide and conquers strategy. Each node in a tree is associated with a set of cases. At the beginning, only the root is present, with associated the whole training set and with all case weights equal to 1. At each node the following, divide and conquer algorithm is executed; trying to exploit the locally best choice, with no backtracking allowed.

Let T be the set of cases associated at the node. The weighted frequency freq (C_i,T) is computed of cases in T whose class in C_i, i €[1,N]. If all cases in T belong to a same class C_j (or the number of cases in T is less than a certain value) then the node is a leaf, with associated
class C_j.

If T1, T2, . . . Ts are the subsets of T and T contains cases belonging to two or more classes, then the information gain of each attribute is calculated.

$$\mathrm{I} = \mathrm{H(T)} - \sum_{\mathrm{i=1}}^{8} \frac{|\mathrm{T_i}|}{|\mathrm{T}|}$$

Where,

$$H(T) = -\sum_{j=1}^{n} \frac{freq(c_j, T)}{|T|} * \log\left(\frac{\mathrm{freq}(c_j, \mathrm{T})}{|T|}\right)$$

is the entropy function.

2.6.5Naive Bayes Probabilistic Model

The most widely used method for text categorization is Naive Bayes Classifier, each word position in a document is defined as an attribute and the value of that attribute to be the word found in that position. Naive Bayes categorization is given by,

$$V_{NB} = argumentP(V_j)P(a|V_j)$$

To conclude, the Naive Bayes Categorization V_{NB} is the categorization that maximizes the probability the words that were actually found in the training documents. Naive Bayes is very popular among spam filters, because it is very fast and simple for both training and testing. Hence, it is simplicity to learn from new examples and the ability to modify an existing model.

2.7 Document Representation

Each document is typically represented by the feature vector or the bag of words. The most common text representation is bag of word approach (BoW). Here, text is represented as a vector. The BoW vectors are then refined by feature extraction, where vectors are removed from the representation using computationally less discrimination value. The set of feature vectors is of very high dimension in the vector space and each vector represents a unique term.

In order to improve the scalability of the text categorization system, dimensionality reduction techniques should be employed to reduce the dimensionality of the feature vectors before they are fed as input to the text classifier. The following are dimensionality reduction techniques applicable to the text categorization system.

2.7.1 Document Term Matrix

If we have a large collection of documents and hence a large number of document vectors then it is more convenient to organize into a matrix. The row vectors of the matrix correspond to terms (words) and the column vectors correspond to documents. Hence, describes the frequency of terms that occur in a collection of documents known as Document–Term matrix. Document frequency of a feature is the number of documents in which the frequency

occurs and is class independent because of its simple computation and good performance [21].

2.7.2 TF-IDF Method

Tf-idf [32] stands for term frequency-inverse document frequency, often used in information retrieval and text mining. It is used to evaluate how important a word to a document in a collection or corpus. The importance increases proportionally to the number of times as a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

Chapter 3

RESEARCH METHODOLOGY

3.1 Automatic Text Classification System Overview

The Top level document classification system is divided into four sub-systems, data acquisition, preprocessing, feature extraction and classification. Each stages of this theoretical model are briefly described in this section. Detail of each subsystem is given in later sections. The top level of data flow diagram of the proposed system is given in Figure 3.1. Various stages have to be performed to achieve automatic text classification for the Nepali documents. Detailed sub-system flow is given in Figure 3.2.

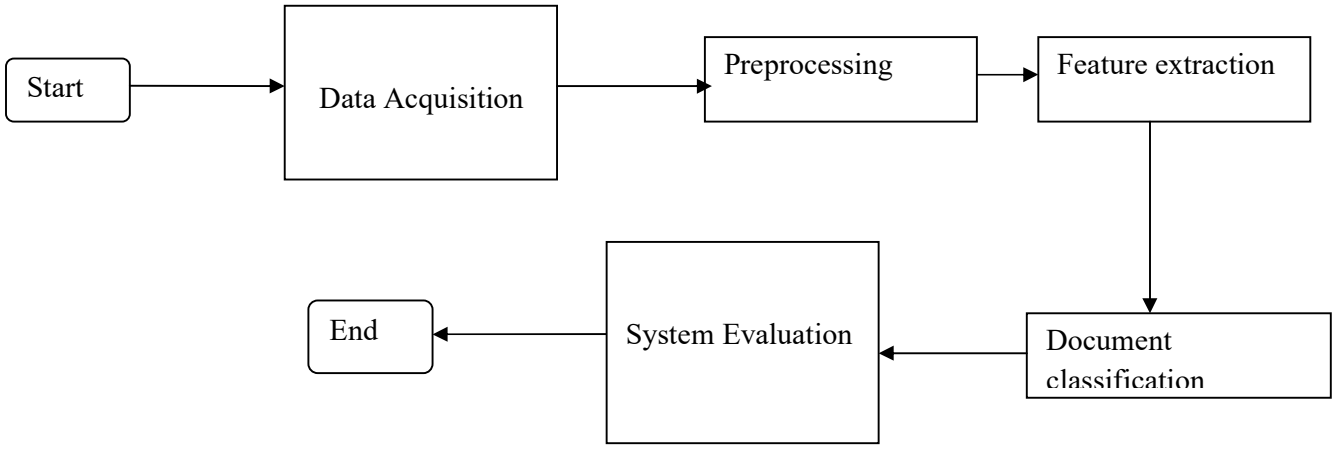


Figure 3.1: Top Level DFD of Automated Text Classification System.

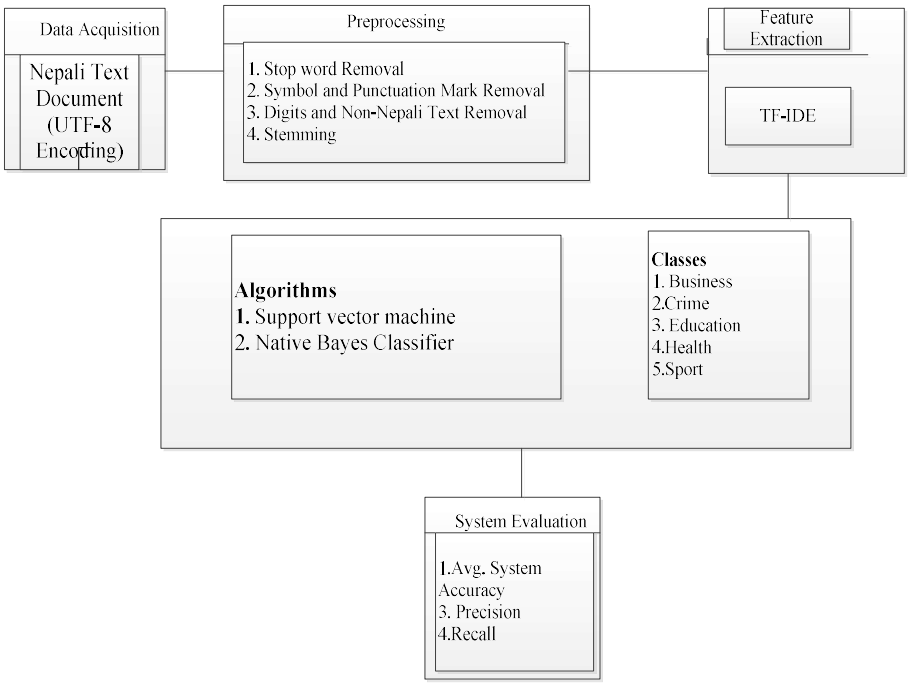


Figure 3.2: The Detail Architecture of Automatic Text Classification System for Nepali Language.

3.2 Data acquisition

Data are acquired from different online and offline Nepali newspaper. Since, data which are collected in huge amount are stored in UTF-8. UTF-8 format is a variable-width encoding that can represent every character in the Unicode character set. The collected documents are transformed into a uniform format which is understandable by machine learning algorithm as input.

3.3 Preprocessing

An approach applied to remove the set of non-content-bearing functional words from the set of words produced by word extraction is known as stop words removal. The next step of text mining process is text preprocessing in which collected documents are analyzed syntactically or semantically. Since, the collected text document is considered as a bag of words because the words and its occurrences are used to represent the document. The algorithm applied in this stage is stemming, digit and non-Nepali text removal, stop word removal, number removal and strip whitespaces. Another task is tokenization which is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such

as punctuation. We have created dictionary for those common words that are useless and has a less discriminative value that do not add meaningful content to the document (auxiliary verbs, conjunctions and articles).

3.3.1 Stop word removal

Stop words are high-frequency words of a language which rarely contribute to useful information in terms of document relevance and appear frequently in the text but provide less meaning in identifying the important content of the document [26, 28]. Those common words that are consider as stop words , useless and has a less discriminative value that do not add meaningful content to the documents are auxiliary verbs , conjunctions and articles. Words are pruned at the processing phase to reduce the number of features vector. The stop word lists for English and other languages are freely available on the Web and often utilized in classification. But, we cannot find easily the stop word list for Nepali language. We have prepared the list of stop words for Nepali language manually for this dissertation. During the removal procedure all the words that appear in a list of stop words are removed by matching from the source documents. Some of the Nepali stop words are given in Figure 3.3. The details of dictionary used for stop word removal is explained in Section 5.2.

Algorithm 3.1 Stop Word Removal

1. Read text document.
2. Match the token of document with token in the stop word dictionary.
3. Remove matched token from document.
4. Repeat until all stop words are not removed.

छ, म, हो, छु, केही, कोही, हामी, मेरो, त्यो, को, हरु, फेरी, हाम्रो, अर्को

Figure 3.3: Sample Stop Words.

3.3.2 Symbols removal

Document may contain some symbols to represent some information. They are not so informative, like the symbols $, #, %, etc. are used to denote some information in the document. So, we need to remove such symbols from the document before feature extraction. As in any language, punctuations are used to organize the text and give the sentence a powerful meaning.

The punctuation in the text summary does not have any value, so we remove all punctuations which are not full stop. The data dictionary used for symbol removal is explained in section 5.2.

Algorithm 3.2 Symbol and Punctuation mark Removal

1 Read text document.
2 Match the token of document with token in the symbol and punctuation mark dictionary.
3 Remove matched token from document.
4 Repeat until all stop words are not removed.

3.3.3 Stemming

In a text document, a word may exist in different morphological variants, stemming reduces such different morphological variant words into the number of unique root words. In text categorization and many other similar tasks, the root word may have different forms. So, it is desirable to combine these morphological variants of the same word into one canonical form. The different morphological variant of the same word which is combined into a single canonical form is called stemming or base word transformation [9].Many NLP applications which use words as basic elements employ stemmers to extract the stems of words. Mainly, it is used in information retrieval systems to improve performance. Actually, this operation reduces the number of terms in the information retrieval system, thus decreasing the size of the index files. Stemming helps to obtain the stem or root of each word, which ultimately helps in semantic analysis and faster processing. There need a specific language dependent stemmer, and is requires some significant linguistic expertise in the language.

A typical simple stemmer algorithm involves removing suffixes/prefixes using a list of frequent suffixes/prefixes, while a more complex one would use morphological knowledge to derive astem from the words. The stemmer which simply prunes the suffixes/prefixes using the list of frequent suffixes/prefixed is very efficient and lightweight approach compared to morphologic parsing. Even though there are some advanced stemmers for languages such as English, the algorithms which they employ do not work well for highly inflected languages such as Nepali. Since, Nepali is a highly inflected language so there are many word forms to denote a single concept. This situation is highly effected for the frequency of a term and therefore words have to be stemmed before getting their frequencies. There is light weight

stemmer and morphological Analyzer that were developed under Madan Puruskar Pustakalya, Nepal [2]. Stemming algorithm for Nepali language is given in Algorithm 3.3.

3.4 Feature Extraction

Feature extraction plays major role in the classification system and it is heart of the classification system. A good feature sets should represent characteristic of a class that helps to distinguish it from other classes, while remaining invariant to characteristic differences within the class. Hence, to improve the accuracy of the classifier, it is necessary to identify a set of "good" features for object representation. To create improved features, measurement of various object properties are carried out to identify good features from a set of raw features [24].

3.4.1Term frequency-inverse document frequency (TF-IDF)

For a machine learning method, the most widely adopted feature weighting scheme in IR, TF- IDF is used, to represent the news as a vector in a vector space model, and it is calculated as follows:

$$a_{ij} = \frac{tf_{ij}.log\frac{|D|}{DF_i}}{\sqrt{\sum_k \left(tf_{kj}.log\frac{|D|}{DF_k}\right)^2}} \qquad (3.1)$$

Where,

tf_{ij} is news in the training set and DF_i is the number of news, containing the term *i*.

The importance of a term in news is measured by the frequency and its inverse document frequency. The more times an item appears in news, the more important it is, and the more times it appears in the training set, the less poorly discriminative it becomes. Often the logarithms of

tf_{ij}or DF_i are taken in order to de-emphasize the increases in weighting for larger values [27]. The TF- IDF weighting of each news text is calculated in the feature extraction procedure of the framework. This is more efficient method to extract the feature, since TF-IDF is constructed based on the word occurs many times in a document. TF-IDF is often used as a weighting factor in text classification.

TF- IDF has many advantages. It is convenient and suitable to use in other languages than English such as Chinese, Japanese, Nepali, because the word segmentation of other language

is more complex than English. This is more efficient method to extract the feature, which TF-IDF. The TF- IDF is constructed based on the word occurs many times in one document. Feature Vector construction is the process of representing the news into a vector form. To represent news in vector form, the TF-IDF weighting value for each word in news is taken as a dimensional value in a vector.

3.5Bayes Theorem

Bayes' Theorem is a statement from probability theory that allows for the calculation of certain conditional probabilities. Conditional probabilities are those probabilities that reflect the influence of one event on the probability of another event. The term generally used in Bayes' theorem is prior probability and posterior probability. The prior probability of a hypothesis or event is the original probability obtained before any additional information is obtained. The posterior probability is the revised probability of the hypothesis using some additional information or evidence obtained.

Bayes' Theorem can be written as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.2}$$

Where,

P (A) is the prior probability of A

P (B) is the prior probability of B

P (A|B) is the posterior probability of A given B

P (B|A) is the posterior probability of B given A

3.5.1 Naive Bayesian Classifier

A naïve Bayes classifier is a probabilistic classifier based on applying Bayes' theorem with strong independence assumptions. Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without believing in Bayesian probability or using any Bayesian methods. The naïve Bayesian classifier was first described in [22] in 1973 and then in [21] in 1992.

An advantage of the Naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification.

Because independent variables are assumed, only the variances of the variables for each class need to be determined not the entire covariance matrix.

3.5.2 The Naive Bayes probabilistic model

The probability model for a classifier is a conditional model

$$P(C|F_1,F_2,..,F_n) \qquad (3.3)$$

Over a dependent class variable C with a small number of outcomes or classes, conditional on several feature variables F1throughFn. The problem is that if the number of features is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, we write

$$P(C|F_1,F_2,..,F_n) = \frac{P(C)P((F_1,F_2,\dots,F_n)|C)}{P(F_1,F_2,..,F_n)} \qquad (3.4)$$

In plain English the above equation can be written as,

$$posterior = \frac{(prior * likelihood)}{evidence} \qquad (3.5)$$

In practice we are only interested in the numerator of that fraction, since the denominator does not depend on and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model,

$$P(C,F_1,F_2,..,F_n)$$

This can be rewritten as follows, using repeated applications of the definition of conditional probability:

$$P(C,F_1,F_2,..,F_n) = P(C)P(F_1|C)P(F_2|C,F_1)P(F_3|C,F_1,F_2),..,P(F_n|C,F_1,F_2,..,F_{n-1}) \qquad (3.6)$$

Now the "Naive" conditional independence assumptions come into play: assume that each feature Fi is conditionally independent of every other feature F_j for j≠i. This means that$P(F_i|C,F_j) = P(F_i|C)$ for I≠j, and so the joint model can be expressed as

$$P(C,F_1,F_2,..,F_n) = P(C)\prod_{i=1}^{n} P(F_i|C) \quad (3.7)$$

This means that under the above independence assumptions, the conditional distribution over the class variable can be expressed like this:

$$P(C,F_1,F_2,..,F_n) = \frac{1}{z}P(C)\prod_{i=1}^{n} P(F_i,C) \quad (3.8)$$

3.6 Support Vector Machine

Support Vector Machine is a framework of structural risk minimization and statistical learning theory developed by Vapnik and his co-workers [12]. SVM is based on the optimal classification hyperplane of linear classification situation. SVM finds a maximum margin separating hyper plane between two classes of data as shown in figure 3.4. It is a non-linear function and density estimation based algorithm.

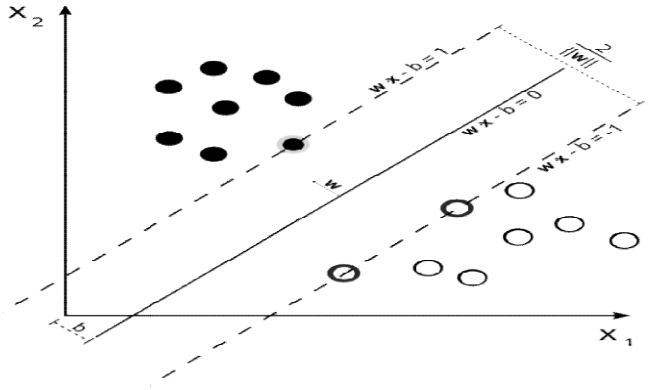


Figure 3.4: Support Vector Machine

In [29], it was indicated that classifier built on SVM shown promising results with its efficiency and effectiveness. SVM can be achieved by non-linear mapping, polynomial functions and sigmoid functions. SVM bypasses the curse of dimensionality by employing kernel functions and enables the straight forward analysis of high-dimensional data, the ability to determine the margin completely as well as the capability of handling high

dimensionality and small sample problems [29]. SVM has a great generalization capability too.

In SVMs hyper-plane that are separate the training data are measured by the maximum margin, therefore all vectors that lie on one side of the hyperplane are labeled as
-1 (w x – b= - 1) and the other side as
+1 (w x – b = 1). Thus when new data is introduced, it maps to the closest support vector based on the maximum margin.
To find the maximum margin, the following algorithms are used, given linear separable vectors $x_i(i = 1, \dots\dots\dots, l), x_i \in R^n$ with labels $y_i = \pm 1$, and for linearly separable space; the decision surface is a hyperplane which can be written as:

$$w * x + b = 0 \tag{3.9}$$

And the equation is

$$g(w) = \frac{1}{2}||w||_2^2 \tag{3.10}$$

With a constant

$$y_i(w * x_i + b) \geq 1, i = 1, \dots\dots\dots, l \tag{3.11}$$

The optimal hyperplane calculation as follows

$$\sum_{i=1}^{l} y_i a_i (x_i * x) + b_0 = 0 \tag{3.12}$$

3.6.1 Multi-Class Classification

There are established methods of using SVM for multi-class classification. Most commonly an ensemble of binary (two-class) classifiers is used for this problem. In such an ensemble each classifier is trained to recognize one particular category versus all other categories. For each category the classifiers are trained on a labeled set where the documents from the corresponding category make up positive examples and rest of the documents become negative examples. In order to classify new data, each classifier in the ensemble is used to produce a confidence value of a point belonging to the corresponding category and the category with greatest confidence is selected.

3.7 Categories of classification for Nepali documents

For the empirical evaluation of the hypothesis, Nepali document database is created by collecting documents from the Nepali newspaper. There are six classes of documents:

1. Business
2. Crime
3. Education
4. Health
5. Sport
6. Politics

More detail about the collected datasets is given in the Section 5.1.

3.8System Evaluation Measures

The correctness of a classification can be evaluated by computing the number of correctly recognized class examples (true positives), the number of correctly recognized examples that do not belong to the class (true negatives), examples that either were incorrectly assigned to the class (false positives) and examples that were not recognized as class examples (false negatives).These four counts constitute a confusion matrix [20]. Measures for multi-class classification based on a generalization of the measures of binary classification for many classes C_i are given below. Where, tp_i represent true positive for class C_i, fp_i represent false positive for class C_i, fn_i represent false negative for class C_i, tn_i represent true negative for class C_i, and μ represent micro averaging.

3.8.1Average System Accuracy

Average system accuracy evaluates the average per-class effectiveness of a classification system.

$$\text{Average Accuracy} = \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + f_i + fp_i + tn_i}}{l}$$

3.8.2 Precision

Precision (also called positive predictive value) is the number of correctly classified positive examples divided by the number of examples labeled by the system as positive.

Micro precision is the agreement of the data class labels with those of classifiers if calculated from sums of per-test decisions.

$$Precision_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} tp_i + fp_i}$$

3.8.2 Recall

Recall (also called sensitivity) is the number of correctly classified positive examples divided by the number of positive examples in the test dataset.

Micro recall is the effectiveness of a classifier to identify class labels if calculated from sums of per-test decisions.

$$Recall_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} tp_i + fn_i}$$

Chapter 4

PROGRAM DEVELOPMENT AND IMPLEMENTION

4.1 Development Methodology and Tools

4.1.1 Programming Language Used

Java is a powerful object-oriented programming language introduced by Sun Microsystems in 1995, which has built-in support to create programs with a graphical user interface (GUI), utilize the Internet, create client-server solutions, and much more. Programs written in Java can run, without change, on any of the common computer operating systems Windows 95/NT, Macintosh, and Unix. A variant of Java programs called applets can be embedded inside a web page and execute on the computer that is viewing the page, automatically and in a secure environment.

As a language, Java is closely related to C++, which is also object-oriented but retains a lot of idiosyncrasies inherited from its predecessor language C. Java has removed the inconsistent elements from C++, is exclusively object-oriented, and can be considered a modern version of C++ Because of its logical structure Java has quickly become a popular choice as a teaching language, and because of its extensive Internet support and the promise of writing programs once and using them on every operating system Java is becoming more and more accepted in industry.

4.1.2 Eclipse IDE

The Eclipse Platform is specially designed for building integrated development environments (IDEs), and arbitrary tools. The Eclipse Platform's principal role is to provide tool providers with mechanisms to use, and rules to follow, that lead to seamlessly-integrated tools. These mechanisms are exposed via well-defined API interfaces, classes, and methods. The Platform also provides useful building blocks and frameworks that facilitate developing new tools. It contains large set of functionality required to build an IDE. It supports both GUI and non-GUI based application development environments and runs on a wide range of operating systems, including WindowsR , LinuxTM, Mac OS X, Solaris AIX and HP-UX. Eclipse Platform enables the tool or application to integrate with other tools and applications also

written using the Eclipse Platform. The Eclipse Platform is turned in a Java IDE by adding Java development components (e.g. the JDT) and it is turned into a C/C++ IDE by adding C/C++ development components (e.g. the CDT). It becomes both a Java and C/C++ development environment by adding both sets of components. However, the Eclipse Platform is itself a composition of components; by using a subset of these components, it is possible to build arbitrary applications. Hence, Eclipse SDK, Eclipse Rich Client Platform (RCP) and Eclipse IDE are popular framework which is widely used.

- The Eclipse SDK includes Java Development Tools and Plug-in Development Environment.

- It includes building applications that work in conjunction with application servers, databases and other backend resources to deliver product providing a rich and consistent experience for its users.

- Eclipse IDE is designed to Support the construction of a variety of tools for application development. It supports tools to manipulate arbitrary content types (e.g., HTML, Java, C, JSP, EJB, XML, and GIF).

4.2 Machine Learning Library and Plug-ins

A piece of program or application that is use to create, debug, maintain and support other applications is called Programming tool. It is a simple program which is integrated together to accomplish a task or support other program, to fix application. They make easier to do some specific tasks such as IDE combine the features of many tools in one package to develop applications. Another term is Plug-in, which is a software component that is used to support a specific feature to an existing software application, to enable customization. There are lots of tools and plug-ins is available.

4.2.1 Weka

Weka (Waikato Environment for Knowledge Analysis) is a comprehensive and free available suite of Java class libraries that support the implementation of machine learning algorithms for data mining tasks. Weka contains tools for: data pre-processing, classification, regression,

clustering, association rules, and visualization. It allows users to apply Weka class libraries of machine learning techniques to their own data regardless of computer platform. It is developed in Java platforms to support data mining tools, a suite of Java packages to provide facilities for developers. The core package contains classes that are accessed from almost every other class in Weka.The most important classes in it are Attribute, Instance, and Instances. An object of class “Attribute” represents an attribute-it contains the attribute’s name, its type, and, in case of a nominal attribute, it’s possible values. An object of class “Instance” contains the attribute values of a particular instance; and an object of class Instances contains an ordered set of instances-in other words, a dataset. Weka is used for Naive Bayes Classifier Training and Testing.

Chapter 5

EXPERIMENTATIONS AND RESULTS

Multi class Nepali text classification system is experimented by creating two training dataset and one test dataset of Nepali text documents of five classes. This chapter describes the datasets and data dictionaries used in the experiments and corresponding empirical results. Training and testing datasets are described the Section 5.1 and all other NLP data dictionaries used in the system are described in the Section 5.2. Experimentation results and graphical analysis are described in the Section 5.3.

5.1 Training and Testing Datasets

We have collected five classes of Nepali text documents for system evaluation. Documents are collected from various online sources, such as www.onlinekhaber.com, www.nagariknews.com, www.swasthyakhabar.com and www.nepalhealthnews.com and various offline sources, such as books, manuscripts, and articles.

1 Crime:

Crime class of documents contains information about unlawful acts or punishable acts. Criminal acts may include murder, rape, theft etc. Sample documents of crime class are given in Figure 5.1.

1	बैतडीमा सामुहिक बलात्कारमा संलग्न दुई जनालाई १३ वर्षको जेल सजाय भएकोछ। जिल्ला अदालत बैतडीले सामुहिक बलत्कारको अभियोगमा पक्राउ परेका दुईजनालाई करिव एक वर्षपछि मंगलबार फैसला सुनाएको हो।एक किशोरीलाई बलात्कार गर्ने दुई जनालाई १३ वर्षको जेल सजायसहित जनही ५० हजार रुपैया जरिवानाको फैसला भएको छ। अदालतका न्यायाधीश हिमालय राजपाठकको एकल इजलाशले एक किशोरीलाई सामुहिक बलात्कार गरेको भन्दै गुजरगाविस ३ का २८ वर्षीय अम्मरराज अवस्थी र भुमेश्वर गाविस २ का सागर भट्टमाथिफैसला सुनाएको हो।
2	कानुन र नेपाल राष्ट्र बैंकको निर्देशनविपरीत बैभवकै पुराना ऋणी रोयल सारी इन्टरप्राइजेजका प्रोपाराइटर मनोजकुमार चौरसियाको एच एन्ड बि डेभलपमेन्ट बैंकको कुलेश्वर शाखामा रहेको खाताबाट बैभव फाइनान्सको नाममा खिनिएको 'गुड फर पेमेन्ट चेक' धितो राखेर कर्जा दिएको खुलेको छ। ब्युरोका अनुसार उनले १२ थान 'गुड फर पेमेन्ट चेक' धितो राखेर प्रतिव्यक्ति पचास लाखका दरले १२ जनाको नाममा मनोजकुमार चौरसियाको व्यक्तिगत जमानीमा कर्जा दिएका थिए।
3	दशगज्जा मिचेर भारतीयले बनाएको कलर्भट, सडक तथा घरको दृश्य खिचेर फर्किदै गरेका नागरीक दैनिक सप्तरीका संबाददाता जितेन्द्रकुमार झा, कान्तिपुर टेलिभिजन संबाददाता आरएन विश्वास र लालपट्टी गाविसका स्थानीय बासिन्दा एवं नेपाल-भारतमैत्री युवा संघ सप्तरीका अध्यक्ष किशोर कुमार यादवलाई हतियारसहित रहेको एसएबीका दुई जवानले झण्डै दुई किलोमिटर उत्तर लालापट्टी गाविस-४ वाट नियन्त्रणमा लिइ भारतको राजपुरस्थित अस्थाइ चौकीमा लगेका थिए।
4	सोर्स कोड चोरी गरी नक्कली सफ्टवेयर बनाएर ठगी गरेको अभियोगमा सात जना पक्राउ परेका छन्। पक्राउ पर्नेमा दुई जना सफ्वेयर कम्पनीका सञ्चालक र अन्य डेभलपर्स छन्। सहकारी संस्थाको दैनिक कारोवारको विवरण राख्ने सक्कली सफ्वेयरको चोरी गरी कारोबार गरेको अभियोगमा केन्द्रीय अनुसन्धान ब्युरोले बिहीबार देवेन्द्र खत्री, उमेश खत्री, पालसकुमार घोष, प्रमनराज शाक्य, मनोज यादव, सुनिल पोटे र सरोज सुवाललाई पक्रेको हो।
5	जिल्ला प्रहरी कार्यालय नतलपरासीले नेपाल टेलिकम र एनसेल मोबाइलको रिचार्ज कार्ड चोर्ने चारजनालाई पक्राउ गरेको छ ।

Figure: 5.1: Sample Crime Documents

2 Education:

Education class contains all the documents related to academic stuffs which contain knowledge, skills, and habits and teaching, training, or research strategies. Sample documents of education class are given in Figure 5.2.

1	त्रिभुवन विश्वविद्यालय परीक्षा नियन्त्रण कार्यालय बल्खुद्वारा विसं २०६९/०७० चैत/वैशाखमा सञ्चालित मानविकी तथा सामाजिकशास्त्र सङ्कायतर्फ स्नातक तह तीन वर्ष विए दोस्रो वर्षको परीक्षाको फरीक्षाफल आज प्रकाशन गरिएको छ।
2	कुन विद्यालय र अस्पताल भवन भूकम्पीय र अन्य प्रकोपको कति जोखिममा छ भन्ने पता लगाउने सफ्टवेयर तयार भएको छ। युएन ह्याबिटाटले गुएनआइएसडिआर र सार्कको सहगोगमा विद्गालग र अस्पताल भवनको जोखिमबारे जानकारी दिने सफ्टवेगर तगार पारेको हो। - उक्त सफ्टवेगरमा भरिएका सूचनाका आधारमा जोखिमको मात्रा पता लगाउन सकिने भएको हो । ह्याबिटाटले उक्त सफ्टवेयरलाई 'रेट्रो मेन्टेनेन्स एसेसमेन्ट टुलिकिट' नाम दिएको छ ।
3	फेसबुक आएपछि अभौतिक एल्बमले ठाउँ लियो। तपाईंका एल्बमबाट फोटो झिकिने दिन सकिए। अझ भनौं, तपाईंको घरमा एल्बम राख्ने चलन नै सकिन थाल्यो। फोटो धुलाउने र साट्ने परम्परा समाप्त भयो। सामाजिक सञ्जालमा रहेको फोटोमा ट्याग गरिदिने प्रचलन सुरु भयो। फोटो हराउने पिर त गयो–गयो, धुलाउने परम्परा पनि सकियो। फोटो सेयर गर्ने र टाइमलाइनमा समावेश गर्ने प्रचलनले सबैभन्दा बढी त फोटोको व्यापार गर्नेलाई असर पार्योप। अनौपचारिक गतिविधिका फोटो धुलाउने प्रचलन अहिले समाप्तप्रायः भएको छ।
4	कूल जनसंख्याको ८१ दशमलव ७२ प्रतिशतमा टेलिफोनको पहुँच पुगेको छ। नेपाल दूरसञ्चार प्राधिकरणले मंगलबार सार्वजनिक गरेको गत साउनसम्मको तथ्यांकअनुसार नेपालमा २ करोड १६ लाख ५१ हजार १ सय २२ जना टेलिफोन प्रयोगकर्ता छन्।
5	मंगलको माटोमा पानी भएको क्युरियोसिटीको अनसुन्धानकर्ता लारी लेसिनले 'साइन्स म्यागजिन' सँग बताएकी छिन्। एक घन फुट धुलोमा १ सय डिग्रीसम्म ताप दिने हो भने भने दुइ पोइन्ट अर्थात करीब एक लिटर पानी प्राप्त हुने उक्त म्यागेजिनमा उल्लेख गरिएको छ ।

Figure: 5.2: Sample Education Documents

3 Health:

Health class of dataset contains all the documents that are related to illness, injury, pain and diagnostics. It also contains documents related to nutrition, health care and health educations. Sample documents of health class are given in Figure 5.3

1	धरानका यातायात मजदुर बुद्धि विश्वकर्मा अचानक बिरामी परे। चिनेजानेकाको सल्लाहमा उनले आयुर्वेदिक औषधि खान थाले। तर, रोग निको भएन। बीपी कोइराला स्वास्थ्य विज्ञान प्रतिष्ठानका डाक्टर सञ्जीवकुमार शर्माले सञ्चालन गरेको घरदैलो स्वास्थ्य शिविरमा जँचाउन गएपछि उनले आफ्ना मिर्गौला खराब भएको थाहा पाए।
2	भिटामिन डीले शरीरका हाड बलियो बनाउने त पहिलेदेखि नै सबैलाई थाहा भएकै कुरा हो। तर हालै गरिएको एउटा अनुसन्धानले गर्भवती आमाले भिटामिन डीको केही बढी मात्रा सेवन गरेमा बालबालिकाको मांसपेशीको विकास राम्रो हुने देखाएको छ। -
3	गर्भावस्थामा भिटामिन डीको कमी भए छोराछोरी युवावस्थामा पनि कमजोर, वृद्धावस्थामा पनि मांसपेशी कमजोर हुनसक्छ। साथै कमजोर स्वास्थ्यलगायत मधुमेह, हाड भाँचिनेलगायत बेलाबेला लड्नेजस्ता समस्या आउने डक्टर हार्वे बताउँछन्। हाल अनुसन्धान समूहले गर्भावस्थामा भिटामिन डीको अतिरिक्त खुराक र जन्मिने बालबालिकाको हाड तथा मांसपेशीमा पर्ने प्रभाव थाहा पाउन १२ सय गर्भवती महिलामा अध्ययन गरिरहेको छ।
4	स्वास्थ्य मन्त्रालयसँग अनुमति नै नलिई आफूखुसी खोप दिँदै हिँड्ने संघ–संस्था पनि फेला परेका छन्। त्यस्ता संघ–संस्थाले गुणस्तरको न्यूनतम मापदण्डसमेत पूरा गरेका हुँदैनन्। डाक्टरहरूका अनुसार एन्टिभाइरल खोपलाई चिस्याएर राखिएको हुनुपर्छ। तापक्रम अलिकति पनि तलमाथि भए प्रभावकारिता शून्यमा झर्छ। टोलटोलमा परिचयपत्र बाँड्दै खोप दिनेले यस्तो 'रेफ्रिजेरेसन'को व्यवस्था गरेका छन् कि छैनन् भन्ने अनुगमन भएको छैन। हचुवा भरमा खोप लिँदा हेपाटाइटिसबाट सुरक्षित भएँ भनी भ्रम सिर्जना हुन्छ भने अर्कातिर रोगको जोखिम झन् बढ्दै जान्छ। -
5	शरीरलाई फिट र फाइन बनाउन नियमित व्यायाम गर्ने । तपाई योगको पनि सहारा लिन सक्नुहुन्छ । वेट ट्रेनिंग, टिवस्ट कल्स र डिप्स जस्ता एक्सरसाइज गर्नाले शरीरमा रक्तसञ्चार छिटो हुन्छ र धेरै ओक लाग्दछ ।

Figure: 5.3: Sample Health Documents

4 Sports:

Sports class of dataset contains the documents related to sports. It includes sport competitions, tournaments, and other related news and events. Sample documents of sports class are given in Figure 5.4.

1	जितपछि रियल मड्रिडको २५ खेलबाट ६३ अंक भएको छ । समान खेलबाट ६० अंक जोडेको बार्सिलोना भने दोस्रो स्थानमा झरेको छ । तेस्रो स्थानको एथलेटिको मड्रिडको पनि ६० अंक छ र उसले एक खेल कम खेलेको छ ।घरेलु मैदान सान्टिएगोमा भएको खेलमा रियल मड्रिडले ३४ औं मिनेटमा इलरामेन्डीको गोलबाट अग्रता लिएको थियो । खेलको ७२ औं मिनेटमा ग्यारेथ बेलले करिव ३० यार्ड टाढाबाट गोल गरे । खेलको ८१ औं मिनेटमा इस्कोले गोल गरेपछि रियल मड्रिड ३–० ले विजयी भयो ।
2	मनाडसँगै मच्छिन्द्र र थ्रीस्टारको उपाधि सम्भावना छ । तर, २६ अंकका साथ चौथो स्थानमा रहेको पुसिल र २३ अंकका साथ पाँचौं स्थानमा रहेको संकटाको उपाधिको सम्भावना लगभग सकिएको छ । त्रिपुरेश्वरस्थित दशरथ रंगशालामा भएको खेलमा संकटाले मनाडलाई पहिलो हाफसम्म गोलरहित बराबरीमा रोकेको थियो । खेलको ११ औं मिनेटमा रुपेश केसीको बल पोलमा लाग्दा मनाङ गोल गर्न चुकेको थियो । तर, ४८ औं अनिल गुरुङले गोल गर्दै मनाडलाई सुरुवाति अग्रता दिलाए । त्यसको टुई मिनेटमा योना इलियासले गोल गरेपछि मनाङ २-० को अग्रता लिन सफल भयो ।
3	खेलको २२ औं मिनेटमा कप्तान सन्तोष साहुखलले गोल गर्दै थ्रीस्टारलाई सुरुवाती अग्रता दिलाए । तर त्यो अग्रता आधा घण्टा पनि टिकाउन थ्रीस्टार असफल रह्यो । खेलको ५३ औं मिनेटमा एपीएफका भेट्टान स्ट्राइकर गणेश लावतीले गोल गर्दै खेल १-१ को बराबरीमा ल्याए । उपाधि होड कायमै राख्न एपीएफ विरुद्ध जित निकाल्नै पर्ने दबाबमा रहेको थ्रीस्टारले दोस्रो हाफमा गोल खाएपछि केही आक्रामण खेल प्रस्तुत गरेको थियो ।
4	हामी गोल गर्न नसक्ने जस्तो देखिएको छौं। मलाई पनि अप्ठ्यारो लागिसकेको छ। म आफै स्ट्राइकर। सबै रामो खेल्न सिकाएर खेलाडीलाई गोल गर्न नसिकाए जस्तो भएको छ।' सुपरसिक्समा टिमको फिनिसिङ सबभन्दा कमजोर देखिएको थापाले बताए। एपिएफले अब लिगको अन्तिम खेल फागुन १६ गते पुलिससँग खेल्नेछ। बल पोसेसनमा अगाडी देखिए पनि पहिलो हाफमा एपिएफले मछिन्द्रको पोस्टमा एउटामात्र गतिलो प्रहार गर्न सक्यो। कोइलापानी पोलस्टार हुदै एपिएफमा आवद्ध भएका जाडबु शेर्पाले १८औं मिनेटमा गरेको लामो प्रहार पोस्ट नजिकैबाट बाहिरियो। मछिन्द्रले पनि पहिलो हाफमा गोल गर्ने एउटा अवसर खेर फाल्यो
5	त्यसपछि एलेक्स डुलान, कप्तान माइकल क्लार्क, स्टेभन स्मिथ र ब्राड हेडिन १–१ रनमा आउट भए। शन मार्शले १ बलमात्र खेल्ने सके। मिचेल जोन्सन र रायन ह्यारिसले ६–६ रन बनाए। दक्षिण अफ्रिकाका डेल स्टेयनले ४ विकेट लिए। अघिल्लो दिन घाइते भएका वायने पार्नेलविना खेलेको दक्षिण अफ्रिकाबाट भेर्नोन फिलान्डरले २ तथा मोर्ने मोर्केल, डुमिनी र डिन एल्गरले १–१ विकेट लिए।

Figure: 5.4: Sample Health Documents

5 Business:

Business class of the text documents contains information about the trade of goods and/or services. Sample documents of business class are given in Figure 5.5.

1	स्थानीयस्तरमा उत्पादित माछालाई बजारसम्म पुऱ्याउन र उत्पादन वृद्धि गर्न आवश्यक रहेको मत्स्य विकास निर्देशनालयका कार्यक्रम निर्देशक रमानन्द मिश्रले बताए । “सरकारले माछामा लगानी बढाउँदै लगेको छ,” उनले भने, “उत्पादक र व्यवसायीसमेत थप सक्रिय भएर लागे बजार विस्तार गर्न सजिलो हुन्छ ।” सरकारले माछा उत्पादन गर्न चाहनेलाई मिसन फिस कार्यक्रममार्फत अनुदान सहयोगसमेत गरेको उनले बताए ।
2	महोत्सवको आयोजना नेपाल मत्स्य व्यसायी संघ र नेपाल फिसरिस सोसाइटीले गरेका हुन् । मत्स्य विकास निर्देशनालयअन्तर्गतको मत्स्य विकास कार्यक्रम, कृषि व्यवसायी प्रवद्धन कार्यक्रम, कृषि अनुसन्धान परिषद् (नार्क), कृषि तथा वन विश्वविद्यालय, त्रिभुवन विश्वविद्यालय र नेपाल उद्योग वाणिज्य महासंघको कृषि उद्यम केन्द्र सहआयोजक थिए ।
3	बुटवल फर्निचर उद्योग संघको शनिबार बुटवलमा सम्पन्न सातौं अधिवेशनका अवसरमा बोल्दै उद्योगीहरुले विना दर्ता संचालित उद्योगहरुका कारण एकातिर राज्यको कर गुमेको र अर्कोतिर अस्वस्थ प्रतिस्पर्धा बढेको गुनासो समेत गरे ।
4	नेपाली व्यापारीहरुले काठमाण्डौंबाट ल्याएको चाइनिज सामानका मुख्य ग्राहक भारतीय नै भएको कञ्चनपुर उद्योग वाणिज्य संघका निवर्तमान उपाध्यक्ष जनकराज भट्ट बताउँछन् । सिद्धनाथ इम्पेक्सका संचालक रहेका उनी भन्छन् “भारतीय पर्यटकले प्रायः चिनियाँ सामान मन पराउँछन् ।” अरु बेलाभन्दा हिउँदमा भारतीय पर्यटक बढी आउने गरेको उनी बताउँछन् । “भारतीय ग्राहकले चाइनिज ज्याकेट, ट्राउजर, झोला, जुत्ता, मखमली कपडा, कम्बल, सल, कस्मेटिक तथा इलेक्ट्रोनिक सामान किन्ने गरेका छन् ।” उनले भने “धनगढी र महेन्द्रनगरका रहेका फैन्सी पसलका ५० प्रतिशत सामानका उपभोक्ता भारतीय पर्यटक नै हुन् ।”
5	भारतको केन्द्रिय राजधानी दिल्लीसहित हरियाणा, पञ्जाब, राजस्थान, उत्तरप्रदेश र उत्तराखण्डबाट घुम्न आउनेहरुलाई लक्ष्य गरेर यहाँ पसल र होटेल खुल्ने क्रम पनि बढेको छ । अधिकांश भारतीय पर्यटकहरु परिवारसाथ आफ्नै वाहनमा आउने गरेका छन् । भारतीयहरु आगमनको रेकर्ड राख्ने चलन नभएकाले उनीहरुको यकिन तथ्यांक भने कुनै निकायसँग छैन ।भारतीयहरु कञ्चनपुर र कैलालीका अधिकांश पर्यटकीयस्थलको भ्रमण गरेर फर्कने गरेका छन् भने केही पहाडी जिल्लासम्म पनि पुग्ने गरेका छन् ।

Figure: 5.5: Sample Business Documents

5.1.1Training Datasets

5.1.1.1 Dataset I

Statistics about dataset I are given in the Table 5.1.Data set I consist of namely five classes of Nepal news text document collected from different data sources like news paper, books and other offline sources .As described in table there are 75 numbers of crime document, 120 numbers of business document, 140 number of health document and 171 number of sport document and 125 numbers of education document.

Table 5.1: Dataset I

Classes	**No. of samples**
Crime	75
Business	120
Health	140
Sports	171
Educational	125
Totals	531

5.1.1.2 Dataset II

Statistics about dataset II are given in the Table 5.2.Data set II consist of namely five classes of Nepal news text document collected from different data sources like news paper, books and other offline sources .As described in table there are 70 numbers of crime document, 127 numbers of business document, 150 numbers of health document and 171 numbers of sport document and 130 numbers of education document.

Table 5.2: Dataset II

Classes	**No. of samples**
Crime	70
Business	127
Health	150
Sports	171
Educational	130
Totals	648

5.1.2 Testing Dataset

Statistics about test dataset is given in the Table 5.3.Test data is used for analyzing system accuracy. As described in table testing data set, it consist of five classes of documents namely crime, business, sports, health and education. The testing data set have 15 numbers of crime document,20 numbers of business document,10 numbers of health document,20 numbers of sports document and 12 numbers of education document .

Table 5.3: Test Dataset

Classes	No. of samples
Crime	15
Business	20
Health	10
Sports	20
Educational	12
Totals	77

5.2 Data Dictionaries

5.2.1 Stop Word Dictionary

Stop word dictionary contains all the common and less informative words. Un-useful words are removed from the document in pre-processing stage to make feature space smaller. The words in the document that matched with the words listed in the stop word dictionary are excluded. Some of the stop words from the stop word dictionary are given in the Figure 5.6.

छ म हो छु केही कोही हामी मेरो त्यो को हरु फेरी हाम्रो अर्को न कुनै लाइ तर अझै छौ सबै बुझे मैले र बुझ्यौं तिमीले किन के भो आइ एम मा कै का कि मै यो यी ले या श्री नै सो की वा भै जे लिए त्यसै त्यस भए एक अरु आफू आए बाट वाट छुटे हुन्छ जुन राख यहि लाई हेर्न आज भयो गर्दै गर्ने गर्न गर्नु पक्ष पनि भन्ने माथि गर्दा हाल रूप रहे सँग वाटै हरू ढंग तथ्य जोड चासो गत दिने पछि सानो घटे एवं कार्य मात्र भन्नु हिजो सम्म हुना भने कुरा त्यहीं लागि सोही तर्फ गए यस यसै अति लिई एक्लै हुँदा हुन बोले साथ राखे प्रति तथा दिनु तह आफ्नै तिर हुनु हने देखि छन् गरी बीच यता कति साथै यस्तै बारे नयाँ आयो पार्नु उनी आजै यहाँ भर भई दिई थिए गरे उक्त दिए जस क्रम होला लिन यस्तो लिनु छैन त्यहाँ जहाँ पर्ने अर्का यस्ता पर्छ हुदै गई उहां लगे उठे अर्कै होस् गरें होस अरू अब बन्नु उता सँघ थप हुँदै चले गयो फेरि अनि बने पाए जैले कैले अथवा उसको यसमा आफ्नो उनका रहयो भनिए नभए हुँदैन नगरे नभई अहिले तापनि समेत गर्नेछ गरियो यसरी गराई यसर्थ पर्दछ त छि हुन् उसका हामै

Figure 5.6: Stop word Dictionary

5.2.2 Symbol Dictionary

The symbols that don't carry the special meanings are removed in the pre-processing stage. Some of the special symbols are given in Figure 5.7.

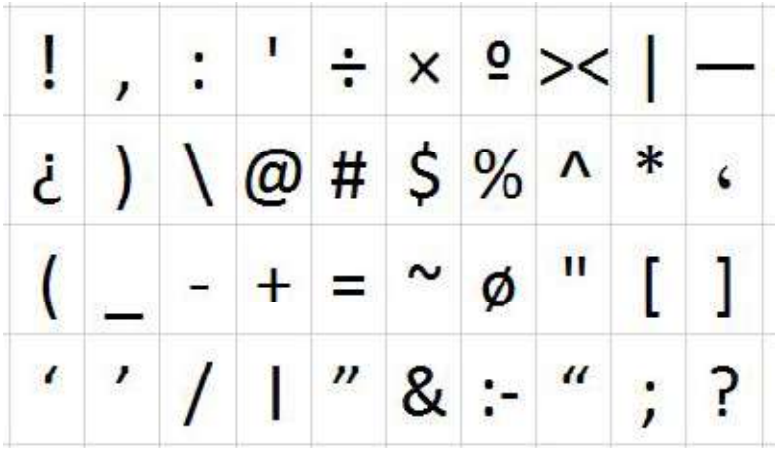


Figure 5.7: Symbol Dictionary.

5.3 Experimentation Results

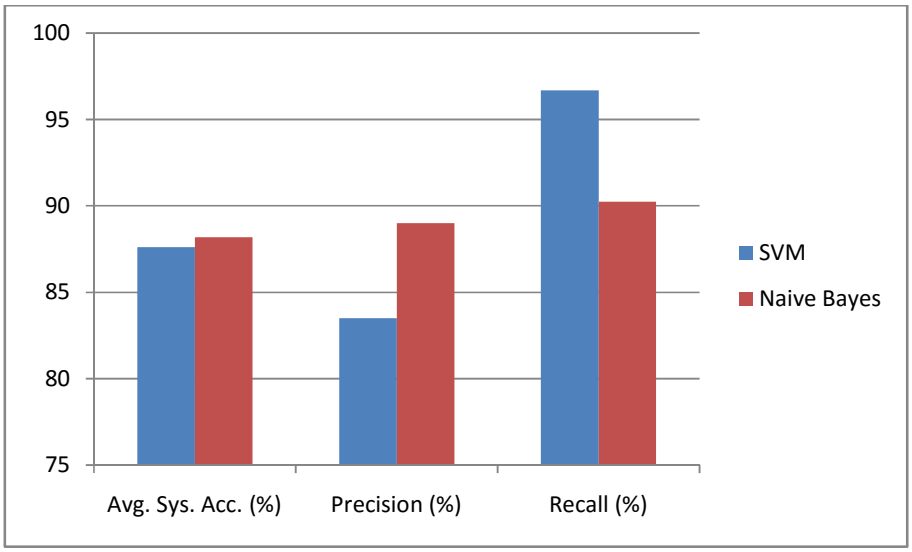
System is trained and tested against collected datasets described in the Section 5.1. Various performance matrices (Section 3.8) are evaluated. This section describes all the empirical results and analysis of the outcomes.

5.3.1 Experiment 1

First experiment is carried in Training Dataset I (Section 5.1.1.1) and Testing Dataset (Section 5.1.2). Experimentation results show Naive Bayes classifier performs better than Support vector machine based classifier. Table 5.4 shows results of the experiment 1. Figure 5.8 shows graphical representation of the results. Table 5.4: Experimentation Results (Experiment 1)

Algorithm	Avg. Sys. Acc. (%)	Precision (%)	Recall (%)
SVM	85.92	84.5	92.12
Naive Bayes	86.17	95.47	87.51

Figure 5.8: Graph of Experiment 1.

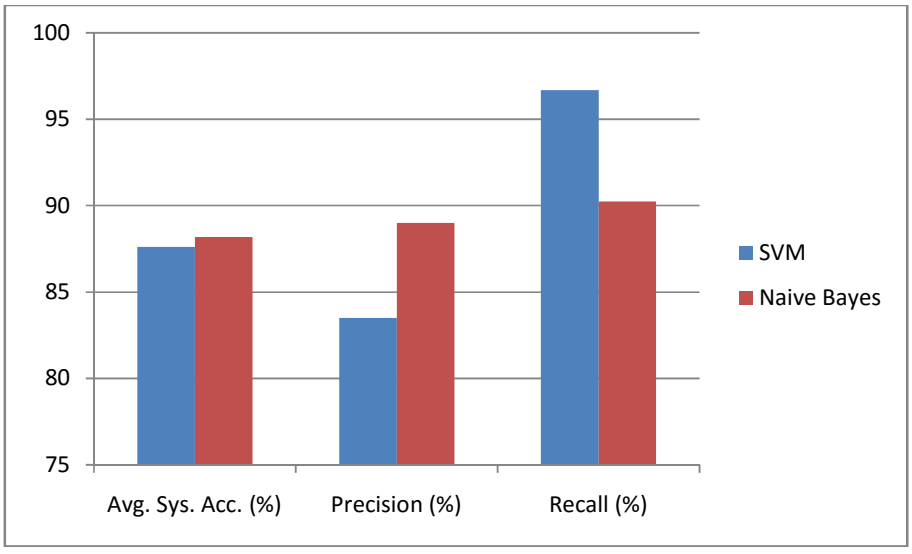


5.3.2 Experiment 2

Second experiment is carried in Training Dataset II (Section 5.1.1.2) and Testing Dataset (Section 5.1.2). Experimentation results show Naïve Bayes Classifier performs better than support vector machine based. Table 5.5 shows results of the experiment 2. Figure 5.9 shows graphical representation of the results. Table 5.5: Experimentation Results (Experiment 2)

Algorithm	Avg. Sys. Acc. (%)	Precision (%)	Recall (%)
SVM	87.6	83.5	96.70
Naive Bayes	88.18	88.99	90.24

Figure 5.9: Graph of Experiment 2.



5.4 Result Analysis

Aggregate results of both the experiments are shown in Table 5.6.

Table 5.6: Aggregate System Results

Algorithm	Avg. Sys. Acc. (%)	Precision (%)	Recall (%)
SVM	86.76	81.03	94.4
Naive Bayes	88.09	80.03	93.87

Results show Naïve Bayes based classifier has high system accuracy than support vector machine based classifier. System results greatly influenced by number of training and testing data and extracted features. Classifier parameters also play important roles for better learning of the system. Computational and efficiency effectiveness can be enhanced by code optimization and distributed computing. Due to the unavailability of standard training and test datasets, system performance cannot be generalized well. But, so far- so good, results are promising and can be enhanced further.

Chapter 6

CONCLUSION

6.1 Conclusion

An automatic multi class text classification problem for Nepali language is addressed in this dissertation work. As the solution of the stated problem, two machine learning based classification techniques are experimented and performance is measured for both the cases. Classification systems take input an unknown text document and assign to a known class among five classes (”Business”,”Crime”,”Education” ”Health”, “Sport”). Input text document ispassed through various pre-processing steps like stop-word removal, symbol removal and stemming. Then, fine grained document is passed into feature extractor, where term frequency based features are extracted. Feature vector is than fed to classification systems-which are previously trained with given datasets and given classes in supervised manner. Empirical results shows, Naïve Bayes classifier performs better than Support vector machine (SVM) based classifier. SVM classification system has the average system accuracy rate of 86.76%, precision rate of 81.03% recall rate of 94.4%. Similarly, Naive Bayes classification system has the average system accuracy rate of 88.09%, precision rate of 80.03% and recall rate of 93.87%.

6.2 Limitations and Future Scope

The performance of the proposed system may further be improved by improving pre-processing techniques. Exploring more features and enhancing data dictionaries can improve classification accuracy. System performance is greatly influenced by training and testing corpus. Classifier parameters also play important roles for better learning of the system. Computational and efficiency effectiveness can be enhanced by code optimization and distributed computing.

Due to the unavailability of standard training and test datasets, system performance cannot be generalized well. But, so far- so good, results are promising and can be enhanced further.

References

[1] Androutsopoulos, I., Koutsias, K., Chandrinos, K, V., Spyropoulos, C. D. (2000). An Experimental Comparison of Naïve Bayes and Keyword-based Anti-spam Filtering with personal email message, In*:* the Proceedings of 23rd ACM SIGIR, p.160-167.

[2] Bal Krishna Bal and Prajol Shrestha, "A morphological analyzer and a stemmer for nepali," Madan Puraskar Pustakalaya,Nepal ,2006.

[3] Cristianini, N. Shawe-Taylor, J. (2000). Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press.

[4] Drucker, H., Wu, D.Vapnik, V. N. (1999). Support Vector Machines for Spam Categorization, IEEE Transaction on Neural Networks, 10 (5) 1048-1054.

[5] Duda, R. O., Hart, P. E., Stork, D. G. (2001).Pattern Classification, John Wiley & Sons, Inc.

[6] Dennis Ramdass & Shreyes Seshasai, Document Classification for Newspaper Articles 6.863 Final Project Spring 2009 May 18, 2009

[7] Fabrizio Sebastiani.Text classification,automatic. In Keith Brown (ed.), *The Encyclopedia of* Language and Linguistics, 2nd Edition, Vol. 14, Elsevier Science, Amsterdam, NL, 2004

[8] Fabrizio Sebastiani. Text Classification for Web Filtering Final POESIA Workshop "Present and Future of Open-Source Content- Based Web filtering" Pisa, IT — 21-22 January, 2004

[9] F. Aiolli, R. Cardin, F. S. 0001, and A. Sperduti, "Preferential text classification: Learning algorithms and evaluation measures," ERCIM News, vol. 2009, no. 76, 2009.

[10] Hagan, M. T., Demuth, H. B., Beale, M. (1995).Neural Network Design, PWS Publishing Company.

[11] Haykin, S. (1994). Neural Networks: Comprehensive Foundation, Macmillan College Publishing Company.

[12] Hearst, M. (1998). Support Vector Machines, IEEE Intelligent Systems, 13 (4) 18-28.

[13] Jackson, P., Mouliner, I. (2002). Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization, John Benjamins Publishing Company.

[14] Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with many Relevant Features, *In:* the Proceedings of 10th European Conference on Machine Learning, p. 143-151.

[15] J. Han and M. Kamber, Data Mining: Concepts and Techniques.

[16] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C. (2002). Text Classification with String Kernels, *Journal of Machine Learning Research,* 2 (2) 419-444.

[17] Mitchell, T. M. (1997).Machine Learning, McGraw-Hill.

[18] Mladenic, D., Grobelink, M. (1999). Feature Selection for unbalanced class distribution and Naïve Bayes. *In:* the Proceedings of International Conference on Machine Learning, p. 256-267

[19] Massand, B., Linoff, G., Waltz, D. (1992). Classifying News Stories using Memory based Reasoning, *In:* the Proceedings of 15th ACM International Conference on Research and Development in Information Retrieval, p. 59-65.

[20] M. Sokolova and G. Lapalme,"A systematic analysis of performance measures for classification tasks," Inf. Process. Manage vol. 45, pp. 427–437, July 2009.

[21] P. Langley, W. Iba and K. Thompson, "An analysis of Bayesian Classifiers.*," in Proceedings of* the Tenth National Conference on Artificial Intelligence, San Jose, CA, 1992.

[22] R. O. Duda and P. E. Hart, Pattern classification and scene analysis, John Wiley and Sons, 1973.

[23] Ruiz, M. E., Srinivasan, P. (2002). Hierarchical Text Categorization Using Neural Networks, *Information Retrieval,* 5(1) 87-118

[24] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distributional word clusters vs.words for text categorization," Journal of Machine Learning Research, vol. 3, pp. 1183–1208, 2003.

[25] Sebastiani, F. (2002).Machine Learning in Automated Text Categorization, ACM Computing Survey, 34 (1) 1-47.

[26] S.Ramasundaram and S.P.Victor, "Text categorization by back propagation network," International Journal of Computer Applications, vol. 8, pp. 1–5, October 2010.Published By Foundation of Computer Science.

[27] Shahi Tej Bahadur, Yadav Abhimanu Yadav,"Mobile SMS Spam Filtering for Nepali Text Using Naïve Bayesian and Support Vector Machine" International Journal of Intelligence Science, 2014, 4, 24-28Published Online January 2014 (http://www.scirp.org/journal/ijis) http://dx.doi.org/10.4236/ijis.2014.41004

[28] T. Ayodele, S. Zhou, and R. Khusainov, "Email classification: Solution with back propagation technique," in ICITST, pp. 1–6, IEEE, 2009.

[29] Tong, S., Koller, D.: "Support Vector Machine Active Learning with Applications to Text Classification". In: J. Mach. Learn. Res. 2 (2002) 45–66

[30] Vandana Korde,"Text classification and classifiers: A survey", International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.2, March 2012.

[31] Wiener, E. D. (1995). A Neural Network Approach to Topic Spotting in Text, The Thesis of Master of University of Colorado.

[32] W. Y.-c. ZHANG Yun-tao, GONG Ling, “An improved tf-idf approach for text classification,”

[33] Yang, Y. (1999). An evaluation of statistical approaches to text categorization, Information Retrieval, 1 (1-2) 67-88.

Appendix A

Sample System Outputs

Sample Input

1	धरानका यातायात मजदुर बुद्धि विश्वकर्मा अचानक बिरामी परे । चिनेजानेकाको सल्लाहमा उनले आयुर्वेदिक औषधि खान थाले तर रोग निको भएन । बीपी कोइराला स्वास्थ्य विज्ञान प्रतिष्ठानका डाक्टर सञ्जीवकुमार शर्माले सञ्चालन गरेको घरदैलो स्वास्थ्य शिविरमा जँचाउन गएपछि उनले आफ्ना मिर्गौला खराब भएको थाहा पाए ।
2	मिर्गौला रोग लागिसकेपछि यसलाई पूर्णतः निको पार्ने कुनै औषधि उपलब्ध छैन। मिर्गौला ९० प्रतिशतभन्दा बढी बिग्रेको अवस्थालाई पूर्ण रूपमा मिर्गौला फेल भएको मानिन्छ । मिर्गौलाको कार्यक्षमता ३५ देखि ५० प्रतिशतसम्म कम हुदासम्म रोगको लक्षण देखिँदैन। त्यसैले अधिकांशले रोग अन्तिम अवस्थामा पुगेपछि मात्र थाहा पाउँछन् । त्यस्तो बेलामा कि त मिर्गौला नै फेर्नुपर्ने हुन्छ कि डायलाइसिस गरेरो रगत सफा गरिरहनुपर्छ । यी उपचार पद्धति सामान्य आर्थिक हैसियतका बिरामीले धान्न नसक्ने महँगो छ । डायलाइसिसका लागि एक वर्षमा करिब १ लाख ५० हजारदेखि तीन लाख रुपैयाँसम्म खर्च लाग्छ भने मिर्गौला प्रत्यारोपण गर्न १० लाख रुपैयाँसम्म खर्च लाग्न सक्छ । यो भनेको नेपालीको औसत आम्दानीले धान्न नसक्ने अवस्था हो ।
3	मिर्गौला रोग लागेपछि सामान्यतः बिहान बढी अनुहार र खुट्टा सुन्निने भोक कम लाग्ने वाकवाकी लाग्ने कमजोरी महसुस हुने छिट्टै थाकेको अनुभव हुने कम्मरको तल्लो भाग दुख्नेजस्ता लक्षण देखिन्छन् । यो रोग लागेपछि पिसाब धमिलो वा रातो हुने रातमा धेरै पटक पिसाब लाग्ने गर्छ ।यस्ता लक्षण देखापरे बेलैमा जाँच गराउनुपर्छ समयमै रोग थाहा पाए उपचार गर्न सजिलो हुन्छ ।
4	नभावना नेपाल नामक संस्थाले ललितपुरस्थित पवन प्रकृति बोर्डिङ स्कुलमा विद्यार्थी र स्थानीय समुदायलाई भ्याक्सिन लगाउँदै गरेको अवस्थामा महानगरीय प्रहरी वृत सातदोबाटोको टोलीले भ्याक्सिनसहित दुई जनालाई समातेको थियो । स्वास्थ्य सेवा विभाग व्यवस्थापन महाशाखाले विभिन्न संस्थाले जथाभावी रुपमा हेपाटाइटिस भ्याक्सिन लगाउँदै गरेको भन्दै ०६८ माघ २ मा हेपाटाइटिस बी भ्याक्सिन लगाउन रोक लगाएको थियो ।

Symbol and Digit Removed

1	धरान यातायात मजदुर बुद्धि विश्वकर्मा अचानक बिरामी चिनेजानेकाको सल्लाह आयुर्वेद औषधि खान थाले रोग निको बीपी कोइराला स्वास्थ्य विज्ञान प्रतिष्ठा डाक्टर सञ्जीवकुमार शर्मा सञ्चालन घर स्वास्थ्य शिविर जँचाउन गएपछि आफ्ना मिर्गौला खराब
2	मिर्गौला रोग लागिसकेपछि पूर्ण निको पार् औषधि मिर्गौला बिग्र अवस्था पूर्ण मिर्गौला फेल मान् मिर्गौलाको रोग लक्षण देख् अधिकांशले रोग अवस्था पाउ त्यस्तो मिर्गौला फेर् डायलाइसिस रगत सफा गरिरहनुपर्छ उपचार पद्धति सामान्य आर्थिक हैसियत बिरामी धान् सक् महँगो डायलाइसिसका करिब लाख लाख रुपैयाँसम्म खर्च लाग् मिर्गौला प्रत्यारोपण लाख रुपैयाँसम्म खर्च लाग् भन् नेपाल औसत आम्दानी धान् सक् अवस्था
3	मिर्गौला रोग लाग् सामान्य अनुहार खुट्टा सुन्निने भोक लाग् वाकवाक लाग् कमजोर महसुस छिट्टै थाक् अनुभव कम्मर भाग दुख् लक्षण देख् रोग लाग् पिसाब धमिलो रातो रात पिसाब लाग् गर् लक्षण देखापरे बेल् जाँच् गराउनुपर्छ समयमै रोग उपचार
4	भाव नेपाल नामक संस्था ललितपुर पवन प्रकृति बोर्डिङ स्कुल विद्यार्थी स्थानीय समुदाय भ्याक्सिन अवस्था महानगरीय प्रहरी वृत टोली भ्याक्सिनसहित समात् स्वास्थ्य सेवा विभाग व्यवस्थापन शाखा संस्था जथाभावी हेपाटाइटिस भ्याक्सिन माघ हेपाटाइटिस भ्याक्सिन रोक

Stemming

1	धरान यातायात मजदुर बुद्धि विश्वकर्मा अचानक बिरामी चिनेजानेकाको सल्लाह आयुर्वेद औषधि खान थाले रोग निको भए बीपी कोइराला स्वास्थ्य विज्ञान प्रतिष्ठा डाक्टर सञ्जीवकुमार शर्मा सञ्चालन घर स्वास्थ्य शिविर जँचाउन गएपछि आफ्ना मिर्गौला खराब
2	मिर्गौला रोग लागिसकेपछि यस पूर्ण निको पार् औषधि मिर्गौला प्रति बिग्र अवस्था पूर्ण रूप मिर्गौला फेल मान् मिर्गौलाको कार्य प्रति दा रोग लक्षण देख् अधिकांशले रोग अवस्था पाउ त्यस्तो बेला मिर्गौला फेर् डायलाइसिस रगत सफा गरिरहनुपर्छ उपचार पद्धति सामान्य आर्थिक हैसियत बिरामी धान् सक् महँगो डायलाइसिसका वर्ष करिब लाख हजार लाख रुपैयाँसम्म खर्च लाग् मिर्गौला प्रत्यारोपण लाख रुपैयाँसम्म खर्च लाग् भन् नेपाल औसत आम्दानी धान् सक् अवस्था
3	मिर्गौला रोग लाग् सामान्य अनुहार खुट्टा सुन्निने भोक लाग् वाकवाक लाग् कमजोर महसुस छिट्टै थाक् अनुभव कम्मर भाग दुख् लक्षण देख् रोग लाग् पिसाब धमिलो रातो रात पिसाब लाग् गर् लक्षण देखापरे बेल् जाँच् गराउनुपर्छ समयमै रोग उपचार
4	भाव नेपाल नामक संस्था ललितपुर पवन प्रकृति बोर्डिङ स्कुल विद्यार्थी स्थानीय समुदाय भ्याक्सिन ल अवस्था महानगरीय प्रहरी वृत सात टोली भ्याक्सिनसहित समात् स्वास्थ्य सेवा विभाग व्यवस्थापन शाखा संस्था जथाभावी हेपाटाइटिस भ्याक्सिन ल माघ हेपाटाइटिस भ्याक्सिन रोक ल

Appendix B
Sample Source Code

Document Preprocessing

```
import com.nlp.dc.stemmer.StemmerStart;
import java.util.ArrayList;
import java.util.List;

public class Preprocessing{
public List<String> stopwordRomoval(List<String> stopwordDict,
List<String> input) {
// Stop word removal from input text.
List<String> output = remove(stopwordDict, input);
return output;
}
public List<List<String>> stopwordRomovalFromLL(List<String> stopwordDict,
List<List<String>> input) {
List<List<String>> output = remove1(stopwordDict, input);
return output;
}
public List<String> symbolRemoval(List<String> symbolDict,
List<String> input) {

// Special symbol removal from input text.
List<String> output = remove(symbolDict, input);
return output;
}
public List<List<String>> symbolRemovalFromLL(List<String> symbolDict,
List<List<String>> input) {
List<List<String>> output = remove1(symbolDict, input);
return output;
}
public List<String> conjunctionWordRemoval(List<String> conjunctionDict,
List<String> input) {
// Conjunction word removal from input text.
List<String> output = remove(conjunctionDict, input);
return output;
}
private List<String> remove(List<String> noise, List<String> input) {
int nRows = input.size();
List<String> output = new ArrayList<String>();

for (int i = 0; i < nRows; i++) {
List<String> tokenArray = new ArrayList<String>();
```

```
String[] tArray = input.get(i).split("[ ,'\"-()â€˜â€™]");

for (int j = 0; j < tArray.length; j++) {
if (!tArray[j].trim().isEmpty())
tokenArray.add(tArray[j].trim());
}
tokenArray.removeAll(noise);
if (!tokenArray.isEmpty()) {
StringBuilder builder = new StringBuilder();
for (String s : tokenArray) {
builder.append(s.trim());
builder.append(" ");
}
output.add(i, builder.toString().trim());
} else {
output.add(i, " ");
}

}
return output;
}

private List<List<String>> remove1(List<String> noise,
List<List<String>> input) {
int nClasses = input.size();
// System.out.println("nClass: "+nClasses);

List<List<String>> output = new ArrayList<List<String>>();
for (int i = 0; i < nClasses; i++) {
int nSamples = input.get(i).size();
// System.out.println("nSamples: "+nSamples);
List<String> sampleArray = new ArrayList<String>();
for (int j = 0; j < nSamples; j++) {
List<String> tokenArray = new ArrayList<String>();
String[] tArray = input.get(i).get(j).split("[ ,'\"()â€˜â€™!^à¥¤ï¿½@!#%&*{}.â€œâ€-]+");

// tokenArray=Arrays.asList(tArray);

if (tArray.length == 0) {
continue;
}
for (int k = 0; k < tArray.length; k++) {
if (!tArray[k].isEmpty() || !tArray[k].trim().isEmpty())
tokenArray.add(tArray[k].trim());
}
// System.out.println("tokenArray: "+tokenArray);
```

```
if (tokenArray.isEmpty()) {
continue;
}
tokenArray.removeAll(noise);
StringBuilder builder = new StringBuilder();
for (String s : tokenArray) {
builder.append(s);
builder.append(" ");
// System.out.println("s: "+s);
}

sampleArray.add(j, builder.toString());
tokenArray.clear();
}
output.add(i, sampleArray);

}
return output;
}
public List<List<String>> stemmerLL(List<List<String>> input) {
List<List<String>> output=new ArrayList<List<String>>();
int nClasses=input.size();
for(int iClass=0;iClass<nClasses;iClass++){
//System.out.println("Class: "+iClass);
int nSamples=input.get(iClass).size();
List<String> sTempOut = new ArrayList<String>();
List<String> sOut = new ArrayList<String>();
for(int iSample=0;iSample<nSamples;iSample++){
String sentence=input.get(iClass).get(iSample);
//                          System.out.println("Class: "+iClass+" Sample: "+iSample+" Document:
"+sentence);
if (sentence.length()>0) {
sTempOut = StemmerStart.stemmer(sentence);
//                                  System.out.println("sTempOut: "+sTempOut);
StringBuilder builder = new StringBuilder();
for (String s : sTempOut) {
builder.append(s);
builder.append(" ");
}
sOut.add(builder.toString());
}else{
sOut.add(" ");
}
}
output.add(iClass,sOut);
```

```
}
return output;
}
public List<List<String>> digitRemoval(List<List<String>> input) {
List<List<String>> output=new ArrayList<List<String>>();
int nClasses=input.size();
for(int iClass=0;iClass<nClasses;iClass++){
int nSamples=input.get(iClass).size();
List<String> sOut = new ArrayList<String>();
for(int iSample=0;iSample<nSamples;iSample++){
String sentence=input.get(iClass).get(iSample);
//                        System.out.println("Class: "+iClass+" Sample: "+iSample+" Document:
"+sentence);
if (sentence.length()>0) {
sentence= sentence.replaceAll("[à¥¦à¥§à¥¨à¥©à¥ªà¥«à¥¬à¥-à¥®à¥¯]+","");
sentence=sentence.replaceAll("[0-9]+", "");
sentence= sentence.replaceAll("[A-Za-z]+", "");
sentence=sentence.replaceAll("[Ã·+,:\\ï¿½'\"()\\â€˜\\â€™!^à¥¤ï¿½$@!#%&*{}.\\â€œ\\â€�\\
â€“\\-]+", "");
}
sOut.add(iSample,sentence);
}
output.add(iClass,sOut);
}
return output;
}
}
```

Feature Extraction

```
import java.util.*;
import java.util.Map.Entry;
public class FeatureExtraction{
        public int noOfWordsInLongestSentence(List<String> sArray) {
                // input is sentence wise array of input document
                int len = 0;
                int nSentences = sArray.size();
                String sentence;
                for (int j = 0; j < nSentences; j++) {
                        sentence = sArray.get(j);
                        String[] tokens = sentence.trim().split("[ ,]+");
                        if (j == 0) {
                                len = tokens.length;
                        }
                        if (tokens.length > len) {
                                len = tokens.length;
                        }
```

```
        }
        return len;
}
public int noOfWords(String sentence) {
        String[] tokens = sentence.trim().split("[ ,]+");
        return tokens.length;
}

@Override
public double lengthOfSentence(int noOfWordsInLongestSentence,
                String sentence) {
        // It returns the relative length by counting the number of
        // words in a sentence and making its length relative
        // to the longest sentence in the current document.
        int noOfWordsInSentence = noOfWords(sentence);
        double output = (double) noOfWordsInSentence
                        / noOfWordsInLongestSentence;
        return output;
}


public double sentencePosition(List<String> input, int sPos, String sentence) {
        // It returns relative sentence position. Each sentence in the document
        // is given a rank ranging from 1 to max (max is the
        // number of sentences in the document). More weight is given to
        // sentence at the beginning than the rest. We compute sentence absolute
        // position feature using equation
        // input-sentence array
        // sentence-particular sentence
        double noOfSentences = input.size();
        if (noOfSentences == 0) {
                return 0.0;
        }
        // int sPos = input.indexOf(sentence);
        double absSPos = ((noOfSentences - sPos + 1) / noOfSentences);
        // System.out.println(absSPos);
        return absSPos;
}


public int neClass(List<List<String>> neDict, String word) {
        // 1-person 2-location 3-organization 4-misc 5-not NE
        // System.out.println(word);
        if (neDict.get(0).contains(word)) {
                // System.out.println(word);
                return 1;
```

```
        } else if (neDict.get(1).contains(word)) {
                return 2;
        } else if (neDict.get(2).contains(word)) {
                return 3;
        } else if (neDict.get(3).contains(word)) {
                return 4;
        } else {
                return 5;
        }
}


public int posTag(List<List<String>> posDict, String word) {

        return 0;
}


public int tf(List<String> tokens, String word) {
        // tokens-token list of given document
        // frequency of the word in the given document
        int tf = Collections.frequency(tokens, word);
        return tf;
}


public int isf(List<String> sentences, String word) {
        // input: sentences (sentence list of given document)
        // output: (number of sentences in which "word" occurs)
        int n = 0;
        for (Object o : sentences) {
                if (o.toString().contains(word)) {
                        n = n + 1;
                }
        }
        return n;
}

public double wordWeight(List<String> input, String word) {
        List<String> sentences = listSentencise(input); // sentence list
        List<String> tokens = listTokanize(input); // token list
        int ns = sentences.size(); // number of sentences
        int tf = tf(tokens, word);
        int isf = isf(sentences, word);
        double tf_isf = (double) tf * Math.log((double) ns / (double) isf);
        return tf_isf;
```

```
}

public double wordWeight1(List<String> sentenceArray,
                List<String> tokenArray, String token) {
        // Term Frequency Ã— Inverse Sentence Frequency (TF-ISF) for a word
        int ns = sentenceArray.size(); // number of sentences
        int tf = tf(tokenArray, token);
        int isf = isf(sentenceArray, token);
        double tf_isf = (double) tf * Math.log((double) ns / (double) isf);
        return tf_isf;
}


public List<Double> sentenceWeight(List<String> input) {
        // input is document list.
        List<Double> weights = new ArrayList<Double>();
        List<Double> tw = new ArrayList<Double>();
        List<String> sentences = listSentencise(input); // sentence list
        List<String> tokens = listTokanize(input); // token list
        int ns = sentences.size(); // number of sentences

        for (Object o : sentences) {
                double wi = sentenceWeight2(sentences, tokens, o.toString());
                tw.add(wi);
        }
        double maxw = Collections.max(tw);
        for (Object o : tw) {
                weights.add((Double) o / maxw);
        }
        return weights;
}

public double sentenceWeight1(List<String> sentenceArray, String sentence) {
        List<String> tokenArray = listTokanize(sentenceArray);

        double wi = 0.0;
        List<String> sentenceTokenArray = sentenceTokanize(sentence);
        int nTokens = sentenceTokenArray.size();
        for (int i = 0; i < nTokens; i++) {
                double twi = 0.0;
                String token = sentenceTokenArray.get(i).toString();
                twi = wordWeight1(sentenceArray, tokenArray, token);
                wi = wi + twi;
        }
        return wi;
}
```

```
public double sentenceWeight2(List<String> sentences, List<String> tokens,
		String sentence) {
	double wi = 0.0;
	String[] tt = sentence.toString().split("[ ,\t]");
	for (int i = 0; i < tt.length; i++) {
		double twi = 0.0;
		if (!tt[i].trim().isEmpty()) {
			twi = wordWeight1(sentences, tokens, tt[i]);
			wi = wi + twi;
		}
	}
	return wi;
}


public List<String> listTokanize(List<String> input) {

	List<String> temp = new ArrayList<String>();

	for (Object o : input) {
		String[] tt = o.toString().split("[ ,?à¥¤\t\n]");
		for (int i = 0; i < tt.length; i++) {
			if (!tt[i].trim().isEmpty()) {
				temp.add(tt[i]);
			}
		}
	}
	return temp;
}


public List<String> sentenceTokanize(String sentence) {
	List<String> temp = new ArrayList<String>();
	String[] tt = sentence.split("[ ,?à¥¤\t\n]");
	for (int i = 0; i < tt.length; i++) {
		if (!tt[i].trim().isEmpty()) {
			temp.add(tt[i]);
		}
	}
	return temp;
}


public List<String> listSentencise(List<String> input) {
```

```
	List<String> temp = new ArrayList<String>();

	for (String s : input) {
		if (s == null || s.length() == 1 || s.isEmpty()
				|| s.trim().equals("") || s.equals("")) { // Ashok: I don't

			// know why

			// s.length()==1

			// works but not

			// s.length()==0.

			// It sucks my 3

			// hours.
			continue;
		}
		String[] tt = s.trim().split("(?<=[?à¥¤]+)");
		for (int i = 0; i < tt.length; i++) {
			if (tt[i].length() != 0 || !tt[i].trim().equals("")
					|| !tt[i].trim().isEmpty()) {
				temp.add(tt[i].trim());
			}
		}
	}
	return temp;
}


public List<String> listSentencise1(List<String> input) {
	List<String> temp = new ArrayList<String>();

	for (String s : input) {
		if (s == null || s.length() == 1 || s.isEmpty()
				|| s.trim().equals("") || s.equals("")) { // Ashok: I don't

			// know why

			// s.length()==1

			// works but not

			// s.length()==0.
```

```
                        // It sucks my 3

                        // hours.
                        continue;
                }
                String[] tt = s.trim().split("[à¥¤?]+");

                for (int i = 0; i < tt.length; i++) {
                        if (tt[i].length() != 0 || !tt[i].trim().equals("")
                                        || !tt[i].trim().isEmpty()) {
                                temp.add(tt[i].trim());
                        }
                }
        }
        return temp;
}


public List<String> listSentencise(String input) {
        List<String> temp = new ArrayList<String>();

        if (input.length() != 0 || input.trim().equals("")) {
                String[] tt = input.trim().split("(?<=[?à¥¤]+)");
                for (int i = 0; i < tt.length; i++) {
                        if (tt[i].length() != 0 || !tt[i].trim().equals("")
                                        || !tt[i].trim().isEmpty()) {
                                temp.add(tt[i].trim());
                        }
                }
        }
        return temp;
}


public List<String> listSentencise1(String input) {
        List<String> temp = new ArrayList<String>();

        if (input.length() != 0 || input.trim().equals("")) {
                String[] tt = input.trim().split("[?à¥¤]+");
                for (int i = 0; i < tt.length; i++) {
                        if (tt[i].length() != 0 || !tt[i].trim().equals("")
                                        || !tt[i].trim().isEmpty()) {
                                temp.add(tt[i].trim());
                        }
                }
        }
```

```
		return temp;
	}


	public double df(List<List<String>> input, String token) {
		// The document frequency of token: the number of documents that contain
		// token:
		// input: class 0: document 0,1,2,...
		// class 1: document 0,1,2,...
		// class 2: document 0,1,2,...
		//
		double df = 1.0; //no avoid divide by zero error
		int nClasses = input.size();
		for (int iClass = 0; iClass < nClasses; iClass++) {
			int nSamples = input.get(iClass).size();
			for (int iSample = 0; iSample < nSamples; iSample++) {
				String doc = input.get(iClass).get(iSample);
				if (!doc.isEmpty()) {
					Boolean found = Arrays.asList(doc.split(" ")).contains(
							token);
					if (found) {
						df = df + 1;
						// System.out.println("Keyword matched the
string. Class:"+iClass+"Document: "+iSample+"df:"+df);

					}
				}
			}
		}

		return df;
	}

	public double idf(List<List<String>> input, String token) {
		int N = 0; // Total number of documents in the collection
		for (int iClass = 0; iClass < input.size(); iClass++)
			for (int iSample = 0; iSample < input.get(iClass).size(); iSample++)
				N = N + 1;

		double df_t = df(input, token);
		double idf = Math.log10(N / df_t);

		return idf;
	}
```

```
public double tf(Map<String,Integer> docFreqMap, String token){
	int tf_td=0;
	if(docFreqMap.containsKey(token)){
		tf_td=docFreqMap.get(token);
	}

	return (1+ Math.log10(tf_td));
}

public Map<String,Integer> wordFrequencyCount(String doc){
	Map<String,Integer> hmap= new HashMap<String,Integer>();
	for(String tempStr : doc.split("[ ]+"))
	{
	 if(hmap.containsKey(tempStr))
	 {
	   Integer i=hmap.get(tempStr);
	   i+=1;
	   hmap.put(tempStr,i);
	 }
	 else
	   hmap.put(tempStr,1);
	}
return hmap;
}

public double tfidf(double tf,double idf){
	return tf*idf;
}

	public double[][] createFeatureVector(HashSet<String> dict,
			List<List<String>> input) {

		// Total number of documents
		int nDocs = 0;
		for (int iClass = 0; iClass < input.size(); iClass++)
			for (int iSample = 0; iSample < input.get(iClass).size(); iSample++)
				nDocs = nDocs + 1;

		// Total number of features
		int nFeatures = 1 + dict.size(); // 1 for class index

		Object[] dictionary = dict.toArray();

		double[][] featureVector = new double[nDocs][nFeatures];
		for (int i = 0; i < nDocs; i++) {
			for (int j = 0; j < nFeatures; j++) {
```

```
                        featureVector[i][j]=0.0;
                    }
                }

                int iDoc = 0;
                for (int iClass = 0; iClass < input.size(); iClass++) {
                    for (int iSample = 0; iSample < input.get(iClass).size(); iSample++) {
                        Map<String, Integer> docFreqMap =
wordFrequencyCount(input.get(
                                iClass).get(iSample));
                        featureVector[iDoc][0] = iClass; //
featureVector[docId][0]=classId

                        for (Entry<String, Integer> entry : docFreqMap.entrySet()) {
                            String iWord = entry.getKey();
                            Integer value = entry.getValue();

                            int iWordIndex =
Arrays.asList(dictionary).indexOf(iWord);
//                              System.out.println(iClass+" "+iSample+ "Key = " +
iWord + " Value = " + value);
                            if (iWordIndex != -1) { //If word is not in dictionary,
leave it.
//                                  System.out.println("Dictionary Index:
"+iWordIndex+" value: "+dictionary[iWordIndex]);
//                                  System.out.println(iWordIndex+1);

                                //Calculate tf-idf of the word
                                double tf=tf(docFreqMap,iWord);
                                double idf= idf(input, iWord);
                                double tfidf=tfidf(tf, idf);
                                featureVector[iDoc][iWordIndex+1] = tfidf;
                            }
//                          System.out.println();
                        }
                        iDoc++;
                    }
                }

                /*System.out.print(" ");
                for (int i = 0; i < dictionary.length; i++)
                    System.out.print(" " + dictionary[i].toString());
                System.out.println();

                for (int i = 0; i < nDocs; i++) {
                    for (int j = 0; j < nFeatures; j++) {
```

```
                        if(j!=0 && featureVector[i][j]==0.0){
                                System.out.print(" -" );
                        }else
                        System.out.print(" " + featureVector[i][j]);
                }
                System.out.println();
        }*/
        return featureVector;
    }

}
```

Naïve Bayes

```
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.bayes.NaiveBayes;
import weka.core.Attribute;
import weka.core.FastVector;
import weka.core.Instance;
import weka.core.Instances;
public class NaiveBayesLearning {
public Classifier NaiveBayes(int numClasses, int attribSize, double[][] featureVector)
                throws Exception {
        // Declare two numeric attributes
Attribute[] fvAttribute = new Attribute[attribSize];
        for (int i = 0; i < attribSize; i++) {
                String attribname = "" + i;
                fvAttribute[i] = new Attribute(attribname);

        }
        // Declare the class attribute along with its values
        FastVector fvClassVal = new FastVector(numClasses); // Classes
        for (int i = 0; i < numClasses; i++) {
                String className = "" + i;
                fvClassVal.addElement(className);
```

```
}

Attribute ClassAttribute = new Attribute("classes", fvClassVal);
FastVector fvWekaAttributes = new FastVector(attribSize + 1);
for (int i = 0; i < attribSize; i++) {
        fvWekaAttributes.addElement(fvAttribute[i]);

}
fvWekaAttributes.addElement(ClassAttribute);

// Create an empty training set
Instances trainingSet = new Instances("Rel", fvWekaAttributes,
                featureVector.length);

// Set class index
trainingSet.setClassIndex(attribSize);
// Fill the training set
Instance iSample = new Instance(attribSize + 1);

for (int i = 0; i < featureVector.length; i++) {
        int k = 0;
        for (int j = 1; j < featureVector[i].length; j++) {
                iSample.setValue((Attribute) fvWekaAttributes.elementAt(k),
                                featureVector[i][j]);
                k++;
        }
        iSample.setValue(
                        (Attribute) fvWekaAttributes.elementAt(attribSize),
                        Integer.toString((int) featureVector[i][0]));
        trainingSet.add(iSample);
```

```
                }
                Classifier cModel = (Classifier) new NaiveBayes();
                cModel.buildClassifier(trainingSet);

                return cModel;
        }

public double[][] testNaive(Classifier cModel,  int numClasses, int attribSize, double[][]
featureVector)
                        throws Exception {
                // Declare two numeric attributes
int numInstances=featureVector.length;

                Attribute[] fvAttribute = new Attribute[attribSize];
                for (int i = 0; i < attribSize; i++) {
                        String attribname = "" + i;
                        fvAttribute[i] = new Attribute(attribname);

                }
                // Declare the class attribute along with its values
                FastVector fvClassVal = new FastVector(numClasses); // Classes
                for (int i = 0; i < numClasses; i++) {
                        String className = "" + i;
                        fvClassVal.addElement(className);
                }
                Attribute ClassAttribute = new Attribute("classes", fvClassVal);
                // Declare the feature vector
                FastVector fvWekaAttributes = new FastVector(attribSize + 1);
                for (int i = 0; i < attribSize; i++) {
                        fvWekaAttributes.addElement(fvAttribute[i]);
```

```
        }
        fvWekaAttributes.addElement(ClassAttribute);

        // Create an empty training set
        Instances testingSet = new Instances("Rel", fvWekaAttributes,
                        featureVector.length);

        // Set class index
        testingSet.setClassIndex(attribSize);
//      System.out.println("Training set Size and class index set");
        // Fill the training set
        Instance iSample = new Instance(attribSize + 1);

        for (int i = 0; i < featureVector.length; i++) {
                int k = 0;
                for (int j = 1; j < featureVector[i].length; j++) {
                        iSample.setValue((Attribute) fvWekaAttributes.elementAt(k),
                                        featureVector[i][j]);
                        k++;
                }
                iSample.setValue(
                                (Attribute) fvWekaAttributes.elementAt(attribSize),
                                Integer.toString((int) featureVector[i][0]));
                testingSet.add(iSample);

        }
        // Test the model
        Evaluation eTest = new Evaluation(testingSet);
        eTest.evaluateModel(cModel, testingSet);
```

```
double[][] output = new double[numInstances][numClasses];
for (int i = 0; i < numInstances; i++)
        for (int j = 0; j < numClasses; j++)
                output[i][j]=0.0;

for (int i = 0; i < numInstances; i++) {
        double y = eTest.evaluateModelOnce(cModel, testingSet.instance(i));
        for (int k = 0; k < numClasses; k++) {
                if ((int) y == k) {
                        output[i][k] = 1;
                        break;
                }
        }

}

/*for (int i = 0; i < numInstances; i++){
        for (int j = 0; j < numClasses; j++){
                System.out.print(" "+output[i][j]);
        }
System.out.println();
}*/

// Print the result Ã  la Weka explorer:
/*String strSummary = eTest.toSummaryString();
System.out.println(strSummary);
System.out.println("class detail: "+eTest.toClassDetailsString());
// Get the confusion matrix
double[][] cmMatrix = eTest.confusionMatrix();
for (int row_i = 0; row_i < cmMatrix.length; row_i++) {
        for (int col_i = 0; col_i < cmMatrix.length; col_i++) {
```

```
                    System.out.print(cmMatrix[row_i][col_i]);
                    System.out.print("|");
                }
                System.out.println();
        }*/
        return output;
}


public void naiveDemo()
                throws Exception {
        // Declare two numeric attributes
        Attribute Attribute1 = new Attribute("firstNumeric");
        Attribute Attribute2 = new Attribute("secondNumeric");

        // Declare a nominal attribute along with its values
        FastVector fvNominalVal = new FastVector(3);
        fvNominalVal.addElement("blue");
        fvNominalVal.addElement("gray");
        fvNominalVal.addElement("black");
        Attribute Attribute3 = new Attribute("aNominal", fvNominalVal);

        // Declare the class attribute along with its values
        FastVector fvClassVal = new FastVector(2);
        fvClassVal.addElement("positive");
        fvClassVal.addElement("negative");
        Attribute ClassAttribute = new Attribute("theClass", fvClassVal);

        // Declare the feature vector
        FastVector fvWekaAttributes = new FastVector(4);
        fvWekaAttributes.addElement(Attribute1);
```

```
fvWekaAttributes.addElement(Attribute2);
fvWekaAttributes.addElement(Attribute3);
fvWekaAttributes.addElement(ClassAttribute);

// Create an empty training set
Instances isTrainingSet = new Instances("Rel", fvWekaAttributes, 10);

// Set class index
isTrainingSet.setClassIndex(3);

// Create the instance
Instance iExample = new Instance(4);
iExample.setValue((Attribute) fvWekaAttributes.elementAt(0), 1.0);
iExample.setValue((Attribute) fvWekaAttributes.elementAt(1), 0.5);
iExample.setValue((Attribute) fvWekaAttributes.elementAt(2), "gray");
iExample.setValue((Attribute) fvWekaAttributes.elementAt(3), "positive");

// add the instance
isTrainingSet.add(iExample);
Classifier cModel = (Classifier) new NaiveBayes();
cModel.buildClassifier(isTrainingSet);

// Test the model
Evaluation eTest = new Evaluation(isTrainingSet);
eTest.evaluateModel(cModel, isTrainingSet);

// Print the result Ã  la Weka explorer:
String strSummary = eTest.toSummaryString();
System.out.println(strSummary);

// Get the confusion matrix
```

```
                double[][] cmMatrix = eTest.confusionMatrix();
                for (int row_i = 0; row_i < cmMatrix.length; row_i++) {
                        for (int col_i = 0; col_i < cmMatrix.length; col_i++) {
                                System.out.print(cmMatrix[row_i][col_i]);
                                System.out.print("|");
                        }
                        System.out.println();
                }
        }
}
```

Evaluation

```
import java.text.DecimalFormat;
import java.util.*;
public class Evaluation{
        FeatureExtractionDaoImpl feDao = new FeatureExtractionDaoImpl();
        int nFeatures = new Integer(9);
        public List<String> summaryGeneration(List<String> input,
                        List<String> ppOutput, List<List<String>> neDict, float nSummary) {
                // input-sentence list
                List<String> summary = new ArrayList<String>();
                double[][] scoreArray = sentenceScoring(ppOutput, neDict);
                double[][] rankArray = sentenceRanking(scoreArray);
                int nRows = rankArray.length;
                // int nSummarySentences = (int)Math.ceil(nRows * nSummary);
                int nSummarySentences = (int) Math.round(nRows * nSummary);
                int[] ssIndexArry = new int[nSummarySentences];
                for (int i = 0; i < nSummarySentences; i++) {
                        ssIndexArry[i] = (int) (rankArray[i][0]);
```

```
            }

            // Order the generated summary in original document order for
            // readability
            Arrays.sort(ssIndexArry);

            // Retrieve summary sentences
            for (int i = 0; i < nSummarySentences; i++) {
                  summary.add(input.get(ssIndexArry[i]));
                  // System.out.println(input.get(ssIndexArry[i]));
            }
            return summary;
      }

      public double[][] sentenceRanking(double[][] scoreArray) {

            Arrays.sort(scoreArray, new Comparator<double[]>() {
                  @Override
                  public int compare(double[] int1, double[] int2) {
                        Double numOfKeys1 = int1[1];
                        Double numOfKeys2 = int2[1];
                        return -numOfKeys1.compareTo(numOfKeys2); // to return result
in

            // decreasing order
                  }
            });
            return scoreArray;
      }

      public double[][] sentenceScoring(List<String> input,
```

```
        List<List<String>> neDict) {
double[][] featureArray = sentenceFeatures(input, neDict);
int nSentences = input.size();
double[][] scoreArray = new double[nSentences][2];

for (int i = 0; i < nSentences; i++) {
        scoreArray[i][0] = i;
        scoreArray[i][1] = featureArray[i][1] + featureArray[i][2]
                        + featureArray[i][3] + featureArray[i][4]
                        + featureArray[i][5] + featureArray[i][6]
                        + featureArray[i][7] + featureArray[i][8];
}
System.out.println("-----------------------------------------------");
System.out.println("Features of Sentences");
System.out.println("-----------------------------------------------");
for (int i = 0; i < nSentences; i++) {
        for (int j = 0; j < 9; j++) {
                System.out.print(featureArray[i][j] + " ");
        }
        System.out.println();

}
System.out.println("-----------------------------------------------");
System.out.println("Score of Sentences");
System.out.println("-----------------------------------------------");
for (int i = 0; i < nSentences; i++) {
        for (int j = 0; j < 2; j++) {
                System.out.print(scoreArray[i][j] + " ");
        }
        System.out.println();
```

```
                }
                return scoreArray;
        }
public double[][] sentenceFeatures(List<String> input,
                        List<List<String>> neDict) {
                // input-sentence list
                int nWordsInLongestSentence = feDao.noOfWordsInLongestSentence(input);
                int nSentences = input.size();
                double[][] featureArray = new double[nSentences][nFeatures];
                String sentence;
                List<String> tokenArray = new ArrayList<String>();

                for (int i = 0; i < nSentences; i++) {

                        if (input.get(i).isEmpty() || input.get(i).trim().isEmpty()
                                        || input.get(i) == null) {
                                featureArray[i][0] = i;
                                featureArray[i][1] = 0;
                                featureArray[i][2] = 0;
                                featureArray[i][3] = 0;
                                featureArray[i][4] = 0;
                                featureArray[i][5] = 0;
                                featureArray[i][6] = 0;
                                featureArray[i][7] = 0;
                                featureArray[i][8] = 0;
                        } else {
                                sentence = input.get(i).toString().trim();
                                // Sentence Index
                                featureArray[i][0] = i;

                                // Length Feature
```

```
                featureArray[i][1] = feDao.lengthOfSentence(
                        nWordsInLongestSentence, sentence);
                // Postion Feature
                featureArray[i][2] = feDao.sentencePosition(input, i + 1,
                        sentence);

                // NE Features
                tokenArray = feDao.sentenceTokanize(sentence);
                int nTokens = tokenArray.size();
                int neResult;
                String word;
                int nPersons = 0, nLocations = 0, nOrganizations = 0, nMisc = 0,
nNotNE = 0;
                for (int j = 0; j < nTokens; j++) {
                        word = tokenArray.get(j);
                        neResult = feDao.neClass(neDict, word);
                        switch (neResult) {
                        case 1:
                                nPersons = nPersons + 1;
                                break;
                        case 2:
                                nLocations = nLocations + 1;
                                break;
                        case 3:
                                nOrganizations = nOrganizations + 1;
                                break;
                        case 4:
                                nMisc = nMisc + 1;
                                break;
                        case 5:
                                nNotNE = nNotNE + 1;
```

```
                                        break;
                                    default:
                                        break;
                                    }

                                    featureArray[i][3] = nPersons;
                                    featureArray[i][4] = nLocations;
                                    featureArray[i][5] = nOrganizations;
                                    featureArray[i][6] = nMisc;
                                    featureArray[i][7] = 0.0; // featureArray[i][7] = nNotNE;
                                                                                    // //Not
NE feature bias the
                                                                                    //
result.(to be improved)
                                }
                                // Term Frequency Features
                                featureArray[i][8] = feDao.sentenceWeight1(input, sentence);
                        }
                }
                return featureArray;
        }

        public void systemEvaluation(List<String> manualSummary,
                        List<String> systemSummary) {
                double precision = precision(manualSummary, systemSummary);
                double recall = recall(manualSummary, systemSummary);
                double beta = 1;
                double fscore = fscore(precision, recall, beta);

                System.out.println("---------------------------------------------");
                System.out.println("Results");
```

```
        System.out.println("\tPrecision: " + roundTwoDecimals(precision * 100)
                        + "%");
        System.out.println("\tRecall: " + roundTwoDecimals(recall * 100) + "%");

}

public String[] systemEvaluation1(List<String> manualSummary,
                List<String> systemSummary) {
        double precision = precision(manualSummary, systemSummary);
        double recall = recall(manualSummary, systemSummary);
        double beta = 1;
        String[] result = new String[3];
        result[0] = roundTwoDecimals(precision * 100) + "%".toString();
        result[1] = roundTwoDecimals(recall * 100) + "%".toString();
        result[2] = roundTwoDecimals(fscore * 100) + "%".toString();

        return result;
}

double roundTwoDecimals(double d) {
        DecimalFormat twoDForm = new DecimalFormat("#.##");
        return Double.valueOf(twoDForm.format(d));
}

public double precision(List<String> manualSummary,
                List<String> systemSummary) {
        int nSystemSumarySentences = systemSummary.size();
        Set<String> intersect = new HashSet<String>(manualSummary);
        intersect.retainAll(systemSummary);
        int nSystemCorrectSummarySentences = intersect.size();
```

```
                // System.out.println("PnSystemSumarySentences: "+nSystemSumarySentences);
                // System.out.println("PnSystemCorrectSummarySentences:
"+nSystemCorrectSummarySentences);

                return (double) nSystemCorrectSummarySentences / nSystemSumarySentences;
        }

        public double recall(List<String> manualSummary, List<String> systemSummary) {
                int nManualSumarySentences = manualSummary.size();
                Set<String> intersect = new HashSet<String>(manualSummary);
                intersect.retainAll(systemSummary);
                int nSystemCorrectSummarySentences = intersect.size();

                // System.out.println("RnManualSumarySentences:
"+nManualSumarySentences);
                // System.out.println("RnSystemCorrectSummarySentences:
"+nSystemCorrectSummarySentences);

                return (double) nSystemCorrectSummarySentences / nManualSumarySentences;
        }

        public double fscore(double precision, double recall, double beta) {
                double f = 0.0;
                if (precision != 0.0 && recall != 0.0) {
                        f = ((beta * beta + 1) * precision * recall)
                                        / ((beta * beta) * precision + recall);
                }
                return f;
        }

        public class values {
```

```
        int val1;
        int val2;

        values(int v1, int v2) {
                this.val1 = v1;
                this.val2 = v2;
        }

        public void Update_VAl(int v1, int v2) {
                this.val1 = v1;
                this.val2 = v2;
        }
}// end of class values

public double cosineSimilarityScore(String Text1, String Text2) {

        double sim_score = 0.0000000;
        // 1. Identify distinct words from both documents
        String[] word_seq_text1 = Text1.split(" ");
        String[] word_seq_text2 = Text2.split(" ");
        Hashtable<String, values> word_freq_vector = new Hashtable<String, values>();
        LinkedList<String> Distinct_words_text_1_2 = new LinkedList<String>();

        // prepare word frequency vector by using Text1
        for (int i = 0; i < word_seq_text1.length; i++) {
                String tmp_wd = word_seq_text1[i].trim();
                if (tmp_wd.length() > 0) {
                        if (word_freq_vector.containsKey(tmp_wd)) {
                                values vals1 = word_freq_vector.get(tmp_wd);
                                int freq1 = vals1.val1 + 1;
                                int freq2 = vals1.val2;
```

```
                        vals1.Update_VAl(freq1, freq2);
                        word_freq_vector.put(tmp_wd, vals1);
                } else {
                        values vals1 = new values(1, 0);
                        word_freq_vector.put(tmp_wd, vals1);
                        Distinct_words_text_1_2.add(tmp_wd);
                }
        }
}

// prepare word frequency vector by using Text2
for (int i = 0; i < word_seq_text2.length; i++) {
        String tmp_wd = word_seq_text2[i].trim();
        if (tmp_wd.length() > 0) {
                if (word_freq_vector.containsKey(tmp_wd)) {
                        values vals1 = word_freq_vector.get(tmp_wd);
                        int freq1 = vals1.val1;
                        int freq2 = vals1.val2 + 1;
                        vals1.Update_VAl(freq1, freq2);
                        word_freq_vector.put(tmp_wd, vals1);
                } else {
                        values vals1 = new values(0, 1);
                        word_freq_vector.put(tmp_wd, vals1);
                        Distinct_words_text_1_2.add(tmp_wd);
                }
        }
}

// calculate the cosine similarity score.
double VectAB = 0.0000000;
double VectA_Sq = 0.0000000;
```

```
                double VectB_Sq = 0.0000000;

                for (int i = 0; i < Distinct_words_text_1_2.size(); i++) {
                        values vals12 = word_freq_vector
                                        .get(Distinct_words_text_1_2.get(i));

                        double freq1 = (double) vals12.val1;
                        double freq2 = (double) vals12.val2;
                        VectAB = VectAB + (freq1 * freq2);
                        VectA_Sq = VectA_Sq + freq1 * freq1;
                        VectB_Sq = VectB_Sq + freq2 * freq2;
}sim_score = ((VectAB) / (Math.sqrt(VectA_Sq) * Math.sqrt(VectB_Sq)));

                // return(sim_score);
                return (roundTwoDecimals(sim_score * 100));
        }
}
```