Tribhuvan University
Institute of Science and Technology

Word Embedding Based Feature Extraction for Nepali News Classification

Dissertation
Submitted to:

Central Department of Computer Science and Information Technology
Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements for the Master's Degree in
Computer Science and Information Technology

by

Ramesh Kumar Chaudhary

7th February, 2019

Tribhuvan University
Institute of Science and Technology

Word Embedding Based Feature Extraction for Nepali News Classification

Dissertation
Submitted to

Central Department of Computer Science and Information Technology
Kirtipur, Kathmandu, Nepal

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science & Information Technology (M.SC.CSIT)

Submitted By:

Mr. Ramesh Kumar Chaudhary
Date: 7th February, 2019

Supervisor:

Prof. Dr. Subarna Shakya
Dept. of Electronics & Computer Engineering, IOE,
Tribhuvan University, Pulchwok, Nepal

Co-Supervisor:

Mr. Tej Bahadur Shahi
Lecturer
CDCSIT, Tribhuvan University Kirtipur, Nepal

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

… … … … … … … … … …

Ramesh Kumar Chaudhary

Date: 7th February, 2019

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Supervisor's Recommendation

We hereby recommend that this dissertation prepared under our supervision by **Mr. Ramesh Kumar Chaudhary** entitled **"Word Embedding Based Feature Extraction for Nepali News Classification**" be accepted as partial fulfillment of the requirements for the degree of M. Sc. in Computer Science and Information Technology. In our best knowledge this is an original work in computer science.

……..

Prof. Dr. Subarna Shakya
Department of Electronics and Computer
Engineering, IOC, Tribhuvan University
Pulchok, Lalitpur, Nepal

Date: 7th February, 2019

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

LETTER OF APPROVAL

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Master's Degree in Computer Science and Information Technology

Evaluation Committee

……..

Prof. Dr. Subarna Shakya
Department of Electronics and Computer Engineering, IOE
Tribhuvan University
Pulchok, Lalitpur, Nepal
(Supervisor)

……..

Ast. Prof. Navraj Paudel
Head of Department (HOD)
Central Department of Computer Science and Information Technology(CDCSIT)
Tribhuvan University
Kritipur, Kathmandu, Nepal

… … … … … … … … … … …

(Internal Examiner)

… … … … … … … … … … …

(External Examiner)

Date: 7th February, 2019

ACKNOWLEDGEMENT

It is a great pleasure for me to acknowledge the contributions of a large number of individuals to this work. I deeply extend my heartily acknowledgement to my respected teacher and dissertation advisor Prof. Dr. Subarna Shakya for giving me an opportunity to work under his supervision and for providing me guidance and support throughout this work. With this regard, I wish to extend my sincere appreciation to respected Co-supervisor Ast. Prof. Tej Bahadur Shahi, Central Department of Computer Science and Information Technology (CDCSIT) for his valuable suggestions.

I would like to express my gratitude to the respected teachers Prof. Dr. Shashidhar Ram Joshi, Mr. Nawaraj Paudel, Mr. Dhiraj Kedar Pandey, Mr. Arjun Singh Saud, Mr. Jagdish Bhatt, Mr. Sarbin Sayami, Mrs. Lalita Sthapit, Mr. Bikash Balami and others staffs of CDSCIT for granting me broad knowledge and inspirations within the time period of two years.

I cannot remain without admiring the efforts put by my friends and others for their exceptional participation on this work. Last but not list, I would like to thank my family members for their constant support and encouragement.

ABSTRACT

A major challenge in topic classification (TC) is the high dimensionality of the feature space. Therefore, feature extraction (FE) plays a vital role in topic classification in particular and text mining in general. FE based on cosine similarity score is commonly used to reduce the dimensionality of datasets with tens or hundreds of thousands of features, which can be impossible to process further. In this study, TF-IDF (Term Frequency Inverse Document Frequency) term weighting is used to extract features. Selecting relevant features and determining how to encode them for a learning machine method have a vast impact on the learning machine methods ability to extract a good model.

Count based feature extraction methods is compared with word to vector feature extraction techniques for Nepali news classification. The results show good classification performance when using the feature extraction techniques based on word to vector for less number of classes and drastically decrease the performance for large sample size. On the other hand result of classification count based technique shows consistent nearly performance for any number of classes. The overall performance of the TF-IDF (Term Frequency Inverse Document Frequency) is far better than both word to vector techniques.

Keywords—*feature extraction, topic classification, cosine similarity score, TF-IDF, CBOW, Skip-gram, Text mining, neural networks, deep learning*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

SVM	Support Vector Machine
TF-IDF	Term Frequency Inverse Document Frequency
CBOW	Continuous Bag of Word
NLP	Natural Language Processing
NLU	Natural Language Understanding
NLG	Natural Language Generation
POS	Post of Speech
AI	Artificial Intelligence
LSA	Latent Semantic Analysis
SVD	Singular Value Decomposition
PCA	Principle Component Analysis
CNN	Convolutional Neural Network
MLP	Multi-Layered Perceptron
RNN	Recurrent Neural Network
LDA	Linear Discriminant Analysis
NDA	Nonlinear Discriminant Analysis
ENA	Ethiopian News Agency

CHAPTER ONE

INTRODUCTION

1.1 Introduction

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. The classification can be flat or hierarchical. [1, 7]

Text Categorization (TC), also known as Text Classification, is the task of automatically classifying a set of text documents into different categories from a predefined set. [4] Consider the case of sorting and organizing emails, files in folder hierarchies so that topic identification that would support topic specific operations be made. On such attempt is the yahoo web directory. [5] If such classification is to be done manually 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).

So the need to employee machine learning to automate the classification is needed.

1.2 Problem Statement

The problem of news classification is to assign a news type label to each news automatically by computer program. It comes under the heading of supervised machine learning technique. For Nepali News, the available technique for other standard language such as English can't be directly used for the classification task. Since the Nepali language is morphologically rich and has many deflection and derivational form of words. The main problem towards classification is not the availability of highly accurate classifier but the number of feature fed to these classifier. If the number of features are correctly chosen, the classification algorithm works well. In this research work, the best feature extraction method will be find out among the some popular technique Term Frequency- Inverse Document Frequency (TF-IDF) [8], Continuous Bag of words (CBOW) and Skip-Gram method. [2]

For the purpose of classification task, the Support Vector Machine (SVM) with best feature extracted will be used to measure the accuracy and efficiency of the feature extraction technique. [12]

1.3 Objectives

The main objectives of this research work is to extract the feature using different technique and efficiently classify news types using SVM. The other objectives are:

1. To build a Nepali news corpus by crawling different online Nepali News portals.
2. To extracting feature vector of Nepali news based on methods: Term Frequency Inverse Document Frequency (TF-IDF), Continuous Bag of words (CBOW) and Skip-Gram method.
3. To build Nepali news type classifier using Support Vector Machine (SVM) and analyze the accuracy of above feature extraction methods.

1.4 Background

1.4.1 Natural Language Processing

Natural Language Processing (NLP) has been developed in 1960 as a subfield of Artificial Intelligence and Linguistics [14]. The aim of NLP is studying problems in the automatic generation and understanding of natural language. A Natural Language is any of the languages naturally used by humans, *i.e.* not an artificial or machine language such as a programming language like C language, Java, Perl etc.

NLP is a convenient description for all attempts to use computers to process natural language. NLP is also an area of artificial intelligence research that attempts to reproduce the human interpretation of language for computer system processing. The ultimate goal of NLP is to determine a system of language, words, relations, and conceptual information that can be used by computer logic to implement artificial language interpretation. NLP includes anything a computer needs to understand natural language (written or spoken) and also generate the natural language. To build computational natural language systems, we need Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLG systems convert information from computer databases into normal-sounding human language, and NLU systems convert samples of human language into more representation that are easier for computer programs to manipulate. Some of important levels of NLP are as follows:

Phonological Analysis: Phonology is the study of sound system in a language. The minimal unit of sound system is the phoneme which is capable of distinguishing the meanings in the words. The phonemes combine to form a higher level unit called syllable and syllables combine to form the words. Therefore, the organization of the sounds in a language exhibits the linguistic as well as computational challenges for its analysis. This level deals with the interpretation of speech sounds within and across words. There are, in fact, three types of rules used in phonological analysis: 1) phonetic rules – for sounds within words; 2) phonemic rules – for variations of pronunciation when words are spoken together, and; 3) prosodic rules – for fluctuation in stress and intonation across a sentence. In an NLP system that accepts spoken input, the sound waves are analyzed and encoded into a

digitized signal for interpretation by various rules or by comparison to the particular language model being utilized.

Morphological Analysis: This level deals with the componential nature of words, which are composed of morphemes – the smallest units of semantic meaning. For example, the word preregistration can be morphologically analyzed into three separate morphemes: the prefix pre, the root "registra", and the suffix "tion". Since the meaning of each morpheme remains the same across words, humans can break down an unknown word into its constituent morphemes in order to understand its meaning. Similarly, an NLP system can recognize the meaning conveyed by each morpheme in order to gain and represent meaning. For example, adding the suffix "ed" to a verb, conveys that the action of the verb took place in the past. This is a key piece of meaning, and in fact, is frequently only evidenced in a text by the use of the -ed morpheme. Typically, a natural language processor knows how to understand multiple forms of a word *i.e.* its plural and singular, for example, *ghar (*घर*)* "house" *ghar-haru (*घरहरु*)* "house-s". From structural point of view, the words can be simple, complex and compound. For example, *ghar* "house", *ghar-haru* "house-Plural", *ghar-ghar* "each house".

Lexical Analysis: At this level, humans, as well as NLP systems, interpret the meaning of individual words. Several types of processing contribute to word-level understanding – the first of these being assignment of a single part-of-speech (POS) tag to each word. In this processing, words that can function as more than one part-of-speech are assigned the most probable part-of speech tag based on the context in which they occur. The lexical level may require a lexicon, and the particular approach taken by an NLP system will determine whether a lexicon will be utilized, as well as the nature and extent of information that is encoded in the lexicon.

Syntactic Analysis: Syntactic analysis uses the results of morphological analysis and lexical analysis to build a structural description of the sentence. The goal of this process, called parsing, is to convert the flat list of words that forms the sentence into a structure that defines the units that are represented by that flat list. The important thing here is that a flat list of words has been converted into a hierarchical structure and that the structures correspond to meaning units when semantic analysis is performed.

Semantic Analysis: It derives an absolute (dictionary definition) meaning from context; it determines the possible meaning of a sentence in a context .The structures created by the syntactic analyzer are assigned meaning. Thus, a mapping is made between individual words into appropriate objects in the knowledge base or data base. It must create the correct structure s to correspond to the way the meaning of the individual words combine with each other. The structures for which no such mapping is possible are rejected.

Example: the sentence "colorless green ideas…." would be rejected as it has no such semantic mapping, because colorless and green make no sense.

Discourse Integration: The meaning of an individual sentence may depend on the sentences that precede it and may influence the meaning of the sentences that follow it.

Example: the meaning of word "it" in the sentence, "you wanted it" depends on the previous discourse context.

Pragmatic Analysis: It derives knowledge from external commonsense information; it means understanding the purposeful use of language in situations, particularly those aspects of language which require world knowledge.

Example: If someone says "the door is open" then it is necessary to know which door "the door" refers to; here it is necessary to know what the intention of the speaker: could be a pure statement of fact, could be an explanation of how the cat got in, or could be a request to the person addressed to close the door.

1.4.2 Major Application of Natural Language Processing

NLP is having a very important place in our day-to-day life due to its large natural language applications. By means of these NLP applications the user can interact with computers in their own mother tongue by means of a keyword and a screen. The few NLP processes are:

- Part-of-speech tagging
- Information retrieval
- Machine translation

- Named entity recognition
- Natural language generation
- Question answering
- Spoken dialogue system
- Text simplification
- Text to speech
- Speech recognition etc.

1.4.3 Computational Linguistics

Computational linguistics is the scientific study of language (i.e. statistical and/or rule-based modeling of natural language) from a computational perspective. Traditionally, computational linguistics was usually performed by computer scientists who had specialized in the application of computers to the processing of a natural language. Computational linguists often work as members of interdisciplinary teams, including linguists (specifically trained in linguistics), language experts (persons with some level of ability in the languages relevant to a given project), and computer scientists. In general, computational linguistics draws upon the involvement of linguists, computer scientists, and experts in artificial intelligence, mathematicians, logicians, cognitive scientists, cognitive psychologists, psycholinguists, anthropologists and neuroscientists, amongst others. Some of the areas of research that are studied by computational linguistics include:

- Computational complexity of natural language, largely modeled on automata theory, with the application of context-sensitive grammar.
- Computational semantics comprises defining suitable logics for linguistic meaning representation, automatically constructing them and reasoning with them.
- Computer-aided corpus linguistics.
- Design of parsers or chunkers for natural languages.
- Design of taggers like POS-taggers.

- Machine translation.

1.4.4 Corpus linguistics

Corpus linguistics is now seen as the study of linguistic phenomena through large collections of machine-readable texts: **corpora**. These are used within a number of research areas going from the descriptive study of the syntax of a language to language learning. Corpus linguistics has developed considerably in the last decades due to the great possibilities offered by the processing of natural language by computers having large storage capacity. The availability of computers and machine-readable text has made it possible to get data quickly and easily and also to have this data presented in a format suitable for analysis. Corpus linguistics is, however, not the same as mainly obtaining language data through the use of computers. Corpus linguistics is the study and analysis of data obtained from a corpus. The main task of the corpus linguist is not to find the data but to analyze it. Computers are useful, and sometimes indispensable, tools used in this process.

1.4.5 Machine learning

It is a recent field of artificial intelligence (AI) which aim to make a machine able to learn as human learns the things. Marvin Minsky (1986) defined learning as *"it is making useful change in the working of our mind".* Machine learning exists in various forms: supervised learning, unsupervised learning, semi supervised or minimally supervised learning, reinforcement learning etc. In its basic form, machine learn the knowledge form some sources and then generalize that knowledge for other instances.

1.4.6 Text Representation

1.4.6.1 Bag of words representation

Under this form, every sentence in the document is considered to be a multi-set or bag of words (or tokens) without considering the grammar or even the word order in it. Here, the occurrence or frequency of the word collectively contributes in features for further

classification. [27] For e.g. consider the following two documents containing sentences as below:

D1: Meera likes dancing a lot.

D2: John too likes dancing but not that much.

For the above documents, one combined list is made:

[“Meera”, “likes”, “dancing”, “a, “lot”, “John”, “too”, “but”, “not”, “that”, “much”]

1.4.6.2 Vector Space Model

This is an algebraic model for text representation. It consists of three stages [5, 10]:

Stage 1: Indexing of the documents where the content bearing terms [6] are extracted from the document text. The terms having very high or very low frequency distract the learning and hence are eliminated. Such words are known as function words [6, 7, and 8]. These include the highly occurring stop words like “a, an, the, on”. For e.g.:

“**New York** is using sand-filled trucks to protect **Thanks giving** parade”.

Here, the words in bold are the content bearing words.

Stage 2: Weighting of the indexed terms for the enhancement of the retrieval of relevant document. There are many ways to give weight to the terms depending upon the application.

$D_j = (w_{1,j}, w_{2,j} \ldots w_{n,j})$ is the representation of document in terms of weights.

Here, each dimension corresponds to an independent term. Zero shows absence of any term from the document.

Stage 3: Ranking of the documents taking the similarity measure into consideration to get the closet words from query document.

1.4.7 Feature Extraction

Feature Vector Construction is an important approach, it provides a lot of information regarding the text documents such as the highest and lowest term frequency for each document. Selecting relevant features and determining how to encode them for a learning machine method can have a vast impact on the learning machine methods ability to extract a good model.

1.4.7.1 Latent Semantic Analysis (LSA)

LSA method is a novel technique in text classification. Generally, LSA analyzes relationships between a term and concepts contained in an unstructured collection of text. It is called Latent Semantic Analysis, because of its ability to correlate semantically related terms that are latent in a text. LSA produces a set of concepts, which is smaller in size than the original set, related to documents and terms [11, 12]. It uses SVD (Singular Value Decomposing) to identify pat- tern between the terms & concepts contained in the text, and find the relationships between documents. The method commonly referred to as concept searches. It has ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts. LSA is mostly used for page retrieval systems and text clustering purposes. LSA overcomes two of the most problematic keyword queries: multiple words that have similar meanings and words that have more than one meaning.

1.4.7.2 FE Based on Cosine Similarity Score

FE based on cosine similarity score is commonly used to reduce the dimensionality of datasets with tens or hundreds of thousands of features, which can be impossible to process further. [1]

1.4.7.3 PCA- Principle Component Analysis

PCA is a well-known technique that can reduce the dimensionality of data by transforming the original attribute space into smaller space. In the other word, the purpose of principle components analysis is to derive new variables that are combinations of the original variables and are uncorrelated. This is achieved by transforming the original variables Y = [y_1, y_2... y_p] (where p is number of original variable) to a new set of variables, T = [t_1, t_2,..., t_q] (where q is number of new variables), which are combinations of the original variables. Transformed attributes are framed by first, computing the mean (μ) of the dataset, then covariance matrix of the original attributes is calculated [5]. And the second step is, extracting its eigenvectors. The eigenvectors (principal components) introduce as a linear transformation from the original attribute space to a new space in which attributes are uncorrelated. Eigenvectors can be sorted according to the amount of variation in the original data. The best n eigenvectors (those one with highest eigenvalues) are selected as new features while the rest are discarded.

1.4.7.4 Artificial Neural Network

Artificial neural networks are one of the main tools used in machine learning. As the "neural" part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize. [22]

1.4.7.5 Convolutional Neural Network

Convolutions are great for extracting features from dataset. Convolutional Neural Networks (CNN) are biologically-inspired variants of MLPs. CNNs are networks composed of several layers of convolutions with nonlinear activation functions like ReLU

or tanh applied to the results. Traditional Layers are fully connected, instead CNN use local connections. Each layer applies different filters (thousands) and combines their results. [30]

1.4.7.6 Neuro-Fuzzy Method

A fuzzy feature evaluation index for a set of features is newly defined in terms of degree of similarity between two patterns in both the original and transformed feature spaces. A layered network is designed for performing the task of minimization of the evaluation index through unsupervised learning process. This extracts a set of optimum transformed features, by projecting n-dimensional original space directly to n-dimensional (n:n) transformed space, along with their relative importance.

1.4.8 Text Classification

Text classification algorithms are at the heart of a variety of software systems that process text data at scale. Email software uses text classification to determine whether incoming mail is sent to the inbox or filtered into the spam folder. Discussion forums use text classification to determine whether comments should be flagged as inappropriate.

1.4.8.1 Rule Based System

Rule-based approaches classify text into organized groups by using a set of handcrafted linguistic rules. These rules instruct the system to use semantically relevant elements of a text to identify relevant categories based on its content. Each rule consists of an antecedent or pattern and a predicted category. [4]

Say that need to classify news articles into 2 groups, namely, Sports and Politics. First, need to define two lists of words that characterize each group (e.g. words related to sports such as football, basketball, LeBron James, etc., and words related to politics such as Donald Trump, Hillary Clinton, Putin, etc.). Next, when want to classify a new incoming text, need to count the number of sport-related words that appear in the text and do the

same for politics-related words. If the number of sport-related word appearances is greater than the number of politics-related word count, then the text is classified as sports and vice versa.

For example, this rule-based system will classify the headline “When is LeBron James' first game with the Lakers?” as Sports because it counted 1 sport-related term (Lebron James) and it didn’t count any politics-related terms.

Rule-based systems are human comprehensible and can be improved over time. But this approach has some disadvantages. They also time-consuming, since generating rules for a complex system can be quite challenging and usually requires a lot of analysis and testing. Rule-based systems are also difficult to maintain and don’t scale well given that adding new rules can affect the results of the pre-existing rules. [4]

1.4.8.2 Machine Learning Based System

Instead of relying on manually crafted rules, text classification with machine learning learns to make classifications based on past observations. By using pre-labeled examples as training data, a machine learning algorithm can learn the different associations between pieces of text and that a particular output (i.e. tags) is expected for a particular input (i.e. text). [5, 6]

The first step towards training a classifier with machine learning is feature extraction: a method is used to transform each text into a numerical representation in the form of a vector. One of the most frequently used approaches is bag of words, where a vector represents the frequency of a word in a predefined dictionary of words. [9]

Then, the machine learning algorithm is fed with training data that consists of pairs of feature sets (vectors for each text example) and tags (e.g. sports, politics) to produce a classification model. [7]

Once it’s trained with enough training samples, the machine learning model can begin to make accurate predictions. The same feature extractor is used to transform unseen text to

feature sets which can be fed into the classification model to get predictions on tags (e.g. sports, politics)

Text classification with machine learning is usually much more accurate than human-crafted rule systems, especially on complex classification tasks. Also, classifiers with machine learning are easier to maintain and you can always tag new examples to learn new tasks. [18]

1.4.8.3 Hybrid System

Hybrids systems combine a base classifier trained with machine learning and a rule-based system, which is used to further improve the results. These hybrid systems can be easily fine-tuned by adding specific rules for those conflicting tags that haven't been correctly modeled by the base classifier. [4, 18]

1.4.8.4 Text Classification Algorithms

Some of the most popular machine learning algorithms for creating text classification models include the Naive Bayes family of algorithms, support vector machines, and deep learning.

1.4.8.4.1 Naïve Bayes

Naive Bayes is a family of statistical algorithms it can be used of when doing text classification. One of the members of that family is Multinomial Naive Bayes (MNB). One of its main advantages is that it can be get really good results when data available is not much and computational resources are scarce.

All its need to know is that Naive Bayes is based on Bayes's Theorem, which help to compute the conditional probabilities of occurrence of two events based on the probabilities of occurrence of each individual event. This means that any vector that represents a text

will have to contain information about the probabilities of appearance of the words of the text within the texts of a given category so that the algorithm can compute the likelihood of that text's belonging to the category. [17]

1.4.8.4.2 Support Vector Machine

Support Vector Machines (SVM) is just one out of many algorithms it can be chosen from when doing text classification. Like Naive Bayes, SVM doesn't need much training data to start providing accurate results. Although it needs more computational resources than Naive Bayes, SVM can achieve more accurate results. [19]

In short, SVM takes care of drawing a "line" or hyperplane that divides a space into two subspaces: one subspace that contains vectors that belong to a group and another subspace that contains vectors that do not belong to that group. Those vectors are representations of training texts and a group is a tag that need to tag to texts with. [7]

1.4.8.4.3 Deep Learning

Deep learning is a set of algorithms and techniques inspired by how the human brain works. Text classification has benefited from the recent resurgence of deep learning architectures due to their potential to reach high accuracy with less need of engineered features. The two main deep learning architectures used in text classification are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). [8]

On the one hand, deep learning algorithms require much more training data than traditional machine learning algorithms, i.e. at least millions of tagged examples. On the other hand, traditional machine learning algorithms such as SVM and NB reach a certain threshold where adding more training data doesn't improve their accuracy. [18]

Deep learning algorithms such as Word2Vec or GloVe are also used in order to obtain better vector representations for words and improve the accuracy of classifiers trained with traditional machine learning algorithms. [2]

CHAPTER TWO

LITERATURE REVIEW

Literature review describes the previous works and findings related to the field of study. This chapter is dedicated to the description of some feature extraction algorithm, which are relevant to this work as well as the classification techniques used in text classification technique after feature extraction.

Various techniques have been proposed to extract the features of the text: TF-IDF [1], CBOW [2, 3], Skip-gram [2, 3], Feature Extraction based on Cosine Similarity [1], Artificial Neural Network [29], and Convolutional Neural Network [31], and Neuro-Fuzzy Method [30]. The basic concept of these techniques is to extract the features of text and then use these features for different purposes such as in classification.

In [1] TF-IDF term weighting was used to extract features. Selecting relevant features and determining how to encode them for a learning machine method have a vast impact on the learning machine methods ability to extract a good model. Two different weighting methods (TF-IDF and TF-IDF Global) were used and tested on the Reuters-21578 text categorization test collection. The obtained results emerged a good candidate for enhancing the performance of English topics FE. Simulation results the Reuters-21578 text categorization showed the superiority of the proposed algorithm.

In [2] proposed two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations was measured in a word similarity task, and the results were compared to the previously best performing techniques based on different types of neural networks. Large improvements was observed in accuracy at much lower computational cost, i.e. it took less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, these vectors provided state-of-the-art performance on test set for measuring syntactic and semantic word similarities.

In [3] presented several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words significant speedup was obtain and also learned more regular word representations. A simple alternative to the hierarchical softmax called negative sampling was also described. An inherent limitation of word representations was their indifference to word order and their inability to represent

idiomatic phrases. For example, the meanings of “Canada” and “Air” cannot be easily combined to obtain “Air Canada”. Motivated by this example, a simple method for finding phrases in text was presented, and showed that learning good vector representations for millions of phrases was possible.

In [29] proposed a number of networks and learning algorithms which provide new or alternative tools for feature extraction and data projection. These networks include a network (SAMANN) for Sammon’s nonlinear projection, a linear discriminant analysis (LDA) network, a nonlinear discriminant analysis (NDA) network, and a network for nonlinear projection (NP-SOM) based on Kohonen’s self-organizing map [32]. They evaluated five representative neural networks for feature extraction and data projection based on a visual judgment of the two-dimensional projection maps and three quantitative criteria on eight data sets with various properties. Sammon [33] proposed a nonlinear projection technique that attempts to maximally preserve all the inter pattern distances.

In [31] demonstrated a way of formulating a neuro-fuzzy approach for feature extraction under unsupervised training. A fuzzy feature evaluation index for a set of features is newly defined in terms of degree of similarity between two patterns in both the original and transformed feature spaces. A layered network is designed for performing the task of minimization of the evaluation index through unsupervised learning process. This extracts a set of optimum transformed features, by projecting n-dimensional original space directly to n-dimensional (n:n) transformed space, along with their relative importance. This method gave better results than PCA [5].

In [30] demonstrated that one can apply deep learning to text understanding from character level inputs all the way up to abstract text concepts, using temporal convolutional networks (ConvNets). They applied ConvNets to various large-scale datasets, including ontology classification, sentiment analysis, and text categorization. They showed that temporal ConvNets can achieve astonishing performance without the knowledge of words, phrases, sentences and any other syntactic

In [7] explored the use of hierarchical structure for classifying a large, heterogeneous collection of Amharic News Text. The approach utilized the hierarchical topic structure to decompose the classification task into a set of simpler problems, one at each node in the classification tree. An experiment had been conducted using a categorical data collected

from Ethiopian News Agency (ENA) using SVM to see the performances of the hierarchical classifiers on Amharic News Text. The findings of the experiment showed the accuracy of flat classification decreases as the number of classes and documents (features) increases. Moreover, the accuracy of the flat classifier decreases at an increasing number of top feature set. The peak accuracy of the flat classifier was 68.84 % when the top 3 features were used. The findings of the experiment done using hierarchical classification show an increasing performance of the classifiers as we move down the hierarchy. The maximum accuracy achieved was 90.37% at level-3(last level) of the category tree. Moreover, the accuracy of the hierarchical classifiers increases at an increasing number of top feature set compared to the flat classifier. The peak accuracy was 89.06% using level three classifier when the top 15 features were used. Furthermore, the performance between flat classifier and hierarchical classifiers are compared using the same test data. Thus, it shows that use of the hierarchical structure during classification has resulted in a significant improvement of 29.42 % in exact match precision when compared with a flat classifier.

In [27] proposed a way to find co-occurrence feature from anchor text of wikipedia pages, proposed a way to incorporate co-occurrence feature to BOW model. Finally the method was analyzed to know how it performs in task of text classification.

In the context of Nepali text, there is little literature available. The basic steps in natural language processing pipeline such as part of speech tagging [12, 13], steaming [14], named-entity recognition [15, 16] have been investigated separately. There is no result available collectively.

Furthermore, the text classification problem for Nepali text is not even studied thoroughly. In [17], authors have proposed simple naïve Bayes classifier to address the problem. Naïve Bayes uses the concept of probability. The parameter in Naïve Bayes was learned from training the module with the Bayesian rule of probability. The representation of text document in the form of the bag of words where it is assumed that each word is independent of other, mainly degrade the performance of this approach. The simple Naïve Bayes was augmented with multinomial lexicon pooling [17]. Paper [18] applied some machine learning strategies for addressing the classification problem of the Nepali documents.

Nepali SMS classification task is discussed in [19] with Naïve Bayes and support vector machine approaches. Here the length of SMS is very limited in comparison to the length of the full text, the number of features such as SMS headings, words, and their frequency was taken to represent an SMS. The SVM and Naïve Bayes based classification technique were implemented to classify SMS into either spam or not spam. Due to the few number of the feature in consideration, the Naïve Bayes outperform the SVM with 5% (92% accuracy for NB and 87% accuracy for SVM)

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Data Set Preparation

The Nepali language belongs to one of the most common scripts, Devanagari, invented by Brahmins around the 11th century. It consists of 36 consonant symbols, 12 vowel symbols and 10 numeral symbols along with different modifiers and half forms. According to current census research, 17 million people worldwide speak the Nepali language. Nepali language character set is given in Table1.

Table 3. 1 Nepali Character Set

a) Numbers

०	१	२	३	४	५	६	७	८	९

b) Vowels

अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	अं	अः

c) Consonants

क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट	ठ
ड	ढ	ण	त	थ	द	ध	न	प	फ	ब	भ
म	य	र	ल	व	श	ष	स	ह	क्ष	त्र	ज्ञ

A collection of Nepali news was collected from various online Nepali News portals using web crawler. The news portal namely ratopati.com, setopati.com, onlinekhabar.com, and ekantipur.com were used to gather text related to different news types. The distribution of news type in the Nepali news corpus is as shown in Table 2.

Table 3. 2 Statistics of Nepali News Corpus

S.N.	News Class	No. of Documents
1	Agriculture	200
2	Automobile	246
3	Bank	617
4	Blog	259
5	Business	307
6	Economy	600
7	Education	185
8	Employment	304
9	Entertainment	634
10	Health	180
11	Interview	330
12	Literature	251
13	Migration	111
14	Opinion	500
15	Politics	550
16	Society	353
17	Sports	700
18	Technology	118
19	Tourism	265
20	World	313
Total		**7023**

3.2 Preprocessing

The text preprocessing cleans the text data to make it ready to use in training and testing of the machine learning model. Preprocessing is done to reduce the noise in the text that helps to improve the performance of the classifier and speed up the classification process, thus aiding in real time news classification. The main preprocessing techniques used are given below.

1. **Tokenization:** Breakdowns the text into sentences and then words. Vertical bar, question mark, and full stop are used to break down the sentences and whitespace and comma are used to break down the words.
2. **Special symbol and number removal:** Special symbols like !, :, ÷, ×, º, >, <, \, /, @, #, $, %, ^, &, *,), (, _, -, +, =, ~, ø, [,], ‘, ”, etc. and numbers, those do not have much importance in classification, are removed.
3. **Stop word removal:** Stop words are high-frequency words that has not much influence in the text are removed to increase the performance of the classification. The list of 255 stop-words like “छ, म, हो, केही, हामी, मेरो, त्यो, हरु, फेरी, आफू, हुन्छ, राख, भयो, गर्नु,पनि, etc.” were collected and removed from the text.
4. **Word Stemming:** Stemming is used to reduce the given word into its stem. Since the word stem reflects the meaning of a particular word, we have segmented the inflected word and derivational word into a stem word so that the dimension of vocabulary reduced in the significant manner [11]. Example:

नेपाल टेलिकमले दुर्गम हिमाली जिल्लाहरुमा सेवा उपलब्ध गराउनका लागि एनटीस्यट प्रविधिको प्रयोग गरिरहेको छ ।

[‘नेपाल’, ‘टेलिकमले’, ‘दुर्गम’, ‘हिमाली’, ‘ जिल्लाहरुमा’, ‘सेवा’, ‘उपलब्ध’, ‘गराउनका’, ‘लागि’, ‘एनटीस्यट’, ‘प्रविधिको’, ‘प्रयोग’, ‘गरिरहेको’, ‘छ’]

3.3 Feature Extraction

Feature Vector construction is the process of representing the news into a vector form. To represent Nepali news in vector form, the TF-IDF weighting value for each word in the text is taken as a dimensional value in a vector, CBOW and Skip-Gram techniques are used. They are calculated the vector as,

3.3.1 TF-IDF Vectorization

In information retrieval or text mining, the term frequency – inverse document frequency (also called TF-IDF), is a well know method to evaluate how important is a word in a document.

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

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. The TF-IDF weighting of each news is calculated in the feature extraction procedure of the framework.

3.3.2 Skip-Gram Model

In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words. [2, 3]

The skip-gram model actually learns two separate embedding for each word w: the word embedding v and the context embedding C. These embedding are encoded in two matrices, the word matrix W and the context matrix C. Each row i of the word matrix W is the 1xd vector embedding v_i for word i in the vocabulary. Each column i of the context matrix C is a d x1 vector embedding c_i for word i in the vocabulary. In principle, the word matrix and the context matrix could use different vocabularies Vw and Vc.

Consider the corpus of length T and currently pointing at the t^{th} word w (t), whose index in the vocabulary is j, so call it w_j ($1 < j < |V|$). The skip-gram model predicts each neighboring word in a context window of 2L words from the current word. So a context window L=2 the context is [w_{t-2}, w_{t-1}, w_{t+1} w_{t+1}] and predicting each of these from word w_j. To normalize the 2L context words, for example w (t+1), whose index in the vocabulary is k ($1 < k < |V|$). Hence the probability $P(w_k/w_j)$ is calculated. It is defined as follows:

$$p(w_k|w_j) = \frac{exp(c_k \cdot v_j)}{\sum_{i \in |V|} exp(c_i \cdot v_j)}$$

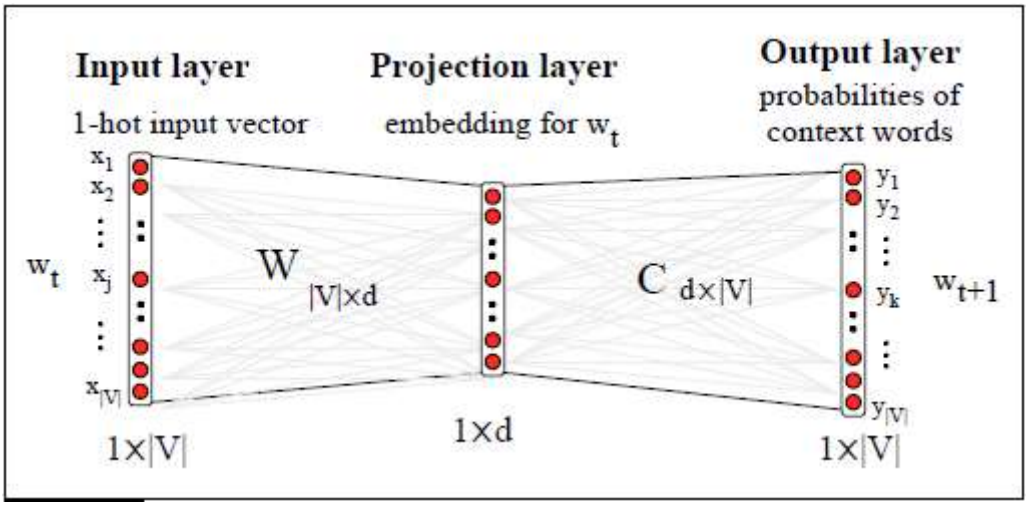

Figure 3. 1 The skip-gram model viewed as a network [2]

3.3.3 CBOW (Continuous Bag of Words)

In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. [2, 3]

Like skip-grams, it is based on a predictive model, but this time predicting the current word w_t from the context window of 2L words around it, e.g. for L = 2 the context is [w_{t-2}, w_{t-1}, w_{t+1} w_{t+1}].

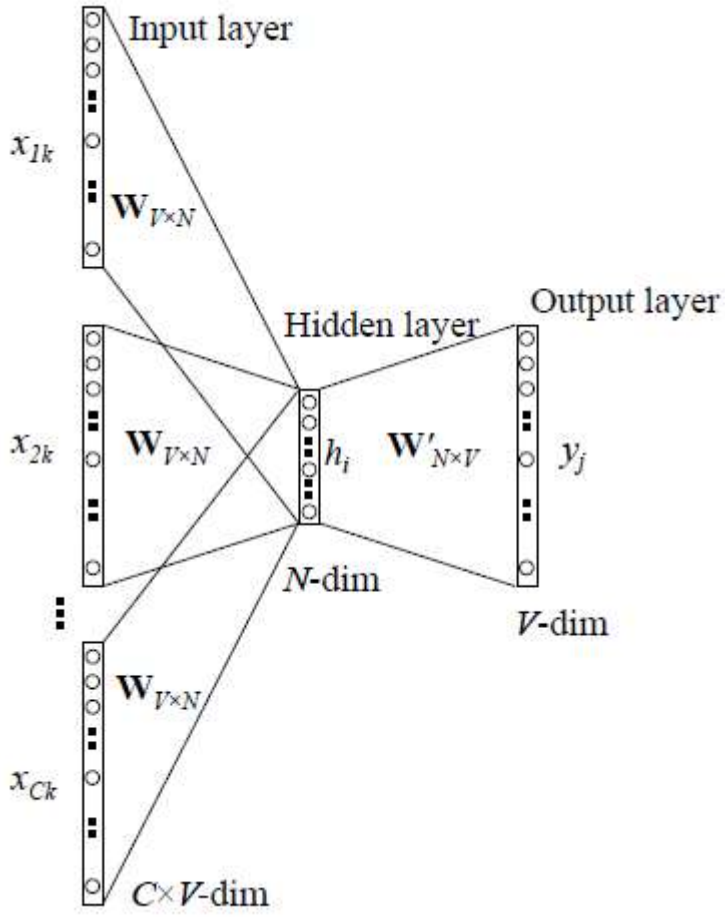

Figure 3. 2 Continuous bag-of-word model [2]

3.4 Classification

Given the example data {(x_i, y_i), i=1….n}, where the xi is input vector and the y_i is its associated label or class. Then the classification task is to learn the discriminate function

$$y=f(x),$$

which correctly classify the example data and optimized so that it will make minimal error on the classification of unseen data.

If the label y is not discrete as above, then this task is called regression. Based on these examples (xi, yi), one is particularly interested to predict the answer for other cases before they are explicitly observed. Hence, learning is not only a question of remembering but also of generalization to unseen cases. [4]

3.4.1 Support Vector Machine (SVM)

This is the supervised machine learning approach that can be used for both classification [4correctly separate the example data into two classes. This hyperplane can be used to make the prediction of class for unseen data. The hyperplane exist for the linearly separable data. [7]

This can be illustrated with figure

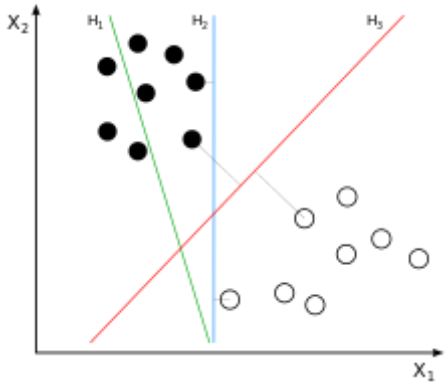

Figure 3. 3 Support Vector Machine [18]

The equation for general hyperplane can be written as

$$w.\,x - b = 0 \qquad \text{(Equation 3.1)}$$

Where x is point vector, w is a weight vector and b is bias. The hyperplane should separate training data $\{(x_i,y_i)$, i=1….n and $y_i \in (+1,-1)\}$ in such way that $y_i\, w.\, x_i - b \geq 1$. The two plane H1 and H2 are supporting hyperplane. We can see that there exist so many hyperplanes that can separate the training data correctly but the SVM find one hyperplane

that maximize the margin between two supporting hyperplanes. It finds the w and b such that the distance (margin) between H1 and H2 is maximum. This can be formulated as optimization problem as

Minimize f = $|w|$ (Equation 3.2)

Subject to constraints $yi\ w.\ xi - b \geq 1$

This can be solved by the variant of quadratic programming technique [13]

3.5 News Type Filtering

The news classification learning and evaluation system pipeline is given in Figure below. It consists of Preprocessing, Feature Extraction, Classification and Evaluation Phases. The complete news dataset that was used in the system training and evaluation was explicitly divided into training and testing sets. The five experiments were conducted to make the more accurate analysis of the outputs. The experimental parameters such as C and gamma (γ) for SVM were determined for their optimal results/outputs. In each experiment, the different optimization parameters were analyzed to reach the optimal output.

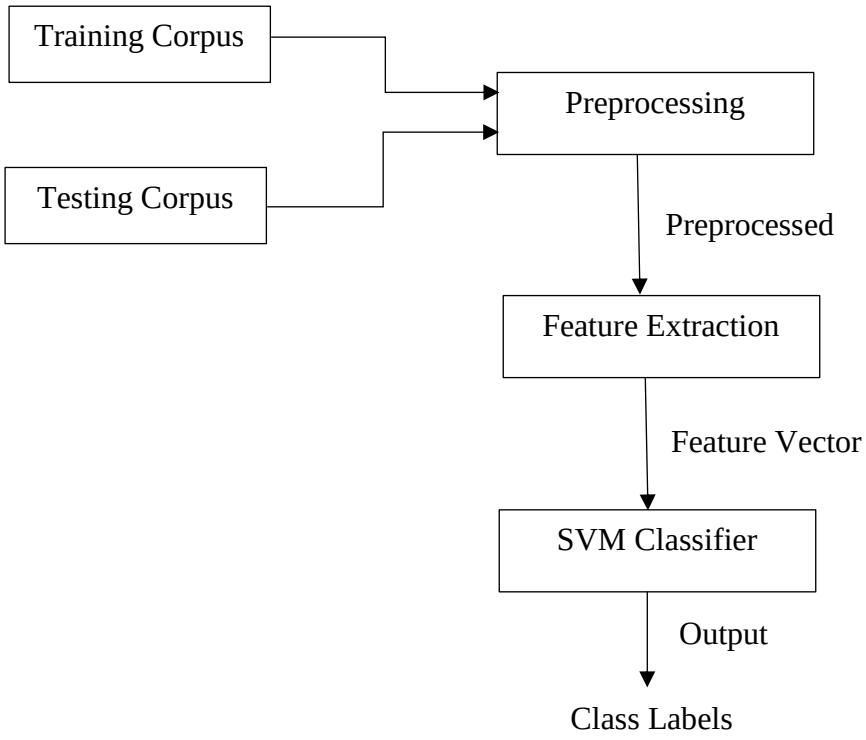

Figure 3. 4 Framework of Classification algorithm

CHAPTER FOUR

IMPLEMENTATION

4.1 Preprocessing

The text preprocessing cleans the text data to make it ready to use in training and testing of the machine learning model. Preprocessing is done to reduce the noise in the text that helps to improve the performance of the classifier and speed up the classification process, thus aiding in real time news classification. Algorithm for preprocessing the text:

1. Obtain a list of all row new text from different newspapers
2. Tokenize the text
3. Remove all stop words and special symbols

4.2 TF-IDF Algorithm

The TF-IDF feature extractor works on the basis of the token frequencies it is fed. The algorithm with which it was implemented alongside the SVM classifier is as follows:

1. Process dataset to obtain IDF for each words
2. Compute TF of each term
3. Obtain a TF-IDF vector representation of text by multiplying corresponding TF and IDF

4.3 Continuous Bag of Words (CBOW) Algorithm

In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context

words more heavily than more distant context words. The algorithm for CBOW is as follows: [2, 3]

1. Get the preprocessed text data and Generate Train set
2. Create Dictionary of words in the corpus
3. Assign unique number to each word
4. Create the array of sentences
5. Arrange the input and output word pair according to window size for training
6. Create one hot encoding for each word.
7. Convert one hot encoding of words into Numpy arrays and place them in X_train and Y_train variables
8. Creating placeholders for X_train and Y_train
9. Define Word embedding dimension
10. Compute Hidden layer
11. Compute Output layer
12. Compute loss function: cross entropy
13. Perform Training operation
14. Now use the hidden layer as lookup table to compute the weight of each words.
15. Print the weight vector

4.4 Skip-gram Algorithm

In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words. The algorithm for CBOW is as follows: [2, 3]

1. Get the preprocessed text data and Generate Train set
2. Create Dictionary of words in the corpus
3. Assign unique number to each word
4. Create the array of sentences
5. Arrange the input and output word pair according to window size for training

6. Create one hot encoding for each word.
7. Convert one hot encoding of words into Numpy arrays and place them in X_train and Y_train variables
8. Creating placeholders for X_train and Y_train
9. Define Word embedding dimension
10. Compute Hidden layer
11. Compute Output layer
12. Compute loss function: cross entropy
13. Perform Training operation
14. Now use the hidden layer as lookup table to compute the weight of each words.
15. Print the weight vector

4.5 Support Vector Machine Classifier

The matrix acquired from the feature extractor is finally used by the SVM classifier. It associates the matrix with the data label, or category which the text was from, and finally a trained classifier is achieved. This trained classifier is later used for predicting the category of unknown texts. Algorithm for training the classifier:

1. Obtain a list of labeled documents to be used for training
2. Perform feature extraction on each document to obtain a feature matrix
3. Compute the corresponding output matrix using document label
4. Use the feature matrix and output matrix to train the SVM
5. Perform hyper parameter optimization

Algorithm for text categorization:

1. Perform feature extraction to obtain a feature vector
2. Feed corresponding vector into the trained SVM

4.6 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

4.7 Tools and Libraries

4.7.1 Anaconda

Anaconda is one of several Python distributions. Python distributions provide the Python interpreter, together with a list of Python packages and sometimes other related tools, such as editors. The packages provide by the Anaconda Python distribution includes all of those that we need. A key part of the Anaconda Python distribution is Spyder, an interactive development environment for Python, including an editor.

4.7.2 Spyder

Spyder is s a powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features. The name SPYDER derives from "Scientific PYthon Development EnviRonment" (SPYDER). It can be used as main environment to learn about Python, programming and computational science and engineering. Useful features include

- provision of the IPython (Qt) console as an interactive prompt, which can display plots inline
- ability to execute snippets of code from the editor in the console
- continuous parsing of files in editor, and provision of visual warnings about potential errors
- step-by-step execution
- variable explorer

4.7.3 TensorFlow

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

4.7.4 Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

4.7.5 Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal. Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

4.7.6 Scikit-learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4.7.7 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy

things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

4.7.8 Seaborn

Seaborn is a Python data visualization library based on mat-plotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

CHAPTER FIVE

RESULT AND ANALYSIS

5.1 Testing Results

The experimental setup is done in five exclusive observations with variation in train and test splits and number of category. Experiments 1 to 5 (aka. Exp1 to Exp5) are respectively trained with 10%, 20%, 30%, 40% and 50 % test splits of total training dataset described in Table 3.2. The experiment is also compared in splitting number of classes in the same corpus. The number of features used in the experiments is 14332, which is equal to the vocabulary size of the dataset.

The experimental result was analyzed for four evaluation parameters: Accuracy, Precision, Recall and F score.

5.1.1 Observation with TF-IDF

The experiments have been first done on a model which uses TF-IDF method as its feature extractor. In its bare-bone form, TF-IDF uses all stems in the vocabulary for feature extraction. Figure 5.1 shows the variation of accuracy, precision, recall and F-Score over 5 experiments. Figure 5.2 shows the variation of accuracy, precision, recall and F-Score over 5 splits of classes. The fluctuation can be attributed to the training data used and the optimum value of ‘C’ calculated by hyperopt during each experiment.

Table 5. 1 Performance of TF-IDF with 20 classes

Experiment	Accuracy	Precision	Recall	F-score
Exp1	**82.91**	**83**	**83**	**83**
Exp2	82.05	82	78	82
Exp3	80.23	80	80	80
Exp4	79.07	79	79	79
Exp5	77.59	78	78	77

Table 5. 2 Performance of TF-IDF with split of classes

No. of classes	Accuracy	Precision	Recall	F-score
2	**100**	**100**	**100**	**100**
5	95.28	95	95	95
10	86.92	87	87	87
15	84.20	84	84	84
20	82.91	83	83	83

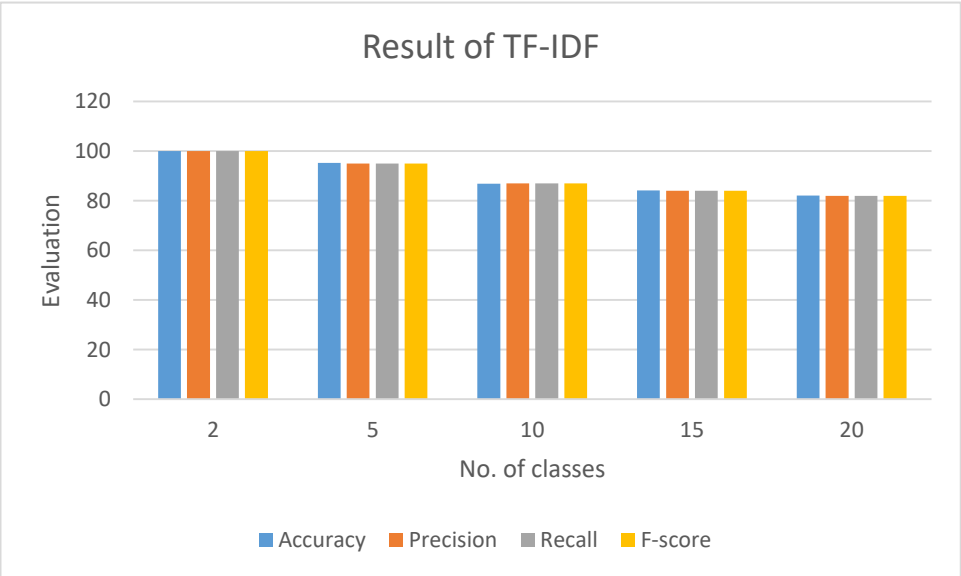

Figure 5. 1 Bar graph of Performance of TF-IDF with of classes

5.1.2 Observation with CBOW

By being fundamentally different to TF-IDF in its approach, CBOW finds the center context word from different words. Figure 5.3 shows the variation of accuracy, precision, recall and f-score over 5 experiments. Figure 5.4 shows the variation of accuracy, precision, recall and f-score over 5 splits of classes. The rise and fall of the values can be attributed to the dataset and the optimum value of 'C' calculated by hyperopt during each experiment.

Table 5. 3 Performance of CBOW with 20 classes

Experiment	Accuracy	Precision	Recall	F-score
Exp1	**40.69**	**36**	**41**	**35**
Exp2	38.86	35	39	33
Exp3	36.21	33	36	30
Exp4	34.50	32	35	28
Exp5	32.94	31	33	26

Table 5. 4 Performance of CBOW with split of classes

No. of classes	Accuracy	Precision	Recall	F-score
2	**75.00**	**76**	**76**	**75**
5	58.52	46	52	45
10	42.88	32	43	34
15	41.52	36	42	35
20	36.37	35	36	30

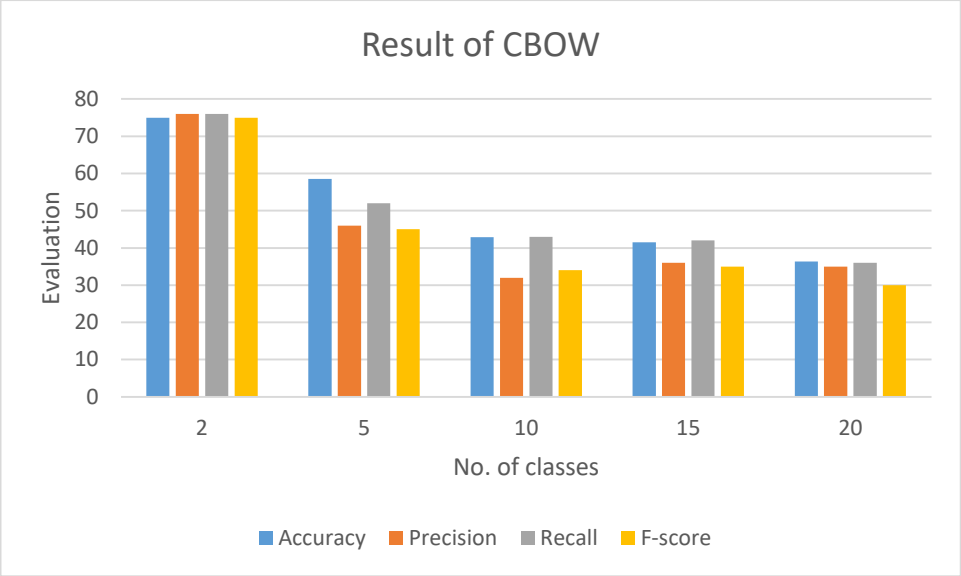

Figure 5. 2 Bar graph of Performance of CBOW with split of classes

5.1.3 Observation with Skip-gram

Like CBOW, Skip-gram finds the context words form center word from different words. Figure 5.5 shows the variation of accuracy, precision, recall and f-score over 5 experiments. Figure 5.6 shows the variation of accuracy, precision, recall and f-score over 5 splits of classes. The rise and fall of the values can be attributed to the dataset and the optimum value of 'C' calculated by hyperopt during each experiment.

Table 5. 5 Performance of Skip-gram with 20 classes

Experiment	Accuracy	Precision	Recall	F-score
Exp1	**33.39**	**29**	**33**	**26**
Exp2	32.05	32	33	25
Exp3	31.67	28	32	26
Exp4	29.02	30	30	25
Exp5	29.27	16	29	19

Table 5. 6 Performance of Skip-gram with split of classes

No. of classes	Accuracy	Precision	Recall	F-score
2	**53.84**	**53**	**54**	**53**
5	48.18	29	48	36
10	37.15	28	37	25
15	32.73	23	33	22
20	29.27	16	29	19

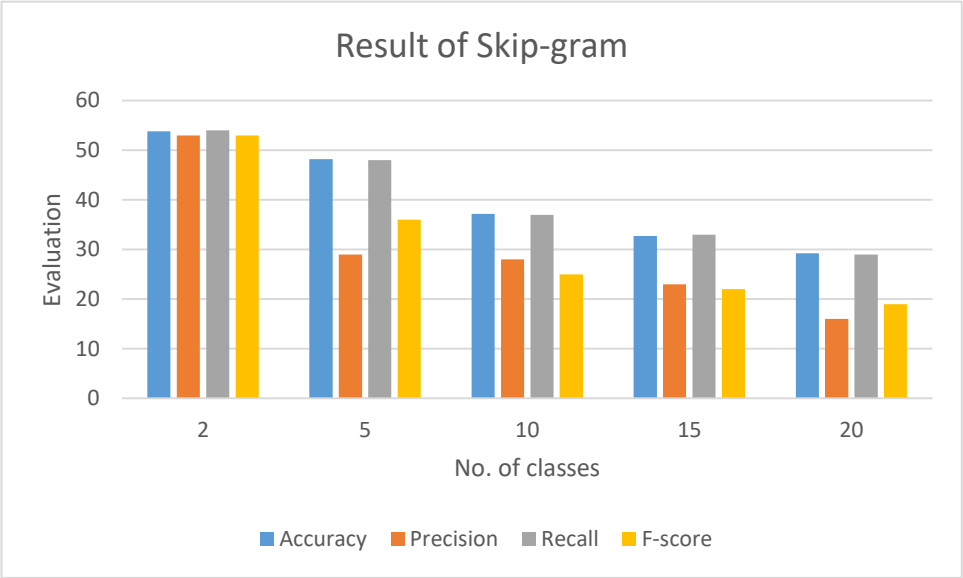

Figure 5. 3 Bar graph of Performance of Skip-gram with split of classes

5.1.4 Overall performance of Feature extraction algorithms

Table 5.7 shows the performance values that were obtained from all three models. Figure 5.4 is a graphical representation of the table that shows the variation of the performance measures. It indicates that the model with TF-IDF implementation has the highest accuracy value, and hence, the best overall performance. The first model features a simplistic approach containing only the TF-IDF feature extractor along with SVM. The second model uses the CBOW for feature extraction. The third model employs the same classification algorithm, SVM, but uses skip-gram feature extraction method.

Table 5. 7 Performance comparison of algorithms

Feature Extraction Algorithm	Accuracy	Precision	Recall	F-score
TF-IDF	**82.91**	**83**	**83**	**83**
CBOW	40.69	36	41	35
Skip-gram	33.39	29	33	26

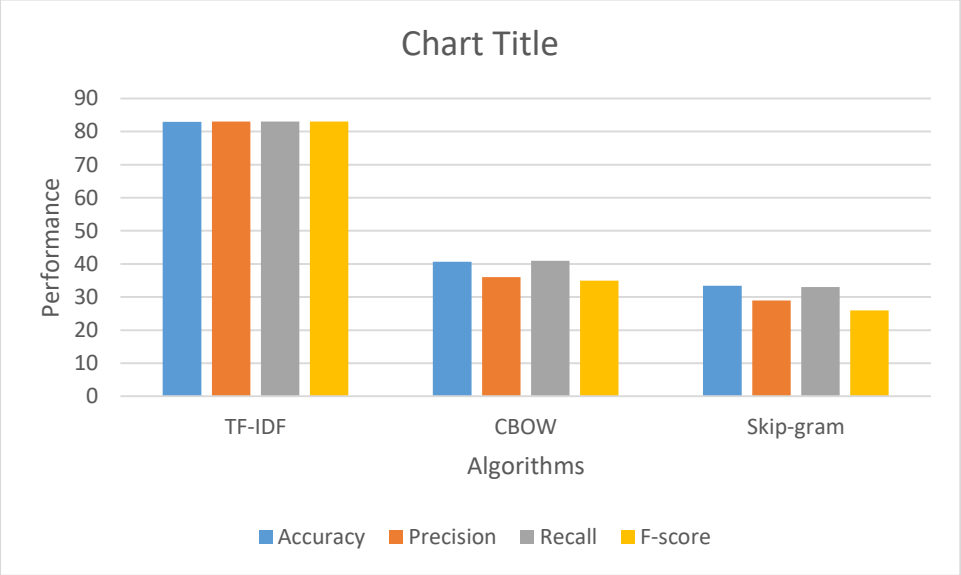

Figure 5. 4 Bar graph of performance comparison of algorithms (TF-IDF, CBOW, and Skip-gram)

5.2 Analysis

5.2.1 Accuracy and F-Score

As figure 7 suggests, the mean accuracy of the 3 models are 82.91 per cent, 40.69 per cent and 33.39 per cent while their F-score are 83 per cent, 36 per cent and 29 per cent respectively. The measure of the classifier's accuracy doesn't always convey the efficiency of the model's performance satisfactorily. It is easily skewed by the unevenness of the data distribution among the categories. However, a better F-score which is the harmonic mean of precision and recall of the model is indicative of the fact that the model is more precise and has a complete prediction ability.

5.2.2 Precession and recall

Figure 5.4 shows that the SVM model with TF-IDF has a better precision and recall over the model with CBOW only by 47 percent and 42 percent respectively. Similarly, the model has a better precision and recall of 54 percent and 50 percent respectively over the model with Skip-gram. The proposed model with TF-IDF has overall performance better than word2vec model.

CHAPTER SIX

CONCLUSION

6.1 Conclusion

Count based feature extraction methods is compared with word to vector feature extraction techniques for Nepali news classification. The results show good classification performance when using the feature extraction techniques based on word to vector for less number of classes and drastically decrease the performance for large sample size. On the other hand result of classification count based technique shows consistent nearly performance for any number of classes. The overall performance of the TF-IDF is far better than both word to vector techniques.

A great deal of research remains in developing document to vector representation of the document. New approaches to setting appropriate category thresholds, estimating probabilities, and selecting features need to be investigated. For practical systems, combinations of count based and word to vector approaches are likely to be the best strategy.

The limitation of the word to vector based feature extraction is speed and only analyze the context. It may be better of grammatical checking in text documents. The other disadvantage of this technique is speed since it based on neural network so it needs to be train before using it. It is better of similarity analysis but text categorization the count based technique better since it counts the number of times word occurrence in the particular document.

References

[1] A. I. Kadhim, Y.-N. Cheah, N. H. Ahamed and L. A. Salman, "Feature Extraction for Co-Occurrence-Based Cosine Similarity Score of Text Document," *IEEE,* 16-17 Dec 2014.

[2] T. Mikolov, . K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *Proceedings of the International Conference on Learning Representations (ICLR 2013),* pp. 1-12, 7 Sep 2013.

[3] W. W. Cohen, "Learning rules that classify e-mail," *in AAAI spring symposium,* vol. 18, p. 25, 1996.

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *Proceeding NIPS'13 Proceedings of the 26th International Conference on Neural Information,* vol. 2, pp. 3111-3119, 2013.

[5] I. Stuart, S.-H. Cha and C. Tappert, "A Neural Network Classifier for Junk E-Mail," *Document Analysis Systems,* vol. 4, p. 442–450, 2004.

[6] X. Carreras and L. Marquez, "Boosting Trees for Anti□Spam Email Filtering," *arXiv preprint cs/0109015,* 2001.

[7] A. K. Tegegnie, A. N. Tarekegn and T. A. Alemu, "A comparative study of flat and hierarchical classification for amharic news text using svm," *I.J. Information Engineering and Electronic Business,* p. 1, 2017.

[8] S. Lai, L. Xu, K. Liu and J. Zhao, "Recurrent Convolutional Neural Networks for Text Classification," *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence,* vol. 333, pp. 2267-2273, 2015.

[9] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *arXiv preprint arXiv:1607.01759,* 2016.

[10] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber and L. E. Barnes, "HDLTex: Hierarchical Deep Learning for Text Classification," *arXiv preprint arXiv:1709.08267,* 2017.

[11] M. HUGHES, I. LI, S. KOTOULAS and T. SUZUMURA, "Medical text classification using convolutional neural networks," *arXiv preprint arXiv:1704.06841,* 2017.

[12] T. B. Shahi, T. N. Dhamala and B. Balami, "Support Vector Machines based Part of Speech Tagging for Nepali Text," *International Journal of Computer Applications (0975 – 8887),* vol. 70, 24, May 2013.

[13] A. Paul, B. S. Purkayastha and S. i. Sarkar, "Hidden Markov Model Based Part of Speech Tagging for Nepali Language," *International Symposium on Advanced Computing and Communication (lSACC), International Symposium on. IEEE,* p. 149–156, 2015.

[14] I. Shrestha and S. S. Dhakal, "A New Stemmer For Nepali Language," *PAN Localization, Working Papers,* vol. 2007, p. 324–31, 2004.

[15] S. B. Bam and T. B. Shahi, "Named Entity Recognition for Nepali Text Using Support Vector Machines," *Intelligent Information Management,* vol. 6, p. 21, 2014.

[16] A. Dey, A. Paul and B. S. Purkayastha, "Named Entity Recognition for Nepali language: A Semi Hybrid Approach," *International Journal of Engineering and Innovative Technology (IJEIT),* vol. 3, no. 8, p. 21–25, February 2014.

[17] S. K. Thakur and V. K. Singh, "A Lexicon Pool Augmented Naive Bayes Classifier for Nepali Text," *Contemporary Computing (IC3), 2014 Seventh International Conference on IEEE,* p. 542–546, 2014.

[18] K. Kafle, D. Sharma, A. Subedi and A. K. Timalsina, "Improving Nepali Document Classification by Neural Network," *Proceedings of IOE Graduate Conference,* p. 317–322, 2016.

[19] T. B. Shahi and A. Yadav, "Mobile SMS Spam Filtering for Nepali Text Using Naïve Bayesian and Support Vector Machine," *International Journal of Intelligence Science,* vol. 4, no. 01, p. 24, January 2014.

[20] C. D. Manning, P. Raghavan and H. Schütze, in *Introduction to information retrieval*, Cambridge University Press, 2008, p. 405–416..

[21] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning,* vol. 20, no. 3, p. 273–297, 1995.

[22] S. Haykin, Neural networks: a comprehensive foundation, Tsinghua University Press, 2001.

[23] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling,* vol. 5, no. 3, p. 1, 1988.

[24] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[25] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management, vol. 45, no. 4, pp. 427–437, 2009.,* vol. 45, no. 4, p. 427–437, 2009.

[26] R. E. Bellman, Dynamic Programming, , Princeton, NJ,, Princeton,NJ, USA: Princeton University Press, 1957.

[27] S. G. K and S. Joseph, "Text Classification by Augmenting Bag of Words (BOW) Representation with Co-occurrence Feature," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 16, no. 1, pp. 34-38, Jan. 2014.

[28] X. Wang and K. Paliwal, "Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition," *Pattern Recognition,* 2003.

[29] J. Mao and A. Jain, "Artificial Neural Networks for Feature Extraction and Multivariate Data Projection," *IEEE Transactions on Neural Networks,* vol. 6, no. 2, pp. 296 - 317, March 1995.

[30] X. Zhang, J. Zhao and Y. LeCun, "Character-level Convolutional Networks for Text Classification," *Courant Institute of Mathematical Sciences, New York University,* vol. 3, 4 Apr 2016.

[31] R. K. Dea, J. Basakb and S. K. Pala, "Unsupervised feature extraction using neuro-fuzzy approach," *Fuzzy Sets and Systems 126,* p. 277–291, 2002.

[32] E. Oja, "Neural networks, principal components, and subspaces," *" Int. J. Neural Syst*, vol. 1, pp. 61-68, 1989.

[33] J. Rubner and K. Schulten, "“Development of feature detectors by self-organization," *Biol. Cybern*, vol. 62, pp. 193-199, 1990.

Bibliography

E. Alpaydın, "Introduction to Machine Learning," Second Edition, *The MIT Press, Cambridge, Massachusetts London, England*, 2010

L. Hamel, "Knowledge Discovery with Support Vector Machine," *John Wiley and Sons Inc.*, New Jersey, USA, 2009

Appendix

Sample program Code for CBOW

```
from gensim.models.word2vec import Word2Vec
from sklearn.pipeline import Pipeline
from collections import defaultdict
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from tabulate import tabulate
from sklearn.model_selection import cross_val_score
import codecs
from sklearn.svm import SVC

X, y = [], []
with codecs.open('stop_nepali.txt', 'r', 'utf-8', errors='ignore' ) as infile:
    for line in infile:
        label, text = line.split("\t")
#         print(text)
        # texts are already tokenized, just split on space
        # in a real case we would use e.g. spaCy for tokenization
        # and maybe remove stopwords etc.
        X.append(text.split())
        y.append(label)
#print(X[1])
X, y = np.array(X), np.array(y)
print(X)
print ("total examples %s" % len(y))

#import numpy as np
#with open('glove.txt', "rb") as lines:
#    wvec = {line.split()[0].decode('utf-8'): np.array(line.split()[1:],dtype=np.float32)
#               for line in lines}
#print(wvec)

glove_small = {}
all_words = set(w for words in X for w in words)
with open('glove.txt', "rb") as infile:
    for line in infile:
        parts = line.split()
        word = parts[0].decode('utf-8')
        if (word in all_words):
            nums=np.array(parts[1:], dtype=np.float32)
            glove_small[word] = nums
#print(glove_small)
```

```
model = Word2Vec(X, size=2, window=5, min_count=3, workers=2)
w2v = {w: vec for w, vec in zip(model.wv.index2word, model.wv.vectors)}
#print(w2v)

class MeanEmbeddingVectorizer(object):
    def __init__(self, word2vec):
#       print('This is initialization')
        self.word2vec = word2vec
        if len(word2vec)>0:
            self.dim=len(word2vec[next(iter(w2v))])
#           print(self.dim)
        else:
            self.dim=0

    def fit(self, X, y):
#       print('This is fit funtions')
        return self

    def transform(self, X):
        return np.array([
            np.mean([self.word2vec[w] for w in words if w in self.word2vec]
                    or [np.zeros(self.dim)], axis=0)
            for words in X
        ])


#and a tf-idf version of the same
class TfidfEmbeddingVectorizer(object):
    def __init__(self, word2vec):
        self.word2vec = word2vec
        self.word2weight = None
        if len(word2vec)>0:
            self.dim=len(word2vec[next(iter(w2v))])
#           print(self.dim)
        else:
            self.dim=0

    def fit(self, X, y):
        tfidf = TfidfVectorizer(analyzer=lambda x: x)
        tfidf.fit(X)
        # if a word was never seen - it must be at least as infrequent
        # as any of the known words - so the default idf is the max of
        # known idf's
        max_idf = max(tfidf.idf_)
        self.word2weight = defaultdict(
            lambda: max_idf,
            [(w, tfidf.idf_[i]) for w, i in tfidf.vocabulary_.items()])
```

```
        return self

    def transform(self, X):
        return np.array([
                np.mean([self.word2vec[w] * self.word2weight[w]
                         for w in words if w in self.word2vec] or
                        [np.zeros(self.dim)], axis=0)
                for words in X
            ])

#etree_glove_small = Pipeline([("glove vectorizer",
MeanEmbeddingVectorizer(glove_small)),
#                    ("extra trees", ExtraTreesClassifier(n_estimators=200))])
#etree_glove_small_tfidf = Pipeline([("glove vectorizer",
TfidfEmbeddingVectorizer(glove_small)),
#                    ("extra trees", ExtraTreesClassifier(n_estimators=200))])
#print("classification begin")
etree_w2v = Pipeline([("word2vec vectorizer", MeanEmbeddingVectorizer(w2v)),
                      ("extra trees", SVC(kernel='rbf', random_state=42, verbose=False, C=1.5,
gamma='auto'))])
etree_w2v_tfidf = Pipeline([("word2vec vectorizer", TfidfEmbeddingVectorizer(w2v)),
                      ("extra trees", SVC(kernel='rbf', random_state=42, verbose=False, C=1.5,
gamma='auto'))])
#print("classification end")


all_models = [
    ("w2v", etree_w2v),
    ("w2v_tfidf", etree_w2v_tfidf),
#    ("glove_small", etree_glove_small),
#    ("glove_small_tfidf", etree_glove_small_tfidf)
]
#print(X)
unsorted_scores = [(name, cross_val_score(model, X, y, cv=5).mean()) for name, model
in all_models]
scores = sorted(unsorted_scores, key=lambda x: -x[1])
print (tabulate(scores, floatfmt=".4f", headers=("model", 'score')))
```