Proceedings of 12th IOE Graduate Conference
Peer Reviewed
ISSN: 2350-8914 (Online), 2350-8906 (Print)
Year: 2022 Month: October Volume: 12

Efficient Word Embedding for Nepali

Bishal Debb Pande [a], Aman Shakya [b]

[a, b] *Department of Electronics & Computer Engineering, Pulchowk Campus, IOE, Tribhuvan University, Nepal*
[a] 075mscsk005.bishal@pcampus.edu.np, [b] aman.shakya@ioe.edu.np

Abstract
Word embedding are a vital part for most modern Natural Language Processing (NLP) task. It is however difficult to identify if the given word embedding model works well or not. Especially with larger models, the time taken to train such models are very large. When we add to this the time required to train the eventual model for NLP task, a very large chunk of time can be spent on just training the different word embedding models to identify which word embedding models works well. Due to this, selecting between different models of word embedding can be very difficult. For this, intrinsic evaluation is used to evaluate the performance of word embedding systems instead of directly using the model for eventual NLP task. But for Nepali, it is difficult due to the lack of resources in Nepali Language. We show that using intrinsic evaluation based from similar language like Hindi with small modifications, we can gain insight about the effectiveness of word embedding. It can be justified based on the result for extrinsic evaluation where in the results are in agreement with the results from intrinsic evaluation. Using this, we find out that among the 3 models considered, the fasttext model performs the best when considering out of vocabulary words.

Keywords
NLP, Word embedding, word2vec, Fasttext, GLOVE

1. Introduction

Even though there has been significant improvement in Natural Language Processing (NLP) over the years, most of these works have been heavily focused on the English language. Due to this, the current state of other languages is far behind in terms of the progress made. Especially for the Nepali language, there has been very little work done in this field.

One of the major hindrances to the work on NLP for the Nepali language has been the availability of a comprehensive word corpus. With most modern methods being heavily dependent on having a large word corpus to be able to provide good performance this introduced a bottleneck in the work that can be done. Until recently, most of the information present in the language had been in written form only, and to use the prevailing information, a significant effort needed to be made to convert them into a suitable format that can be consumed by the NLP systems. This in turn means that the focus for such work lies on the conversion of the dataset to a consumable format and the actual NLP tasks become secondary. But with the expansion of social media and other digital platforms over the years, the consumption of digital textual data has increased significantly. This pattern has replicated itself in the Nepali language as well. This has meant that there is increased availability of digitized textual content in Nepali as well. These data can be used to expand the work on the Nepali language.

Word embedding provides a representation of words in vector space which can be utilized by other downstream tasks. Most modern techniques for word embedding are based upon the linguistic hypothesis: "A word is characterized by the company it keeps." [1]. It suggests that even though two words might have different syntactic construction their similarity in sense may be captured by the places where they occur within the language. This also implies that instead of needing to have complete information about the construct and meaning of each word, one can get insight into the language by having usage information of words. These representations also have some interesting properties present in them. Similar words tend to lie close together each other in the vector space and form clusters. One other interesting property is shown between analogous words where

simple vector operations can be used to get analogous words.

This means that to be able to work on NLP tasks, it is vital to have a good method for having word embeddings. In the context of NLP in the Nepali language as well, there is a lack of work done in word embedding, and carrying out work on the creation of effective word embedding systems for the Nepali Language is vital. This can also provide a basis for other works to be carried out in the language.

Our contribution in this paper includes an intrinsic analysis of the word embedding model using the similarity set and analysis of the three different models.

2. Related Literature

Word embedding is the method by which the textual data are converted into a format such that they can be used by computer systems. Earlier approaches utilized a sparse representation of words which had difficulty in establishing the semantic relationship between words. Over time, these methods have been replaced by neural approaches which produce a dense representation of words in vector space.

Word2Vec [2] is one of the leading approaches used for word embedding. It consists of two approaches: The Continuous Bag of Words (CBOW) and the skip-gram. The two algorithms are very similar but have slightly different approaches. The CBOW uses the surrounding context words to predict the target word while the skip-gram predicts the context words for the given target word. Loss is calculated in terms of the prediction made and optimization is carried out based on it to provide a better representation. In [3] the authors have tried to provide a deeper intuition into the word2vec model with the mathematical modeling of the model. This tries to explain why the model works and how the underlying representation of the language is captured by the word2vec model.

The Global Vectors for Word Representation (GloVe) [4] is another popular approach used for word embeddings. In it, the word embeddings are created based on the co-occurrence matrix. The co-occurrence matrix is created using the global statistics of text in the given corpus. The advantage it provides over word2vec is in terms of computational complexity, speeding up the process significantly.

The Fasttext model proposed by [5] has a slightly different approach for creating word embedding. It may be considered a variation of the Word2Vec model. In it, instead of using n words around the target word, it uses the character n-gram and vector representation is constructed from them. This allows for the improvement in the performance for out of vocabulary words that might share the same character n-grams as previously trained words.

As word embeddings are extensively used in modern NLP tasks, few of the recent works in the Nepali language have also used word embedding techniques. In [6], the author has released a 300-dimensional word embedding model along with the corpus used for it. Singh et al [7], have utilized the word2vec and Fasttext model in the Nepali language for NER. They have also analyzed the use of a few variations of these word embeddings and how the performance is affected by it. This work also highlights the significance of word embedding and the effect they have on the performance of the NLP system. Koirala et al [8] have carried out an analysis of different Nepali word embeddings using intrinsic and extrinsic analysis. They use clustering based on relatedness and sentiment for intrinsic analysis and news classification for extrinsic analysis. They have also utilized BERT-based models but due to the lower size and the limited data, the performance is not that good.

Similar research has also been carried out in other Asian languages with low resources. In [9] the authors have created the word similarity dataset and carried out the intrinsic evaluation on six different low-resource Indian languages. In [10], the authors have carried out similar research on the Sinhala language and compared the effect of the use of different word embedding and analyzed the performance on both extrinsic and intrinsic tasks.

3. Methodology

Figure 1 below defines the overall processes that are utilized.

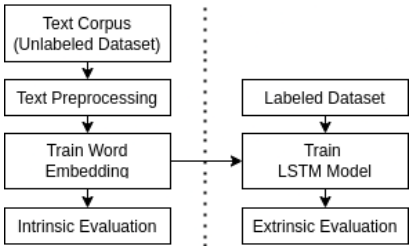

Figure 1: Methodology

3.1 Data Collection

The dataset consists of a collection of multiple publicly available datasets that have been created previously for various tasks. The first publicly available dataset utilized is the Nepali Monolingual dataset [11]. It was produced in 2006 in the framework of the project Bhasha Sansar. It consists of a general corpus which consists of the collection of written text from a wide range of sources such as internet websites, newspapers, books publishers, and authors. This corpus had been created to allow for corpus analysis that requires a very large corpus. This corpus consists of 1,400,00 words. The other publicly available corpus utilized is the Nepali English Parallel Corpus [12]. It consists of the translation of 4325 sentences having over 100,000 words taken from the PENN Treebank corpus. The corpus also has POS tagging applied to it with 43 POS tags. Also, a small dataset consisting of Nepali words along with NER tags is collected from [7].

In addition to these, the dataset also consists of data scraped from various sources such as news sites and Nepali Wikipedia.

3.2 Preprocessing

Preprocessing is one of the vital tasks that need to be carried out. In the first step, the text is separated into sentences. Since the dataset consists of text in other languages as well, such text is removed based on the Unicode range of Nepali words. Also, numbers, punctuation, and other special characters are removed in this step.

In addition to this, the use of stop word removal and stemming of words are carried out. The predefined set of Nepali stop words was taken for this and any words in the set were removed from the sentences. For stemming, the python package from [13] is used which separates postfix based on dictionary look-up.

In each sentence, after initial pre-processing has been carried out, an additional check is done to see the total number of words present in the sentences. Sentences that have 3 or fewer words are truncated.

3.3 Training Word Embedding

The creation of word embedding is carried out in an unsupervised manner. The cleaned corpus is provided from which a dense representation is created. The models for word2vec, GloVe, and Fasttext are considered. The implementation of these is readily available and already existing packages in Python are utilized for creating the word embeddings.

3.3.1 Hyper-parameter tuning

One of the challenges for training a word embedding system is the selection of various parameters and hyperparameters. A few of the major parameters to consider are embedding dimension, window size, and the number of epochs of training. All the models are trained for a selected set of embedding dimensions. It has been generally observed that larger embedding sizes have better performance. This pattern however cannot be observed indefinitely. After a certain embedding size is reached, the performance slowly starts to deteriorate. The major bottleneck for this is the size and the variety of word corpus available. The representation in the higher dimension starts to become sparse in case enough words are not available.

The window size can also affect performance. Window size refers to the number of neighbors of a word that will be considered during the training. A large value generally would mean that more context is generated from the corpus, but the major limitation to this is the larger computational power and the training time required for training a mode with the increase in window size. Each model is trained for a few different values of window size to analyze the effect it has on the performance.

The number of epochs for training is set differently from one model to another. The consideration made is in terms of the computational time required. The number of epochs for each variation has been set such that the training time required will be similar across the models for a given embedding dimension. This is done as computational time is also a major constraint that needs to be considered.

3.4 Performance analysis

The performance evaluation for word embedding can be broadly classified into two categories. They are intrinsic and extrinsic evaluations.

Intrinsic evaluation evaluates the performance of the word embedding on an intermediate task which is different from any specific NLP task. These tasks may or may not individually assure that performance improvement in the given intermediate task necessarily replicates itself with the eventual task where the embedding is to be used. But these tasks help provide a quick and efficient method for analysis

of performance. In addition, these analyses are not as heavily dependent on large sets of labeled datasets. A few of the popular methods are cluster examination, analogous pair finding, and similarity measure for target word pairs.

Extrinsic evaluation evaluates the performance of word embedding based on the performance that they provide to the NLP tasks such as classification, Named Entity Recognition (NER), Part of Speech (POS) tagging, and so on. These are the tasks where one would eventually want to improve the performance through the improvement in the embedding.

Since the time required for extrinsic evaluation is significantly higher, the evaluation based on intrinsic methods is utilized for the initial model evaluation. Among the various intrinsic methods, word similarity measurement for a collection of pairs of words is utilized as they provide a comprehensive measurement of performance. For the final evaluation of the model, extrinsic evaluation is utilized using NER tagging as the reference NLP task.

3.5 Intrinsic Evaluation

As described before, word embeddings have the property that similar words tend to lie close to each other and form clusters. To evaluate if the given word embedding has good performance or not, we can evaluate the similarity score of the embedding of words. A measure of performance can be obtained from the results thus obtained. If words have a similar meaning, they have high similarity scores and unrelated words have low scores. It gives a general idea that word embedding has been able to learn the underlying representation of the language. For example, the word vectors for Rupees, Dollars, and Yen should lie close to each other as they all represent the currency of different countries. Similarly, the word vectors for student and cabbage will be far apart from each other as they do not have any relatedness between them.

The word representation given by the embedding system is a dense vector and the similarity between any two words can be obtained by using the cosine similarity. The similarity score obtained by using the cosine similarity method lies between 0 and 1. The similarity score of 1 indicates that the vectors are identical while the score of 0 means that there is no similarity at all. The cosine similarity of two vectors A and B can be calculated as:

$$similarity = \frac{A \cdot B}{|A||B|} \tag{1}$$

At the moment, an existing dataset for word similarity does not exist for the Nepali language. Due to the limitation of time, the creation of a custom dataset in Nepali was not feasible. Since Hindi and Nepali are based on Devanagari and have similar semantic and syntactic rules, similarity datasets in [9] present in Hindi is used with a simple translation. The Hindi word similarity dataset consists of 235 pairs of words consisting of 318 unique words. The dataset can be used in the Nepali language pretty easily. Out of the 318 words around 200 words are words that also exist in the Nepali language. Around 50 words are very similar to the Nepali words and only require a minor change to get the Nepali words The remaining words can be used with simple translations. The image below shows an example of these words.

Type	Examples
Words present in Both Hindi and Nepali	अभ्यास, संकट, परिवार, बाघ
Requiring minor changes	रुपय'>'रुपैया', 'पांच'>'पाँच', 'आंख'>'आखा'
Simple translation	खीरा, मक्खन, मां, सूखा

Figure 2: Words in similarity dataset

Each pair of words are assigned a similarity score in the range of 0 to 10 with 0 meaning no similarity and 10 meaning very similar. One point to consider is that though the antonyms have the opposite meaning to the given word they are thought to be related and thus have higher similarity scores than one would anticipate. It is taken in the context that they are similar words utilized to describe the same phenomenon.

To get a numeric measurement of the performance of the embedding system, the mean absolute error can be calculated between the similarity score present in the dataset and the cosine similarity score obtained using the word embedding vectors for the pair of words by converting them in the same scale. Since the dataset is built considering antonyms as also being high similarity scores, the absolute value of the cosine similarity is only taken.

3.6 Extrinsic evaluation

In the final step, extrinsic evaluation is utilized. For extrinsic evaluation, a simple LSTM-based NER tagger is utilized. This method is utilized in the last step only due to the larger turnover time for a single experiment. The different word embedding models for the choice of hyperparameters that gave better results before are utilized in this step. 5-fold cross-validation is utilized to get a measure of the model in terms of precision, recall, and f1 score.

Precision recall and f1-scores are calculated based on the confusion matrix. In the confusion matrix, we have the count of each of the case's true positives (TP), false positive (FP), false negative(FN), and true negative (TN) based upon the actual class of a given sample and the predicted class by the predictor.

In the case of NER, we have a case of multi-class classification. The precision and recall are calculated per class using the one vs all method i.e. for a target class all other classes are considered as negatives and based on it the calculation is made to get the per-field accuracy.

The dataset for NER has three entities tagged in it: Person, Organization, and location. The tagging has been done in the IO scheme. In this scheme, the text within the entities is classified as the target entity, and any other text is classified as other. It consists of a total of 3600 sentences. The total count per class is shown below.

Table 1: NER Dataset class split

Class	Count
PER	5059
ORG	3811
LOC	2313
Others - O	67904
Total entities w/ O	79087

A simple LSTM-based model has been utilized for the prediction of the entities. The LSTM model can keep important context over a longer period than a traditional recurrent neural network. The basic principle of working LSTM can be seen in the figure above. The implementation of the model has been done in PyTorch. Different word embedding models with appropriate hyperparameters have been used for creating word embedding models. The difference between the tests is only in terms of the word embedding used. To avoid any random noise providing inaccurate results, 5-fold cross-validation has been carried out to get the result of the NER task for a given embedding model.

4. Result

After initial runs trying to finalize the prepossessing, it was observed that the stemming of words provided better results than training without stemming. For stemming, the suffix extraction was carried out. The advantage of stemming was that the OOV tokens were decreased. This was because the count of the root word had increased which meant that the chance that the word was truncated for not meeting the minimum threshold was decreased. One thing to consider though is that stemming needs to be also carried out on the dataset for any subsequent NLP task which can increase the complexity of the NLP task.

4.1 Word similarity

At first, the performance evaluation is carried out using the translated wordsim pairs of words. The mean absolute error is calculated between the word similarity given by the word embeddings and the predefined words. The table below shows the result in the form of 1- error. Thus, larger values signify that the model is learning better. This was used to determine the various values of the hyperparameter that needs to be provided. The model was trained using the dataset which had been

In the first set of tests, a test on the embedding dimension was carried out. For this, window sizes 50, 100, 150, 200, and 300 were used. These window sizes were based on the popular window sizes used in other research in this field. The results for the word similarity task for different embedding sizes were as follows:

Table 2: Comparison of models at different embedding sizes with OOV words

Model	50	100	150	200	300
CBOW	65.14	64.45	64.02	63.78	63.37
Skipgram	75.6	75.17	75.85	75.64	74.33
Fasttext	81.79	81.68	**82.63**	82.62	82.37
GLOVE	67.17	67.57	67.75	68.69	68.76

Most of the models we can see have got better results with the increase in the embedding sizes. The

embedding size of around 150 to 200 seems to be optimum for the corpus that has been used.

The next set of tests was carried out to know which size of the window was giving the better result. For this, a choice of window size was made the choices were 2, 3, 5, 10, and 15. The time taken for the training with a larger window size was higher thus, the window size of up to 15 was only considered. The table below shows the result of the word similarity task at different embedding sizes.

Table 3: Comparison of models at different window sizes

Model	2	3	5	10	15
CBOW	64.27	64.01	64.08	65.17	65.76
Skipgram	70.3	72.25	75.96	79.73	80.24
Fasttext	77.48	80.39	81.57	80.41	79.58
GLOVE	62.95	65.88	69.15	71.51	71.71

Here we can see that the result for the higher window sizes was better most of the time. Only the Fasttext model peaked at the window size of 5. For other models, the performance on window sizes 10 and 15 was only slightly better but the time taken was significantly higher. Based on this, for further experimentation, a window size of 10 was chosen.

4.2 Named Entity Recognition

In the next step, extrinsic evaluation was carried out using the NER task. The models of embedding size 200 with a window of 10 were utilized for the training of word embedding for this task. For the NER task, the choice of hyperparameters was made as per the ones mentioned in [7]

The overall accuracy, precision, recall, and F1-score of the model for different word embeddings were as follows:

Table 4: Comparison of models for NER Task

Model	Acc	Precision	Recall	F1
Fasttext	94.59	83.47	65.60	73.46
CBOW	94.22	83.40	60.82	70.34
Skipgram	94.60	83.35	65.17	73.11
Glove	93.96	69.33	64.08	66.59

The output of the NER also gives us a similar conclusion. We have observed that the performance of the Fasttext and the Skip-gram models are almost similar with the Fasttext model being fractionally better.

5. Conclusion

Based on the experiments carried out, it is observed that the Fasttext model performs better as compared to other models. The Fasttext model utilizes the sub-word embedding scheme which can be attributed to the improved performance. Since in Nepali different subwords have significant importance in the overall meaning of the word it might have been a determining factor. The skip-gram model also gives comparable results to the Fasttext model. The Glove model was not that good when compared to other models. It is also observed that the window size of 10 gives good results for Nepali and that the use of stemming also can significantly improve the performance of the model.

In addition to this, it is also seen that the use of datasets from related languages like Hindi with small adjustments can also give a good indication of the working of Nepali Word embedding.

6. Future Work

One of the major limitations of the models used in the project was that they are not contextually aware. Due to this, the same word embedding is given for polysemy words even when they are used in a different context. Use of models which take into consideration the context might also help better the performance. Another improvement that can be done in the model is the use of better initialization of the word embedding models.

Even though various experiments were carried out there is still a limitation in Nepal in terms of the corpus size when compared to English and other languages with rich datasets. Due to this, there is still a large scope for improving the results obtained. One of the tasks that can be carried out is the use of a common crawl dataset for Nepali.

References

[1] J. R. Firth. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, 1952-59:1–32, 1957.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[3] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.

[4] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2016.

[6] Rabindra Lamsal. 300-dimensional word embeddings for nepali language, 2019.

[7] O. M. Singh, A. Padia, and A. Joshi. Named entity recognition for nepali language. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pages 184–190, Dec 2019.

[8] Pravesh Koirala and Nobal B. Niraula. NPVec1: Word embeddings for Nepali - construction and evaluation. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 174–184, Online, August 2021. Association for Computational Linguistics.

[9] Syed Sarfaraz Akhtar, Arihant Gupta, Avijit Vajpayee, Arjit Srivastava, and Manish Shrivastava. Word similarity datasets for Indian languages: Annotation and baseline systems. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 91–94, Valencia, Spain, April 2017. Association for Computational Linguistics.

[10] Dimuthu Lakmal, Surangika Ranathunga, Saman Peramuna, and Indu Herath. Word embedding evaluation for Sinhala. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1874–1881, Marseille, France, May 2020. European Language Resources Association.

[11] Nepali monolingual written corpus. `http://catalog.elra.info`.

[12] Urdu-Nepali-English Parallel Corpus. `http://www.cle.org.pk/software/ling_resources/UrduNepaliEnglishParallelCorpus.htm`.

[13] Nepali stemmer. `https://github.com/oya163/nepali-stemmer`.