Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Kirtipur, Kathmandu

In partial fulfillment of the requirement for Master's Degree in Computer Science and Information Technology (M.Sc. CSIT), Second Semester

Seminar Report on:

"Comparative Analysis of Word Embedding Models for Nepali Text Classification"

Submitted To:

Central Department of Computer Science and Information Technology

Submitted By:

Sudhan Kandel (623/077)

August, 2022

Tribhuvan University

Institute of Science and Technology

Supervisor's Recommendation

I hereby recommend that this Seminar report, prepared under my supervision by **Mr. Sudhan Kandel** entitled **"Performance Analysis of Word Embedding Models for Nepali Text Classification"** be accepted as fulfillment in partial requirement for the degree of Master's of Science in Computer Science and Information Technology. To the best of my knowledge, this is an original work in computer science.

Asst. Prof. Bikash Balami

Central Department of Computer Science and Information Technology

Tribhuvan University

Institute of Science and Technology

LETTER OF APPROVAL

This is to certify that the seminar report prepared by **Mr. Sudhan Kandel** entitled **"Performance Analysis of Word Embedding Models for Nepali Text Classification"** in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

Asst. Prof. Sarbin Sayami
(HOD)
Central Department of Computer Science and Information Technology

Asst. Prof. Bikash Balami
(Supervisor)
Central Department of Computer Science and Information Technology

(Internal)

ACKNOWLEDGEMENT

The success and final outcome of this seminar report required a lot of guidance and assistance from many people and I am very fortunate to have got this all along the completion. I am very glad to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed supervisor **Asst. Prof. Bikash Balami** Central Department of computer science and Information Technology for his valuable supervision, guidance, encouragement, and support for completing this seminar report. I am also thankful to **Asst. Prof. Sarbin Sayami**, HOD of Central Department of Computer Science and Information Technology for his constant support throughout the period. Furthermore, with immense pleasure, I submit by deepest gratitude to the Central Department of Computer Science and Information Technology, Tribhuvan University, and all the faculty members of CDCSIT for providing the platform to explore the knowledge of interest. At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly.

Sudhan Kandel (623/077)

ABSTRACT

The task of classifying unstructured document to the proper category to which it belongs to is become difficult task because the volume of information shared over the internet increase swiftly. Text classification is the task of allocating the document into one or more number of predefined categories. Additionally, this is very famous in the field of information retrieval, text summarization and so on. However, all the application areas transformation of text into feature vectors play the important role.Previously several text representation techniques have been proposed by different authors to capture the real semantics of the web documents, but this has become hinderance because of semantic mismatch and multiple meanings of words. Thus, this seminar report proposes a word embedding to solve the document representation problem in Nepali news classification. To achieve this task, this seminar work utilizes Word2Vec and ELMo word embedding algorithms to represent documents. And which are also used in conjunction with Nepali news classification algorithm to determine the most effective embedding model. In this report two machine learning algorithm namely SVM and Random Forest are used.

Context dependent and independent word representation model are used in this seminar report. Here total of 5145 Nepali news is scraped from e-kantipur for analyzing these models with text classification model, to which it includes four different categories. All in all, the Random Forest machine learning algorithm provide the highest accuracy of 74 percent with the ELMo word representation. However, scenario with SVM looks different, it gives the highest accuracy of 68 percent with Word2Vec word vectorization model.

Keywords: *Word2Vec, ELMo, Word-Embedding*

Contents

List of Figures

List of Tables

List of Abbreviations

- **HTML: Hyper Text Markup Language**
- **LSTM: Long Short Term Memory**
- **NLP: Natural Language Processing**
- **NLTK: Natural Language Processing**
- **PCA: Principal Component Analysis**
- **SBD: Sentence Boundary Disambiguation**
- **SOTA: State of The Art**
- **SVM: Support Vector Machine**

Chapter 1

INTRODUCTION

1.1 Overview

Text classification is a construction problem of models which can classify new documents into pre-defined classes. It involves assigning documents into their predefined categories based on their contents. In many real-world scenarios, the ability to automatically classify documents into a fixed set of categories is highly desirable. This allows users to find desired information faster by searching only the relevant categories and not the entire information space. The importance of text classification is even more apparent when the information space is huge such as the World Wide Web [1].
Word embedding is a way to represent a word with fixed-length vector of continuous real numbers. It maps a word in a vocabulary to a latent vector space where words with similar context are in proximity. Through word embedding, a word is converted to a vector that summaries both the word's syntactic and semantic information. Consequently, word embedding is particularly suitable to be used as feature representations in neural network model for downstream natural language processing task, such as text classification, machine translation, sequence learning [2].
Finding such representation for words and sentences has been one hot topic over the last few years in the field of Natural Language Processing and has led to many improvements in core NLP task. This seminar report seeks to compare the effect of different word representation for the Nepali text classification. Here, Word2vec and ELMO word embedding techniques are used for the analysis purpose these are the two category of word representation technique namely context dependent and context independent show in the figure 1.1. All in all, this

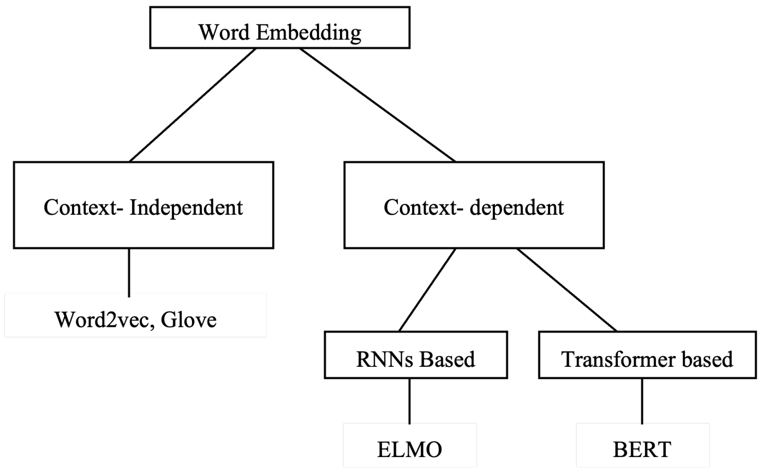


Figure 1.1: A taxonomy of word embedding

transformation has two important beneficial properties that is dimensionality reduction for efficient representation and contextual similarity for expressive representations.
The remaining part of this seminar report is organized as follow. Literature reviews to this study are described in Chapter 2. In Chapter 3, have described the proposed methodology. The implementation part is described in Chapter 4. Finally, result and conclusion is included in the chapter 5 and 6 respectively.

1.1.1 Problem Statement

The huge unstructured data in the digital world is one of the reasons where the application of various machine learning techniques is possible. However, the complex morphology of text due to various forms of existing consonants and vowel letters make it difficult in feature extraction from the document. Since the different languages are morphologically rich and complex, natural language processing task such as classification and machine translation has become complex because of inappropriate word embedding techniques.

1.2 Objective

- To implement and analysis context dependent and independent word embedding model.
- To implement SVM and Random Forest Machine learning algorithms for text classification.

Chapter 2

LITERATURE REVIEW

Several machine learning algorithms have been developed for performing the natural language processing task, where word embedding techniques play the significant role. Additionally, number of researches have been carried out in this topic, some of these are explained in the following section.

In paper [2], the author purposed a comparative analysis of different word embedding models namely continuous bag of words, skip gram, Glove (Global Vectors for word representation) and Hellinger- PCA. Additionally, models were compared on different parameters such as performance with respect to size of training data, basic over-view, and relation of context and target words, memory consumption etc. All in all, the author concluded that Glove is the best model as compared to other models as it is scalable to large corpus and worked well even with small corpus.

NPVec1, which consist of 25 SATA word embedding for Nepali language that derived from a large corpus using Glove, Word2Vec, fastText, and BERT is discussed in the paper [4]. It also provided intrinsic and extrinstic evaluation of these embedding using well established metrices and methods. These models were trained using 279 million word tokens and were the largest embedding ever trained for Nepali language. Macro Precision, Recall and F1 metrics were used for the evaluation of the classification model. On average, the F1 scores for word embedding mod- els exceeded the baseline scores by a margin of 5 percent.

The author of the paper [1], conducted experiments on both classic and contextu-alised word embedding for the purpose of text classification. Moreover, to study the impact of word embedding on different dataset, the author selected four benchmarking classification datasets with varying average sample length for comparing both single-label and multi-label classification tasks. This study recommended that choosing CNN over BiLSTM for document classification dataset, where the context in sequence was not as indicative of class membership as sentence datasets. All in all, the author concluded that contextualized embedding was matched well with BiLSTM for SST-2-like classification datasets, while CNN collaborates well with classic embedding for document dataset like 20NewsGroup.

In the paper [3], the aim of the task was to evaluate system for early risk prediction on the internet, in particular to identify users suffering from eating disorders. In the controlled setting, the author evaluated the performance of three different word representation methods: random indexing, GloVe, and ELMo. The best model in term of F1 score turned out to be a model with GloVe vector as input to the text classifier and multi-layer perceptron.

In paper [6], discussed the performance of three word embedding models namely, word2vec in tensorflow, word2vec from Gensim package and FastText model. Additionally, dataset used in this research work contained 521391 unique words to produce the clusters and evaluated their performance in terms of accuracy and efficiency.

Chapter 3

METHODOLOGY

3.1 Methodology

This seminar report is more focus on feature extraction of Nepali news dataset for the text classification task. Additionally, web scraping process is used for the data collection. The figure 3.1 illustrate the step carry out in this seminar report.

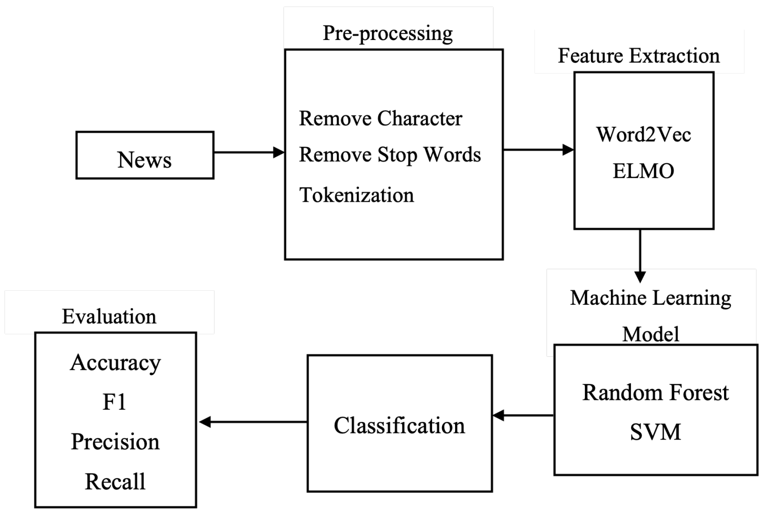


Figure 3.1: Overall Process

3.2 Dataset Discription

There are total of 5145 news articles which are scraped from www.ekantipur.com, in four different categories. Categories are Health, National, Opinion, Business. The detail of the dataset is given in the table 3.1.

Table 3.1: Dataset Description

Category	Total Data
Health	1029
Opinion	1029
National	1029
News	1029

3.3 Data Preprocessing

This step cleans the document by removing HTML tags and noisy characters present in the document, by using different python libraries such as NLTK and regular expression.

3.3.1 Remove Special Characters

Special characters play the negative role in the machine learning algorithm. So, they are removed by using the regular expression library available in the python programming language.

3.3.2 Remove Stop Words

A stop words is a commonly used word that a search engine has been programmed to ignore, both when indexing entries for searching and retrieving them as the result of a search query. Additionally, these data may not play the significant role for the result, so here remove them easily by storing a list of words by considering to be stop words. NLTK (Natural Language Toolkit) in python has a list of stop words stored in 16 different languages.

3.3.3 Tokenization

Tokenization describes splitting paragraphs into sentences, or sentences into individual words. For the former Sentence Boundary Disambiguation (SBD) can be applied to create a list of individual sentences. This relies on a pre-trained, language specific algorithms like the Punkt Models from NLTK. Sentences can be split into individual words and punctuation through a similar process. This seminar report uses tokenization in preprocessing task by using Punkt Models from NLTK.

3.4 Feature Extraction

3.4.1 Word2Vec

Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus. It was developed by Tomas Mikolov [5] as a response to make the neural-network-based training of the embedding more efficient then has become the de facto standard for developing pre-trained word embedding. Word2Vec is a method to construct such embedding. It can be obtained using two methods: Skip Gram and CBOW. The figure 3.2 show the architecture of CBOW

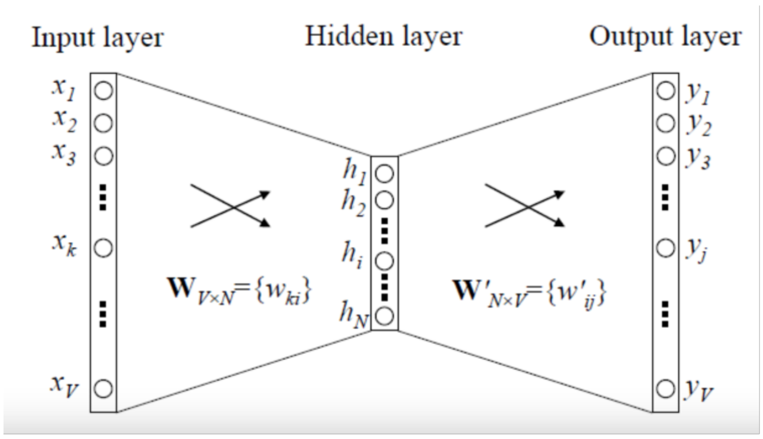


Figure 3.2: The CBOW architecture predicts the current word

Word representations in Word2Vec are taken from a simple neural network, which consists of:

- An input layer
- A projection layer
- One hidden layer
- One output layer

The input is the one hot encoded representation of a word, and the projection layer has dimensionality N*D, while the hidden layer is typically 300 units and learns dense representation. Additionally, the hidden layer has no activation function.

3.4.2 ELMo

The ELMo is a novel way to represent words in vectors or embedding. It was developed by Sarzynska-Wawer, [7], and these word embeddings are helpful in achieving state-of-the-art (SOTA) results in several NLP tasks such as text classification, machine translation and sentiment analysis. Unlike most widely used word embedding, ELMo word representations are functions of the entire input sentence. They are computed on top of the two-layer biLMs with character convolution, as a liner function of the internal network state. This setup allows to do semi-supervised learning, where the biLM is pretrained at a large scale and easily incorporated into a wide range of existing NLP architectures. The ELMo architecture is shown in the figure 3.3.

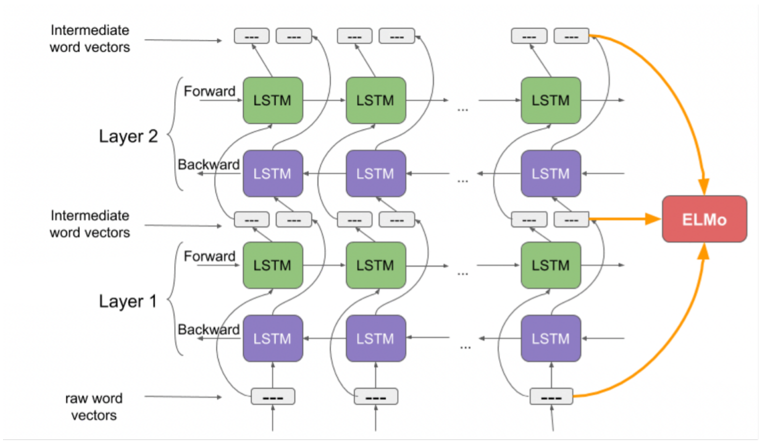


Figure 3.3: Architecture of ELMo

3.5 Machine Learning Model

3.5.1 SVM

The SVM algorithm is a 'simple' linear classification/regression algorithm. It tries to find a hyper plane which separates the data in two classes as optimally as possible. Here as optimally as possible means that as many points as possible of label A should be separated to one side of the hyper plane and as points of label B to the other side, while maximizing the distance of each point to this hyper plane. The hyperplane used in the support vector machine is shown in the figure 3.4.

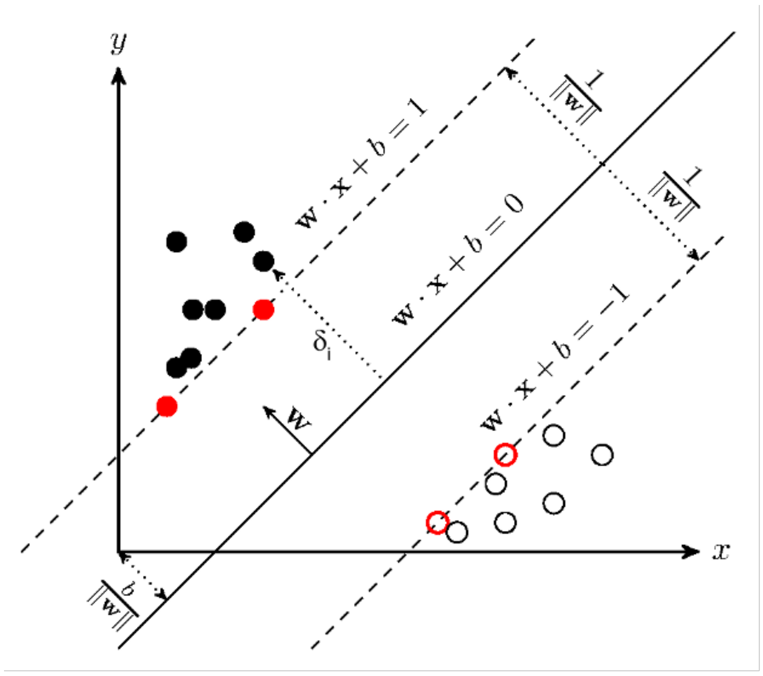


Figure 3.4: Support Vector machine hyper plane

Steps involved in SVM algorithm:

Step 1. Read the training data set.

Step 2. Prepared the pattern matrix.

Step 3. Select kernel function to use.

Step 4. Select parameter of the kernel function and value of c

Step 5. Execute the training algorithm

3.5.2 Random Forest

Random forest is a supervised machine learning algorithm that is used widely in classification and regression problems. It builds decision tree on different samples and takes their majority vote for classification and average in case of regression. Additionally, one of the most important features of the random forest algorithm is that it can handle the data set containing continuous variable as in the case of regression and categorical variables as in the case of classification.

Random forest algorithm is worked on the bagging principal. It is also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses random sample from the dataset. Hence each model is generated from the samples provided by the original data with replacement known as row sampling. Additionally, each model is trained independently which generate results. The final output is based on majority voting after combining the results of all models. All in all, the step which involves combining all the results and generating output based on majority voting is known as aggregation. The overall structure of the random forest algorithm is depicted in the figure 3.5.

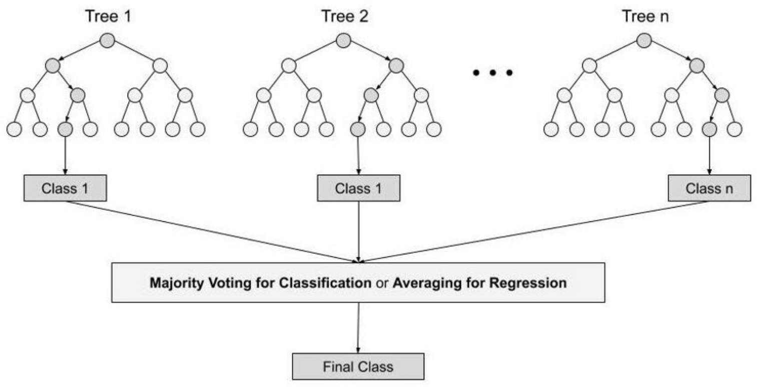

Tree 1
Tree 2
Tree n
Class 1
Class 1
Class n
Majority Voting for Classification or Averaging for Regression
Final Class

Figure 3.5: Random Forest Classifier

Steps involved in random forest algorithm:

Step1: n number of random news are taken from the given dataset.

Step2: individual decision trees are constructed for each taken sample news.

Step3: each decision tree will generate an output.

Step4: final output is considered based on majority voting or average for the news classification

3.6 Evaluation Techniques

3.6.1 Accuracy

In news classification, Accuracy Score is the ratio of correctly predict news classes to the total number of input data points. It can be calculated by using following formula.

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

Where,

TP is the number of True Positive instance

FN is the number of False Negative instance

FP is the number of False Positive instance

TN is the number of True Negative instance

3.6.2 Precision

Precision is the ratio of number of True Positive to the total number of Predicted Positive . It measures out of the total predicted positive, how many are actually positive. All in all, precision score is calculated by using following formula.

$$PrecisionScore = \frac{TP}{(FP + TP)}$$

Where,
TP is the number of true positive instance
FP is the number of false positive instance

3.6.3 Recall

Recall is the ratio of number of True Positive to the total number of Actual Positive. It measures out of the total actual positive, how many are predicted as True Positive, and it can be calculated by using following formula.

$$RecallScore = \frac{TP}{(FN + TP)}$$

Where,
TP is the number of true positive instance.
FN is the number of false-negative instance.

3.6.4 F1 Score

It is an important evaluation matric for binary classification that combines Precision and Recall. F1 score is the harmonic mean of precision and recall, whcih is obtained by the following calculation.

$$F1Score = 2 * \frac{PrecisionScore * RecallScore}{(PrecisionScore + RecallScore)}$$

Chapter 4

IMPLEMENTATION

The implementation include in this seminar report is carried out in the python programming language. Following python libraries are used for the implementation purpose.

- **Pandas:** pandas is one of the most important libraries of the python programming language, it is used to import the dataset and to obtain the statistical description of the dataset.
- **Matplotlib:** This library has been used for plotting graphs. The functions used from this library are as follows: plot(); plot various visualization, show(); to visualize the plots, line(); plot a line graph.
- **Sklearn:** This library has been used for splitting the train and test data.
- **TensorFlow:** It can be used across a range of task but has a particular focus on training and inference of deep neural network. In this seminar report we use 2.5.3 version of TensorFlow.
- **Keras:** It provides a python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

4.1 Data Collection

Dataset used in this seminar report is collected by performing web scraping process. For this, beautiful soup library is used which is freely available in python programming language. The code snippets for the data collection is depicted in the appendix [A].

4.2 Preprocessing

After the data collection process has been completed, the preprocessing task is inaugurated. This process plays a vital role in feature extraction as well as in the machine learning algorithm. It includes different activities such as removing secondary languages, special characters.

4.3 Feature Extraction

Processed data is not directly trained into the machine learning model. In this report, both context dependent and independent techniques are used for converting the text data into the numeric form. Before generating the vector of the data, a news corpus is generated by splitting total word present in the news to a single list and then implement the following techniques.

4.3.1 Word2Vec

The Word2Vec word embedding module is loaded by using genism library available in the python programming language. Additionally, this model only supports the tokenized words so, news is tokenized before passing into it. All in all, different hyperparameters such as size, window size, minimum counts etc., need to be pass for the training process. The code snippets for training this model is depicted in the appendix [B].

4.3.2 ELMo

The ELMO model is introduced by Google. It is a pretrained model which is loaded by the help of TensorFlow hub library. Then the preprocessed data is passed for training. Moreover, because of inefficient of computational resources such as Ram, data is divided by taking a window size 20. This model works based on LSTM, so it takes a little bit more time for training as compared to others module. The code snippets for training this model is depicted in the appendix [C].

4.4 Model Develop and Train

In this seminar report, SVM and Random forest machine learning algorithms are used for the analysis purpose. These models are loaded from the sklearn library available in python programming language. Additionally, sigmoid kernel is used for classifying the data in SVM. The preprocessed dataset is splitted into train and test in 8:2 ratio for training the model.

4.5 System Specification

The experiment is carried out in the machine with following specifications:

- **Processor**: Apple Chip M1
- **Ram**: 8 GB
- **System Type**: MacOS Monterey
- **No of Cores**: 8
- **No of Threads**: 4

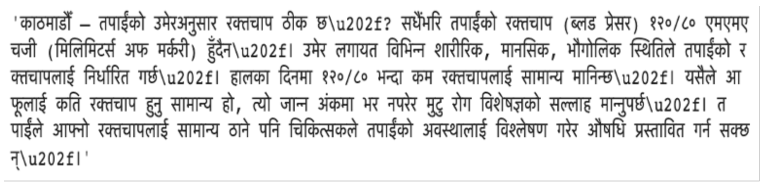
Chapter 5

RESULT AND ANALYSIS

The result of the seminar report is to analysis word embedding models based on news classification. For getting the result, number of pre-processing tasks is performed which is explained in the following section.

5.1 Preprocessing

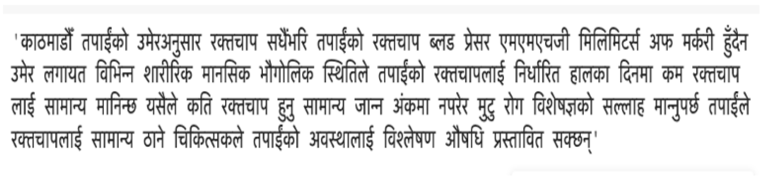
After data collection process is completed, this is not directly supplied into machine learning algorithm so, some preprocessing tasks need to be performed such as removing stop word, special character and identifying most frequent word etc. The sample of raw news is shown in the figure 5.1.



```
'काठमाडौं – तपाईंको उमेरअनुसार रक्तचाप ठीक छ\u202f? सधैंभरि तपाईंको रक्तचाप (ब्लड प्रेसर) १२०/८० एमएमए
चजी (मिलिमिटर्स अफ मर्करी) हुँदैन\u202f। उमेर लगायत विभिन्न शारीरिक, मानसिक, भौगोलिक स्थितिले तपाईंको र
क्तचापलाई निर्धारित गर्छ\u202f। हालका दिनमा १२०/८० भन्दा कम रक्तचापलाई सामान्य मानिन्छ\u202f। यसैले आ
फूलाई कति रक्तचाप हुनु सामान्य हो, त्यो जान्न अंकमा भर नपरेर मुटु रोग विशेषज्ञको सल्लाह मान्नुपर्छ\u202f। त
पाईंले आफ्नो रक्तचापलाई सामान्य ठाने पनि चिकित्सकले तपाईंको अवस्थालाई विश्लेषण गरेर औषधि प्रस्तावित गर्न सक्छ
न्\u202f।'
```

Figure 5.1: Sample of Raw Dataset

Stop words are removed by using the list of stop-words available in NLTK library. Additionally, some special characters are also removed from the raw dataset. All in all, stemming and lemmatization is not performed with these raw datasets. Dataset after completing preprocess is shown in the given figure 5.2.



'काठमाडौँ तपाईंको उमेरअनुसार रक्तचाप सधैंभरि तपाईंको रक्तचाप ब्लड प्रेसर एमएमएचजी मिलिमिटर्स अफ मर्करी हुँदैन उमेर लगायत विभिन्न शारीरिक मानसिक भौगोलिक स्थितिले तपाईंको रक्तचापलाई निर्धारित हालका दिनमा कम रक्तचाप लाई सामान्य मानिन्छ यसैले कति रक्तचाप हुनु सामान्य जान्न अंकमा नपरेर मुटु रोग विशेषज्ञको सल्लाह मान्नुपर्छ तपाईंले रक्तचापलाई सामान्य ठाने चिकित्सकले तपाईंको अवस्थालाई विश्लेषण औषधि प्रस्तावित सक्छन्'

Figure 5.2: Sample Preprocess Dataset

5.2 Time Performance Analysis

After the preprocessing task is completed, both Word2vec and ELMO modules are run with varying hyperparameter for the comparative analysis. Additionally, the time stamp required for training each of the models is captured.

5.2.1 Word2Vec

Here Word2vec model is trained with the processed data for fixed number of features and window size but having different minimum count values. By this analysis it is concluded that this model is trained comparatively fast for minimum count 3 as compared to others. The calculated data are tabulated in the table 5.1

Table 5.1: Time Taken by Word2Vec for different minimum count

Number of Features	200	200	200	200
Minimum Count	1	2	3	4
Window Size	5	5	5	5
Training Time(Second)	77	59	49	50

The above calculation is visualized in the form of line graph, and which is displayed in the figure 5.3.

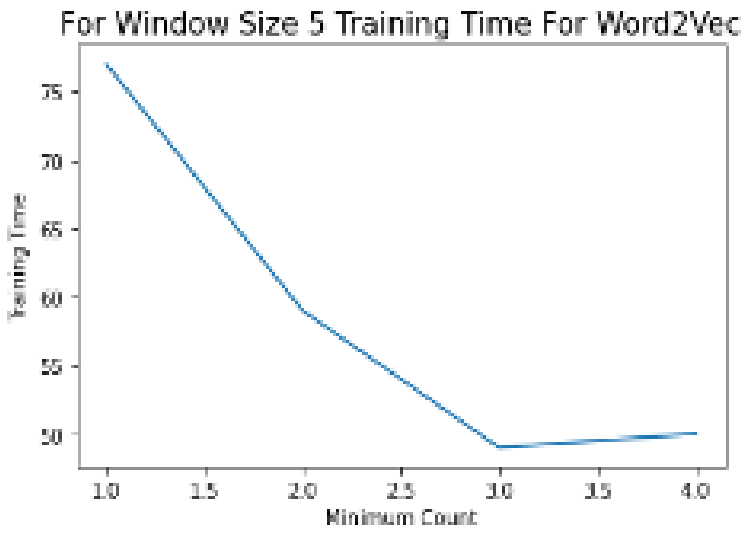


Figure 5.3: For fixed window size training time for Word2Vec

Now, here train the same model with same training dataset for various window size, Training time is unexpectedly raised while expanding the window size. The numerical estimates are tabulated in table 5.2 .

Table 5.2: For Fixed Minimum Count Training time for Word2Vec

Number of Features	200	200	200	200
Minimum Count	1	1	1	1
Window Size	4	8	10	12
Training Time(Second)	67	113	133	153

5.2.2 ELMo

Here ELMO model is trained with the processed data for fixed number of features. By this analysis it is concluded that this model takes longer time for training as compared to Word2Vec. This model takes 1 hours and 36 minutes for generating word embedding vector of preprocessing data. The calculation is depicted in the table 5.3 below.

Table 5.3: Training time for ELMO

Number of Features	200
Training Time	1 hour 36 minutes

5.3 Model Evaluation

5.3.1 SVM

For Word2Vec performance of the SVM model is remained constant while increasing the window size. Additionally, it gives maximum of 68 test accuracy for window size equal to eight. Different window size's values are tabulated in the table 5.4.

Table 5.4: SVM Model Accuracy for Word2vec with Different Window Size

Number of Features	200	200	200	200
Minimum Count	1	1	1	1
Window Size	4	8	10	12
Model Accuracy	67	68	68	68

The above calculation is visualized in the form of bar graph and which is displayed in the figure 5.4.

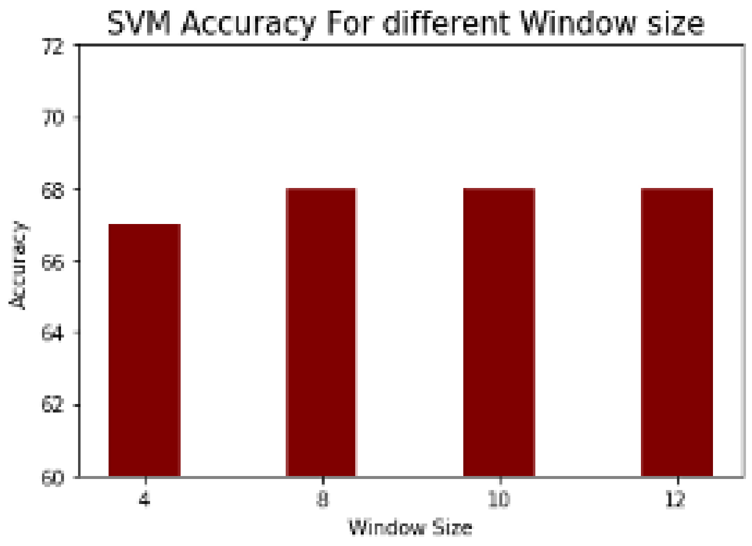


Figure 5.4: SVM Model Accuracy for Word2vec with Different Window Size

Here Support Vector Machine Learning algorithm again train with the same dataset for various minimum count. This model showing slight fluctuation behavior while increasing the minimum count to 2,3,4 and so on. This model gives the highest of 67 percent test accuracy on odd number of minimum counts in our sample count. Finally, different test accuracy on different minimum count is tabulated in the table 5.5.

Table 5.5: SVM Model Accuracy for Word2vec with Different Minimum Count

Number of Features	200	200	200	200
Minimum Count	2	3	4	5
Window Size	8	8	8	8
Model Accuracy	67	66	67	66

The above calculation is visualized in the form of bar graph and which is displayed in the figure 5.5.

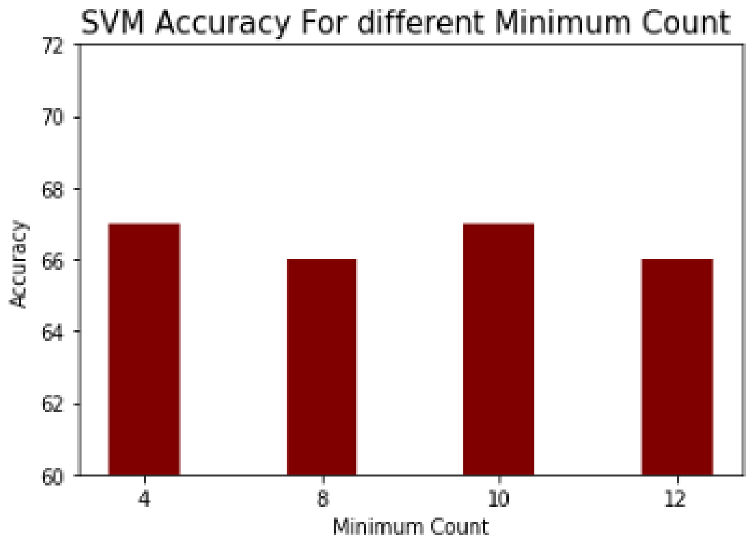
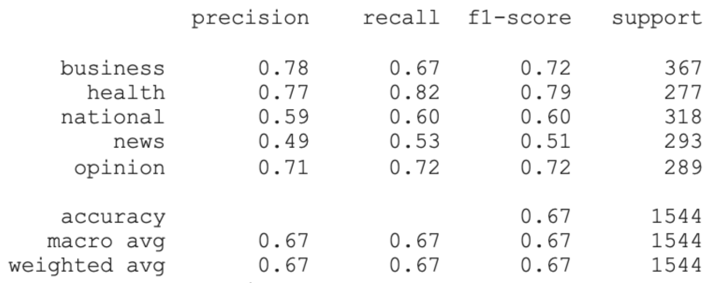


Figure 5.5: SVM Model Accuracy for Word2vec with Different Minimum Count

In this seminar report ELMo Word embedding model is also used for the analysis. Because of insufficient of the computational resources, this model is only trained by taking features size of 20. Support Vector Machine learning algorithm gives similar accuracy of 67 percent for the ELMo vectorization as compared to Word2Vec. The detailed classification report of SVM by using ELMO word embedding is shown in the figure 5.6.



```
              precision    recall  f1-score   support

    business       0.78      0.67      0.72       367
      health       0.77      0.82      0.79       277
    national       0.59      0.60      0.60       318
        news       0.49      0.53      0.51       293
     opinion       0.71      0.72      0.72       289

    accuracy                           0.67      1544
   macro avg       0.67      0.67      0.67      1544
weighted avg       0.67      0.67      0.67      1544
```

Figure 5.6: SVM Classification Report for ELMO Word embedding

5.3.2 Random Forest

For Word2Vec performance of the Random Forest model is increased slightly as compared to the SVM model. For taking 200 features and a single count this model gives the highest of 73 percent test accuracy on window size 4. The data are presented in the table 5.6.

Table 5.6: Random Forest Model Accuracy for Word2vec with Different Window Size

Number of Features	200	200	200	200
Minimum Count	1	1	1	1
Window Size	4	8	10	12
Model Accuracy	73	70	70	71

The above calculation is converted into the form of bar graph which is depicted in the 5.7.

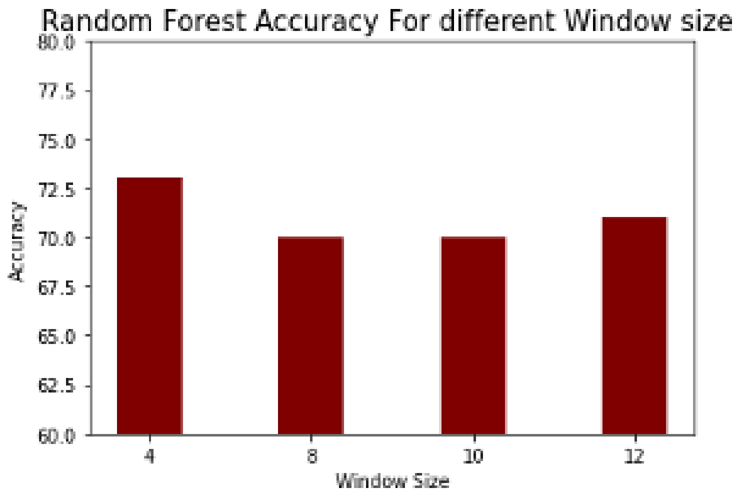


Figure 5.7: Random Forest Model Accuracy for Word2vec with Different Window Size

For varying number of minimum counts and constant number of features and window size. The model accuracy is plummeted noticeably. Finally, different test accuracy on different minimum count is tabulated in the table 5.7.

Table 5.7: SVM Model Accuracy for Word2vec with Different Minimum Count

Number of Features	200	200	200	200
Minimum Count	1	2	3	4
Window Size	4	4	4	4
Model Accuracy	73	71	71	71

The above calculation is depicted in the figure .

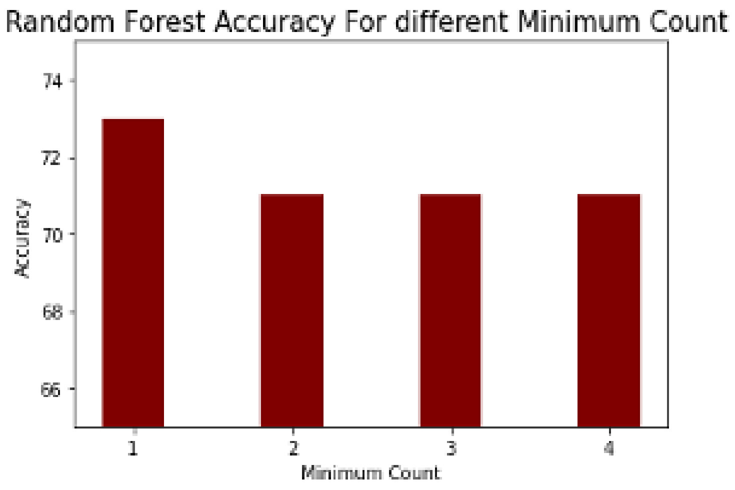
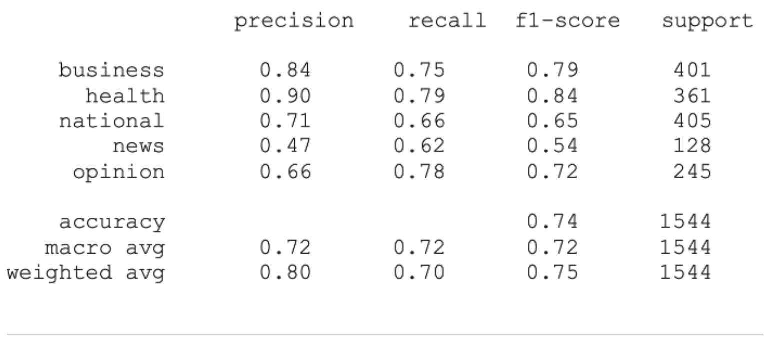


Figure 5.8: Random Forest Model Accuracy for Word2vec with Different Window Size

The word embedding value generated by the ELMO is also used for training the random forest machine learning algorithm. For this embedding random forest gives slightly higher accuracy as compared to the ELMO, it secures 74 percent test accuracy for the news classification system. All in all, classification report is shown in the figure 5.9.



```
              precision    recall  f1-score   support

    business       0.84      0.75      0.79       401
      health       0.90      0.79      0.84       361
    national       0.71      0.66      0.65       405
        news       0.47      0.62      0.54       128
     opinion       0.66      0.78      0.72       245

    accuracy                           0.74      1544
   macro avg       0.72      0.72      0.72      1544
weighted avg       0.80      0.70      0.75      1544
```

Figure 5.9: Random Forest Classification Report for ELMO Word embedding

Chapter 6

CONCLUSION

Comparative analysis between different word embedding model is done in this seminar report. The goal of the task is to evaluate system for the Nepali news classification. So, this report evaluated the different word representation methods in the framework of a controlled task. Nepali is a complex language with a wide range of vocabulary containing many rare words. Language structure use of complex words, multiple meaning in different context all these reasons make it difficult to choose one model as the best for Nepali word classification. Additionally, the content of the dataset also plays the vital role in deciding this.

The two word embedding techniques, Word2Vec and ELMO as well as two machine learning algorithms namely SVM and Random Forest are used for the comparative study for the Nepali text classification. Additionally, the dataset for this study is managed by web scraping process. As this report evaluation shows, the model with ELMO vector slightly caught up and surpassed the model with Word2Vec, obtaining higher test accuracy.

It is also observed that using Random Forest to classify news led to the high-test accuracy as well as precision and recall as compared to the classification report generated by the Support Vector Machine learning algorithm. However, this model gives the slight balanced classification report.

Contrary to the expectation to this seminar, contextualized ELMo vectors did not result in notable performance gain compared to regular Word2Vec model. This could be in part since the stemming with the Nepali text is not easily done in this time. However, increasing the size of dataset and the big overhead in training and testing time for the ELMo model could improve the results.

References

[1] BM Ajose-Ismail, Olawale Victor Abimbola, and SA Oloruntoba. Performance analysis of different word embedding models for text classification. *International Journal of Scientific Research and Engineering Development*, 3(6):1016–1020, 2020.

[2] Snehal Bhoir, Tushar Ghorpade, and Vanita Mane. Comparative analysis of different word embedding models. In *2017 International conference on advances in computing, communication and Control (ICAC3)*, pages 1–4. IEEE, 2017.

[3] Elena Fano. A comparative study of word embedding methods for early risk prediction on the internet, 2019.

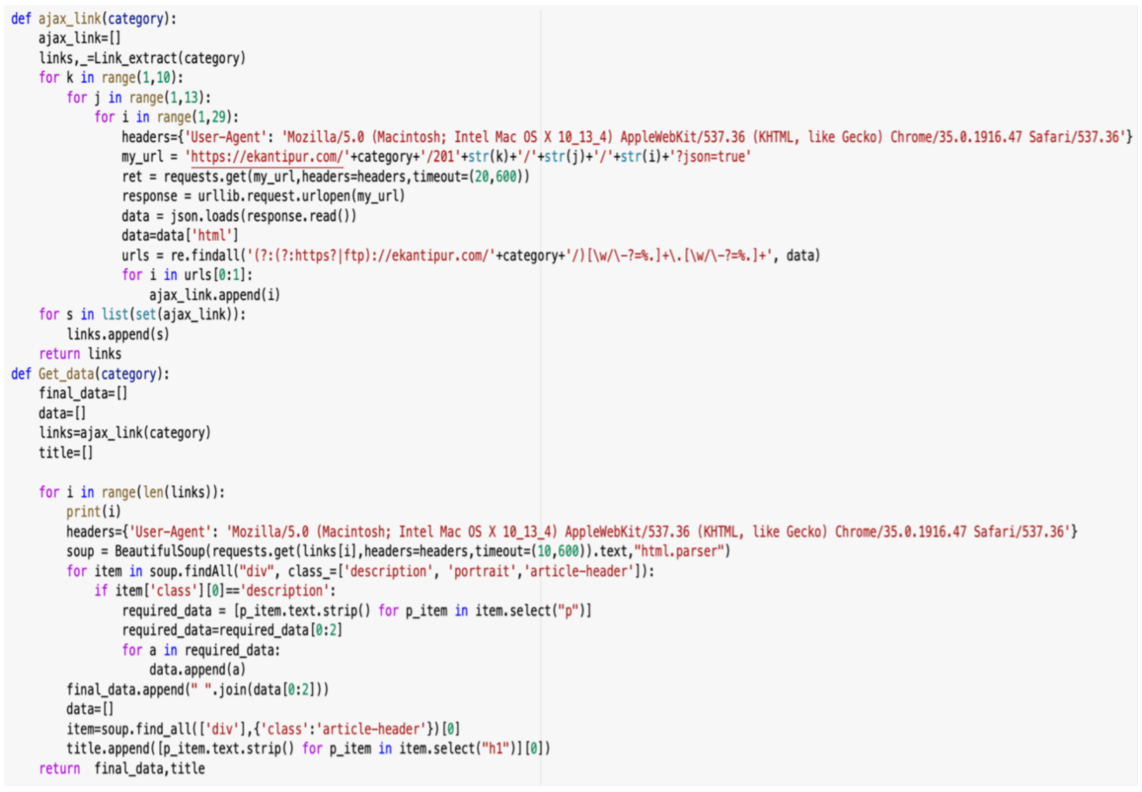
[4] Pravesh Koirala and Nobal B Niraula. Npvec1: Word embeddings for nepali-construction and evaluation. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 174–184, 2021.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Zakia Sultana Ritu, Nafisa Nowshin, Md Mahadi Hasan Nahid, and Sabir Ismail. Performance analysis of different word embedding models on bangla language. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–5. IEEE, 2018.

[7] Justyna Sarzynska-Wawer, Aleksander Wawer, Aleksandra Pawlak, Julia Szymanowska, Izabela Stefaniak, Michal Jarkiewicz, and Lukasz Okruszek. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research*, 304:114135, 2021.

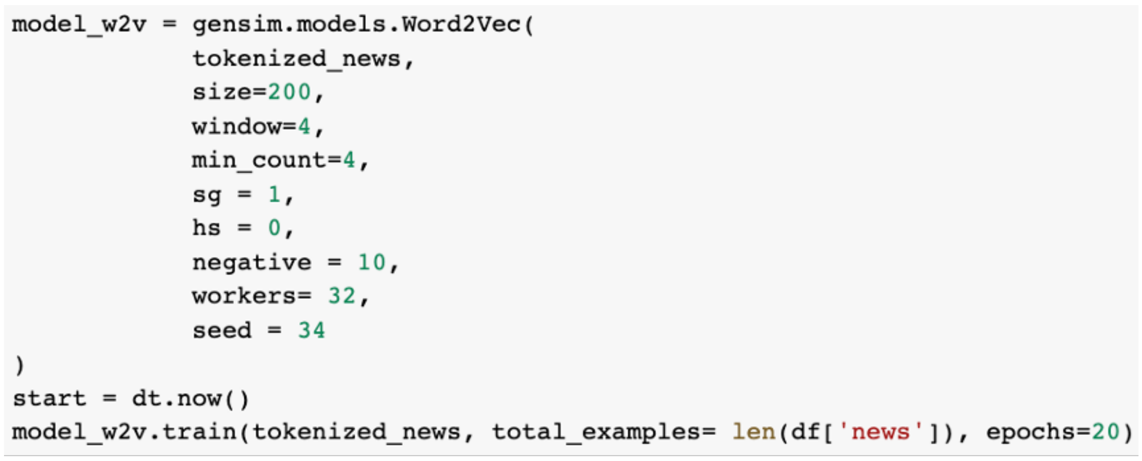
Appendix



```python
def ajax_link(category):
    ajax_link=[]
    links,_=Link_extract(category)
    for k in range(1,10):
        for j in range(1,13):
            for i in range(1,29):
                headers={'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.47 Safari/537.36'}
                my_url = 'https://ekantipur.com/'+category+'/201'+str(k)+'/'+str(j)+'/'+str(i)+'?json=true'
                ret = requests.get(my_url,headers=headers,timeout=(20,600))
                response = urllib.request.urlopen(my_url)
                data = json.loads(response.read())
                data=data['html']
                urls = re.findall('(?:(?:https?|ftp)://ekantipur.com/'+category+'/)[\w/\-?=%.]+\.[\w/\-?=%.]+', data)
                for i in urls[0:1]:
                    ajax_link.append(i)
    for s in list(set(ajax_link)):
        links.append(s)
    return links
def Get_data(category):
    final_data=[]
    data=[]
    links=ajax_link(category)
    title=[]

    for i in range(len(links)):
        print(i)
        headers={'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.47 Safari/537.36'}
        soup = BeautifulSoup(requests.get(links[i],headers=headers,timeout=(10,600)).text,"html.parser")
        for item in soup.findAll("div", class_=['description', 'portrait','article-header']):
            if item['class'][0]=='description':
                required_data = [p_item.text.strip() for p_item in item.select("p")]
                required_data=required_data[0:2]
                for a in required_data:
                    data.append(a)
        final_data.append(" ".join(data[0:2]))
        data=[]
        item=soup.find_all(['div'],{'class':'article-header'})[0]
        title.append([p_item.text.strip() for p_item in item.select("h1")][0])
    return  final_data,title
```

[A] Code snippet for Scraping Data.



```python
model_w2v = gensim.models.Word2Vec(
            tokenized_news,
            size=200,
            window=4,
            min_count=4,
            sg = 1,
            hs = 0,
            negative = 10,
            workers= 32,
            seed = 34
)
start = dt.now()
model_w2v.train(tokenized_news, total_examples= len(df['news']), epochs=20)
```

[B] Code snippet for Word2Vect model training.

```python
import tensorflow_hub as hub
import tensorflow as tf

elmo = hub.Module("https://tfhub.dev/google/elmo/2", trainable=True)
def elmo_vectors(x):
  embeddings = elmo(x.tolist(), signature="default", as_dict=True)["elmo"]

  with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())
    # return average of ELMo features
    return sess.run(tf.reduce_mean(embeddings,1))
list_train = [df[i:i+20] for i in range(0,df.shape[0],20)]
elmo_train = [elmo_vectors(x['news']) for x in list_train]
```

[B] Code snippet for ELMO model training.

```
['छ', 'र', 'पनि', 'छन्', 'लागि', 'भएको', 'गरेको', 'भने', 'गर्न',
 'गर्ने', 'हो', 'तथा', 'यो', 'रहेको', 'उनले', 'थियो', 'हुने',
 'गरेका', 'थिए', 'गर्दै', 'तर', 'नै', 'को', 'मा', 'हुन्', 'भन्ने',
 'हुन', 'गरी', 'त', 'हुन्छ', 'अब', 'के', 'रहेका', 'गरेर', 'छैन',
 'दिए', 'भए', 'यस', 'ले', 'गर्नु', 'औं', 'सो', 'त्यो', 'कि', 'जुन',
 'यी', 'का', 'गरि', 'ती', 'न', 'छु', 'छौं', 'लाई', 'नि', 'उप',
 'अक्सर', 'आदि', 'कसरी', 'क्रमशः', 'चाले', 'अगाडी', 'अझै', 'अनुसार',
 'अन्तर्गत', 'अन्य', 'अन्यत्र', 'अन्यथा', 'अरु', 'अरुलाई', 'अर्को',
 'अर्थात', 'अर्थात्', 'अलग', 'आए', 'आजको', 'ओठ', 'आत्म', 'आफू',
 'आफूलाई', 'आफ्नै', 'आफ्नो', 'आयो', 'उदाहरण', 'उनको', 'उहालाई',
 'एउटै', 'एक', 'एकदम', 'कतै', 'कम से कम', 'कसै', 'कसैले', 'कहाँबाट',
 'कहिलेकाहीं', 'का', 'किन', 'किनभने', 'कुनै', 'कुरा', 'कृपया',
 'केही', 'कोही', 'गए', 'गरौं', 'गर्छ', 'गर्छु', 'गर्नुपर्छ', 'गयौ',
 'गैर', 'चार', 'चाहनुहुन्छ', 'चाहन्छु', 'चाहिए', 'छू', 'जताततै',
 'जब', 'जबकि', 'जसको', 'जसबाट', 'जसमा', 'जसलाई', 'जसले', 'जस्तै'
 'जस्तो', 'जस्तोसुकै', 'जहाँ', 'जान', 'जाहिर', 'जे', 'जो', 'ठीक',
 'तत्काल', 'तदनुसार', 'तपाईको', 'तपाई', 'पर्याप्त', 'पहिले',
 'पहिलो', 'पहिल्यै', 'पाँच', 'पाँचौं', 'तल', 'तापनी', 'तिनी',
 'तिनीहरू', 'तिनीहरुको', 'तिनिहरुलाई', 'तिमी', 'तिर', 'तीन',
 'तुरुन्तै', 'तेस्रो', 'तेस्कारण', 'पूर्व', 'प्रति', 'प्रतेक',
 'प्लस', 'फेरी', 'बने', 'त्सपछि', 'त्सैले', 'त्यहाँ', 'थिएन',
 'दिनुभएको', 'दिनुहुन्छ', 'दुई', 'देखि', 'बरु', 'बारे', 'बाहिर',
 'देखिन्छ', 'देखियो', 'देखे', 'देखेको', 'देखेर', 'दोस्रो', 'धेरै',
 'नजिकै', 'नत्र', 'नयाँ', 'निम्ति', 'बाहेक', 'बीच', 'बीचमा', 'भन',
 'निम्न', 'निम्नानुसार', 'निर्दिष्ट', 'नौ', 'पक्का', 'पक्कै', 'पछि',
 'पछिल्लो', 'पटक', 'पर्छ', 'पर्थ्यो', 'भन्छन्', 'भन्', 'भन्छु',
 'भन्दा', 'भन्नुभयो', 'भर', 'भित्र', 'भित्री', 'म', 'मलाई', 'मात्र',
 'माथि', 'मुख्य', 'मेरो', 'यति', 'यथोचित', 'यदि', 'यद्यपि', 'यसको',
 'यसपछि', 'यसबाहेक', 'यसरी', 'यसो', 'यस्तो', 'यहाँ', 'यहाँसम्म',
 'या', 'रही', 'राखे', 'राख्छ', 'राम्रो', 'रूप', 'लगभग', 'वरीपरी',
 'वास्तवमा', 'बिरुद्ध', 'बिशेष', 'सायद', 'शायद', 'संग', 'संगै',
 'सक्छ', 'सट्टा', 'सधै', 'सबै', 'सबैलाई', 'समय', 'सम्भव', 'सम्म',
 'सही', 'साँच्चै', 'सात', 'साथ', 'साथै', 'सारा', 'सोही', 'स्पष्ट',
 'हरे', 'हरेक'], dtype='<U11')
```

[C] List of Stop Words