Improving Nepali Document Classification by Neural Network

Kaushal Kafle [1], Diwas Sharma [2], Aayush Subedi [3], Arun Kr. Timalsina [4]

[1, 2, 3, 4] *Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering, TU, Nepal*
Corresponding Email: [4] t.arun@ioe.edu.np

Abstract
Text classification is the task of classifying documents into predefined categories automatically. This paper compares different text classification methods based on their effectiveness on the Nepali language.
The lack of a standard Nepali corpus prompted for the creation of a manual data set by crawling various Nepali news sites. Nepali document classification is severely limited by the complexity of the language morphology. Document classification with word2vec employs neural network and simplifies the process of automatically categorizing Nepali documents while increasing the precision and recall over previously implemented techniques such as TF-IDF. Results from 3 models, SVM with TF-IDF only, SVM with word2vec and cosine similarity with TF-IDF and LSI show that the word2vec model outperforms the TF-IDF only method by 1.6 per cent and the cosine similarity with LSI method by 2.2 per cent.

Keywords
Support Vector Machine – Text Classification – Feature extraction – TF-IDF – Word2vec – LSI – Neural Network – Nepali Corpus

Introduction

Automatic document classification has been a computationally challenging part of any language. It is even more challenging to algorithmically classify documents written in a language such as Nepali that is morphologically rich and structurally complex. This is also the primary reason why there is a lack of research done on Nepali language based text classifiers. A job of a text classifier is to automatically assign a given document to its pre-specified category. Documents may be classified based on a number of attributes such as document types (images, videos, texts, etc), their contents (stories, novels, poems, etc), their authors, etc. It has a wide range of applications such as spam filtering, document indexing, text filtering, news organization, etc.

The first problem of text classification is its feature extraction. It consists of acquiring a vector representation of the documents that can be used by the learning algorithm. Two methods, TF-IDF and word2vec [1], have been used for feature extraction in the paper. TF-IDF is a simple method that does not consider the order of the words in a sentence for classification purposes. Word2vec, however, is used when the contexts of the words are also to be taken into consideration. This paper focuses on the effects and efficiency of both the techniques along with their comparison with each other.

Feature extraction using TF-IDF involves the calculation of IDF, which gives a measure of how important a word is to categorize a document. For example, the word 'करोड' generally appears in documents related to business only, so its IDF measure is relatively high. On the other hand, word such as 'यसरी' has a high frequency in all documents so it is not particularly useful to determine which category a document should belong to. Hence, its IDF measure is low.

On the other hand, when the skip-gram architecture for word2vec is given any word, it calculates the probability of other words that occur in its surrounding within a specified window. For example, words such as 'राजधानी' and 'उपत्यका' have a high probability of appearing nearby 'काठमाडौँ', but unrelated words like 'धारा' and 'कमल' have a low probability. This is reflected on the representation of the words by word2vec. The word2vec algorithm in this classification model has been implemented using gensim.[2]

Classification of a document is done using either a supervised or an unsupervised learning method. SVM (Support Vector Machine) is an example of a supervised learning method. It is observed that SVMs consistently achieve good performance on text categorization tasks, outperforming existing methods substantially and significantly [3]. Using one-vs-all technique, SVM is used to build a model that can automatically assign input documents into any one of the categories based on the training data. The performance of the model largely depends on the configuration of the hyper-parameters of the training algorithm. Hyperopt [4], a Python library for serial and parallel optimization, has been employed for calculating the optimal value of the regularization parameter (C) in SVM.

1. Related Works

Primarily, most of the works on text classification were based on English language. Apart from English language, some form of text classification system exists for European languages such as Italian, German, Spanish, etc. and Asian languages such as Arabic, Chinese and Japanese.

Nepali is a morphologically rich language that has a fairly complicated orthography. Due to this, many language features has to be taken into consideration to build an efficient text classification model. Even so, commendable efforts have been made in the field of Nepali text classification using various methods.

Shahi and Yadav analyse the effects of two classification techniques, the Naive Bayes and SVM, to develop a Mobile SMS spam filtering for Nepali text [5].

Dangol and Timalsina implement various Nepali language specific features such as filtering stop-words, word replacements and removal of word suffices using Nepali language morphology to reduce the number of dimensions in Vector Space Model [6].

Similarly, Bam and Shahi classify rigid designators in Nepali text such as proper names, biological species and temporal expressions into some predefined categories which plays an important role in different fields such as Machine translation, Information Extraction, Question Answering System, etc. [7].

Currently, the number of commercial Nepali text classification system that categorizes the given documents into predefined specific categories based on their content is almost non-existent. This could be attributed to the fact that a reliable consistency and accuracy required at a commercial level hasn't been achieved because of the complexity of the Nepali language and also a lack of linguistic resources in Nepali such as a lack of generic stemmer, an accurate POS (Part-Of-Speech) tagger, stop-word filter, etc.

The Nepali stemmer model developed by Bal and Shrestha [8] is used for the purpose of this research.

2. Methodology

2.1 Data Distribution

A new Nepali corpus was built by collecting news documents from various popular Nepali news sites using a web crawler[1] based on Scrapy[2], a popular framework for extracting data from websites.

Table 1: Data distribution

Category	No. of Documents
automobiles	283
finance	837
crime	491
employment	206
entertainment	2068
health	1021
literature	404
politics	2879
society	711
sports	2052
technology	1221
tourism	772
others	488

A total of 13433 documents of different categories were collected from news portals such as Ratopati, Setopati, Onlinekhabar, Nepalipatra, etc. The collected dataset was further filtered and manual categorization was done to introduce more categories. The articles which didn't fall distinctly under any pre-specified category were kept

[1]https://github.com/kad4/crawler
[2]https://github.com/scrapy/scrapy

under the 'other' category during the feature extraction phase in order to improve accuracy of the model. This category was later excluded during the classification stage. The diverse nature of the collected data is shown in Table 1. Finally, the filtered articles in their respective categories were used in the training and validation of the model.

2.2 Stemmer

A stemmer tailored for Nepali texts was used to tokenize the words in the dataset and strip them off of suffixes. These tokens were then subsequently passed to the feature extractors. Example:

दूध बिक्री गरेर गुजारा चलाउँदै आएका यहाँका सर्वसाधारण/कृषक आफैं दुग्ध प्रशोधन कारखाना खोलेर मालिक बन्न थालेका छन् ।

['दूध', 'बिक्री', 'गुजारा', 'यहाँ', 'सर्वसाधारण', 'कृषक', 'आफैं', 'प्रशोधन', 'कारखाना', 'खोल', 'माल', 'छन्']

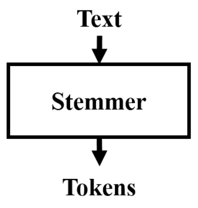

Figure 1: Use of stemmer

2.3 TF-IDF Feature Extractor

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. Preprocess dataset to obtain IDF for top 1000 stems
2. Tokenize and stem given text
3. Compute tf of stems obtained during preprocessing
4. Obtain a tf-idf vector representation of text by multiplying corresponding tf and idf

2.4 Word2Vec Feature Extractor

The word2vec implementation used in this research uses a single hidden layer neural network having 300 neurons. The number of neurons in the input and the output layer is the number of tokens obtained after filtering out tokens having frequency less than 5 in the whole corpus. Following a typical skip-gram model, the word tokens are fed into the input neurons and for each word token, the corresponding set of words within a fixed window of size 10 that appear alongside the word tokens are fed to the output neurons. Tokens with a frequency of less than 5 are ignored. The negative sampling size is also kept at 5. Based on these two inputs, the hidden layer creates a matrix that keeps the information of the context of the words within the fixed window size.

1. Train Neural Network for word2vec using skip-gram model with negative sampling
2. Tokenize and stem given text
3. Obtain the word vectors for individual words
4. Average the word vectors to obtain the vector representation for text

2.5 SVM 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 hyperparameter optimization

Algorithm for text categorization:

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

The flowchart of the overall system can be seen in figures 2 and 3.

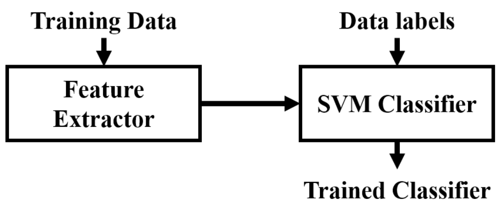

Figure 2: Training of classifier

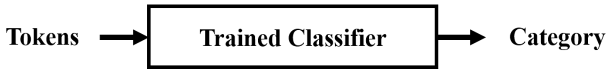

Figure 3: Category prediction using trained classifier

3. Evaluation

3.1 Optimization of Regularization Parameter (C)

The performance of the classification algorithm, SVM, depends on its regularization parameter referred to as 'C'. In order to automatically calculate the optimal value of 'C' in each experiment cases, Hyperopt has been integrated into the training model. Equation 1 is the cost function used by hyperopt to calculate C. Over the course of 10 experiment cases, hyperopt assigns a value for C and calculates the corresponding accuracy of each cases. The value of C that corresponds with the highest accuracy is used as the Regularization parameter C by SVM.

$$loss = 1 - accuracy \tag{1}$$

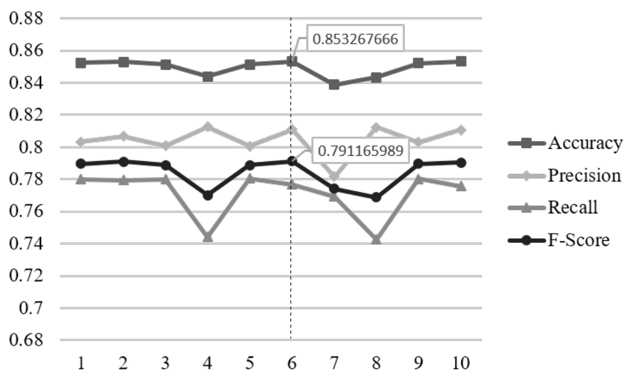

Figure 4: Hyper parameter optimization with hyperopt

3.2 Experiment procedure

Each experiment consists of 10 sub-experiments which the hyperopt uses to obtain the optimal value for the regularization parameter, C. The experiments consisted of 5 fold cross validation in which test data size of 33.33 per cent of the total was used.

3.3 Observation using 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. For optimum results, the model uses the 1000 most frequent stems for feature extraction. Because of this sheer number of stems fed into the SVM, it has a high dimensionality. Figure 5 shows the variation of accuracy, precision, recall and F-Score over 5 experiments. The fluctuation can be attributed to the training data used and the optimum value of 'C' calculated by hyperopt during each experiment.

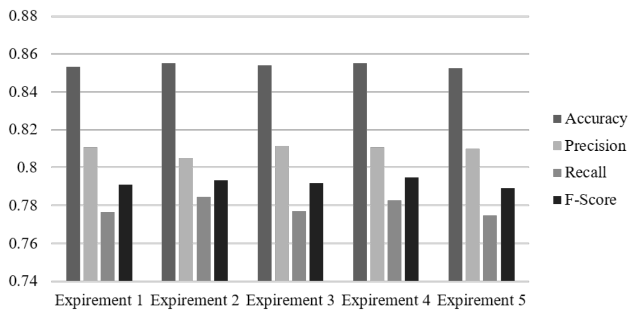

Figure 5: Performance values obtained using TF-IDF

3.4 Observation using Latent Semantic Indexing

Dangol's model [6] uses cosine smilarity with TF-IDF for text classification and applies Latent Semantic Indexing (LSI) for dimensionality reduction. It uses other language specific features such as stopwords-filter, suffix-stripping, etc. to reduce the number of dimensions further. Applying feature reduction techniques on the model helps to improve its efficiency in regards to the time and storage required to train it along with its performance measure.

As a context-counting method, LSI is much more sensitive to the parameter choices [9]. Due to this, it has the tendency to produce much less desirable result when the parameter value is unsuitable. Even though impressive results were obtained using language-specific approaches such as stemmers, stop-word filters, etc. they rely heavily on in-depth knowledge of the dataset, language-specific semantics and performance tuning.

On the other hand, context-predicting method such as word2vec with the help of neural networks tend to be generally less sensitive to the parameter choices [9]. Given any dataset, they can churn out relatively better results when compared to the context-counting methods. The system model employed in this research uses word2vec as the feature extractor to specifically exploit this advantage of the neural networks in text classification.

3.5 Observation using word2vec

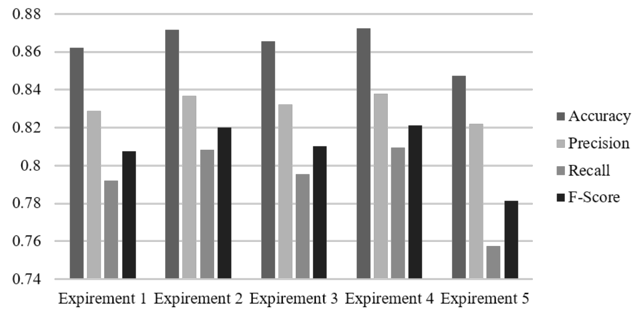

Figure 6: Performance values obtained using word2vec

By being fundamentally different to LSI in its approach, word2vec improves the dimensionality reduction in the training model even more. This saves a large amount of stem overhead and processing time. Figure 6 shows the variation of accuracy, precision, recall and f-score over 5 experiments. 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.

3.6 Comparison of performance across models

Table 2 shows the mean performance values that were obtained from all three models. Figure 7 is a graphical representation of the table that shows the variation of the performance measures. It indicates that the model with word2vec implementation has the highest Fscore value, and hence, the best overall performance.

Table 2: Mean performance values comparison obtained from different methods

Measures	TF-IDF	LSI	word2vec
Accuracy	0.854045276	0.937694	0.863845162
Precision	0.809654627	0.795888	0.831501748
Recall	0.779189265	0.781929	0.792444839
F-Score	0.792013304	0.785624	0.808097445

The first model features a simplistic approach containing only the TF-IDF feature extractor along with SVM. The second model uses the cosine similarity method along with TF-IDF and LSI for feature extraction. The third model employs the same classification algorithm, SVM, but uses a neural network-based feature extraction method word2vec.

As figure 7 suggests, the mean accuracy of the 3 models are 85.4 per cent, 93.7 per cent and 86.4 per cent while their mean F-score are 79.2 per cent, 78.6 per cent and 80.8 per cent respectively.

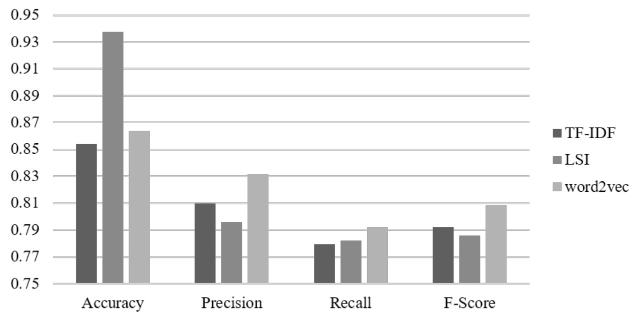

Figure 7: Performance comparison with reference

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. Figure 7 shows that the SVM model with word2vec has a better precision and recall over the model with TF-IDF only by 2.18 per cent 1.33 per cent respectively. Similarly, the model has a better precision and recall of 3.56 per cent and 1.05 per cent respectively over the model with LSI. This results in the F-score of the word2vec model being superior to the TF-IDF only method by 1.6 per cent and the LSI method by 2.2 per cent. The proposed model with word2vec lags behind the LSI model only in terms of accuracy, but has a higher overall precision and recall, leading to a higher F-score.

4. Conclusion

The lack of research works and the complexity of Nepali as a language made it difficult to find reliable linguistic resources that made our task tedious. The nonexistence of a standard Nepali corpus led us to create our own corpus by crawling various Nepali news sites. However, after the completion of the classification model, there were encouraging results that neural network, particularly word2vec, brought to automatic Nepali document classification.

In this paper, parameters such as accuracy, precision, F-score and recall have been used to evaluate the efficiency of all the models. The SVM model with TF-IDF as feature extraction had no integrated dimensionality reduction feauture. The next model uses cosine similarity for text classification, TF-IDF for feature extraction and also adds LSI for dimensionality reduction. However, the third model with SVM and word2vec outperforms each of them in terms of F-score by 1.6 per cent and 2.2 percent respectively. The training model is also greatly simplified by employing a neural network for learning. It is also less sensitive external factors and parameters which helped to maintain a much consistent results of document categorization over all the experiments.

References

[1] T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

[2] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[3] Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

[4] James Bergstra, Daniel Yamins, and David D Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML (1)*, 28:115–123, 2013.

[5] Tej Bahadur Shahi and Abhimanu Yadav. Mobile sms spam filtering for nepali text using naïve bayesian and support vector machine. *International Journal of Intelligence Science*, 4(01):24, 2013.

[6] Dinesh Dangol and Arun K. Timalsina. Effect of nepali language features on nepali news classification using vector space model. 2013.

[7] Surya Bahadur Bam and Tej Bahadur Shahi. Named entity recognition for nepali text using support vector machines. *Intelligent Information Management*, 2014, 2014.

[8] Bal Krishna Bal and Prajol Shrestha. A morphological analyzer and a stemmer for nepali. *PAN Localization, Working Papers*, 2007:324–31, 2004.

[9] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247, 2014.