


# Building Machine Learning System with Deep Neural Network for Text Processing

Shashi Pal Singh<sup>1</sup>, Ajai Kumar<sup>1</sup>, Hemant Darbari<sup>1</sup>, Anshika Rastogi<sup>2</sup>,  
Shikha Jain<sup>2</sup>, and Nisheeth Joshi<sup>2</sup>

<sup>1</sup> AAIG, Center for Development of Advanced Computing, Pune, India  
{shashis, ajai, darbari}@cdac.in

<sup>2</sup> Banasthali Vidyapith, Banasthali, Rajasthan, India  
anshikarastogi1992@gmail.com, shikhaj959@gmail.com,  
jnisheeth@banasthali.in

**Abstract.** This paper provides the method and process to build machine learning system using Deep Neural Network (DNN) for lexicon analysis of text. Parts of Speech (POS) tagging of word is important in Natural language processing either it is speech technology or machine translation. The recent advancement of Deep Neural Network would help us to achieve better result in POS tagging of words and phrases. Word2vec tool of D14j library is very popular to represent the words in continuous vector space and these vectors capture the syntactic and semantic meaning of corresponding words. If we have a database of sample words with their POS category, it is possible to assign POS tag to the words but it fails when the word is not present in database. Cosine similarity concept plays an important role to find the POS Tags of the words and phrases which are not previously trained or POS Tagged. With the help of Cosine similarity, system assign the appropriate POS tags to the words by finding their nearest similar words using the vectors which we have trained from Word2vec database. Deep neural network like RNN outperforms as compare to traditional state of the art as it deals with the issue of word sense disambiguation. Semi-supervised learning is used to train the network. This approach can be applicable for Indian languages as well as for foreign languages. In this paper, RNN is implemented to build a machine learning system for POS-tagging of the words in English language sentences.

**Keywords:** Natural Language Processing (NLP) · Machine learning · Recurrent Neural Network (RNN) · Cosine similarity · Word2vec

## 1 Introduction

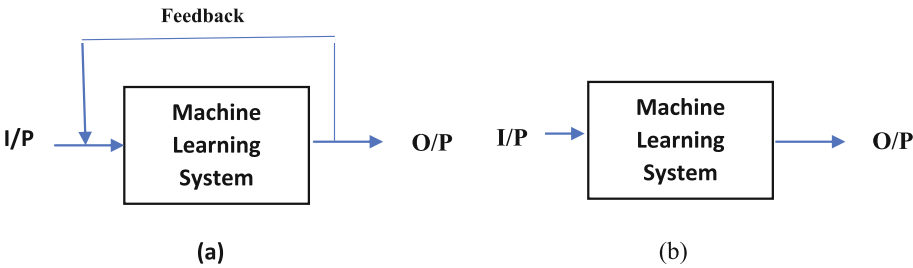
Machine learning is one of the most popular research topic in NLP application area. Different techniques and strategies are available to train the system. Deep learning is very hot topic and this approach have proven its importance to build a better machine learning system as compare to state-of-the-art [6]. It is a science that provide us tools and techniques that make a system capable to sense the input from the environment and perform like a human brain [11]. Our main focus exclusively on text processing that

how learning methods can be implemented using deep neural networks so that the system produce more accurate results.

### 1.1 Deep Neural Networks

Deep learning uses neural networks but with two or more hidden layers. These networks are called deep neural networks (DNNs). DNNs have the capability to learn features so they are very useful in machine learning processes. They are very good in learning semantic and syntactic representations of text (words, sentences, structures etc.). These networks go through the training phase followed by inference phase. Structure and training process of DNN depend on the performing task [4].

In Training Phase, Feedback loop is present here, in order to improve the system on the basis of error (difference between actual output and target). Weights are modified with each iteration to minimize the error and to build well efficient and robust system. On the other hand, In Inference Phase, a well-trained system produces output according to the given input. There are only feed-forward network connections (Fig. 1).



**Fig. 1.** (a) Training phase and (b) interface phase

### 1.2 Machine Learning

Machine learning is an important part of artificial intelligence. Machine learning methods are used in training of the computer system and make it capable to perform specific task. Learning can be based on rules, algorithms, features etc. A well-trained system sense the environment and with the help of its knowledge, it will produce the required output.

## 2 Vector Representation of Words

First task is to find out vector value of a word. Word embedding concept is used here. It is the vector representation in a dense and low dimensional space. A large corpus wiki [14] (3 GB) is go through the training which is necessary for learning purpose. The word2vec is neural network, required to produce the vectors that will be used as input for deep networks. The idea use here is to find the numeric value of a word that is based

on the context words. Different models and learning algorithms are available for this task. The whole training is based on unsupervised learning [7, 8].

There have been a number of models, algorithms to address the general word embedding problem with neural network. Here, the approach defined is closely related to word2vec that first map input to the continuous space vector representation and then word vectors works as a lookup table.

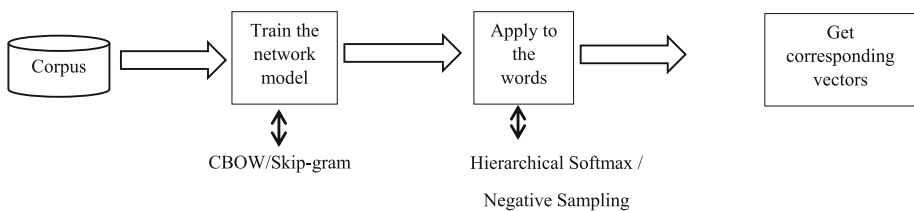
Skip-gram and CBOW Models are two ways of creating the “task” for the neural network, where we create “labels” for the given input. Both architecture describe how the neural network “learns” the underlying word representation for each word. Since learning word representation is essentially unsupervised, learning algorithms perform well to “create” labels to train the model [8].

## 2.1 Learning Algorithms

Two popular Learning algorithms are hierarchical softmax and Negative sampling. In hierarchical softmax sampling use Huffman tree concept and assign short codes to frequent words. But it is not useful as number of epochs increases. In this neighbouring word is pulled closer or away from a word subset. This word subset is chosen from tree structure and may be different for each word. Negative sampling method is a good algorithm that close the neighbour words and some words are pushed away. This complete working is depending on the approach, maximization problem and the idea use here is minimization of the log-likelihood of sampled negative instances [9].

## 2.2 Dimensionality Reduction

Dimensionality Reduction is also a part of word2vec processing. Initially, one dimension per word is allocated. To convert a high-dimensional input into a fixed-size vector representation, dimension reduction technique is used [12] (Fig. 2).

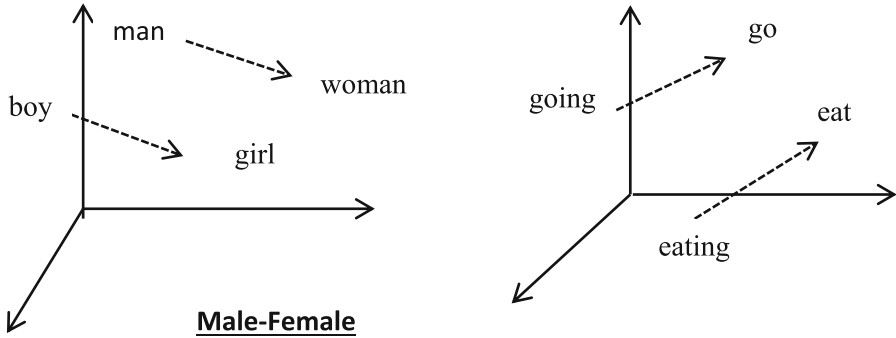


**Fig. 2.** Block diagram of word2vec processing

Word2vec places the words in continuous space where each word is represented by vectors of fixed dimensions (usually 100–500). Original features in text processing is too sparse so we need distribute representation that enables more flexibility when it uses in language modelling step. Vectors of a word can be find out if the vectors of other words in same dimension are known [8, 9]. For example

$$\mathbf{Vector}[\text{girl}] = \mathbf{Vector}[\text{man}] + \mathbf{Vector}[\text{woman}] - \mathbf{Vector}[\text{boy}] \quad (1)$$

Following figures visualize this more clearly (Fig. 3);



**Fig. 3.** Continuous space representation of words

Now, these vectors can be used in different machine learning applications according to need. For example, to find the similarity between the words or to find the vector of a new word, in sentiment analyses, text summarization etc. [12, 15].

### 2.3 Cosine Similarity

It is a way to find the similarity among the words by finding the dot product of their vector representation. The cosine similarity between two vectors can be given by the equation: [13]

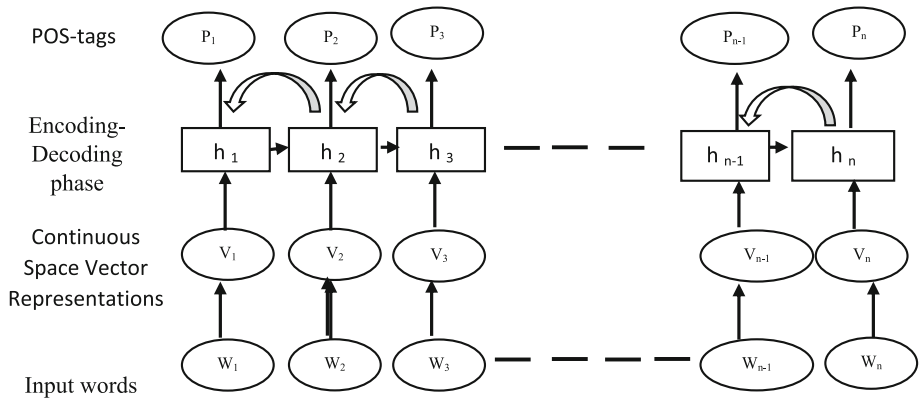
$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|} \quad (2)$$

## 3 Word2vec Implementation in NLP

We can use vector representations of words in various NLP applications to train the neural network systems. Word-vectors capture syntactic and semantic information which is important in text processing tasks like POS-tagging, semantic analyses, machine translation etc. [10]. To assign the POS-tags to the token in the input sentence, we have created a table in database which contains some sample of words with their POS-category. When tokens generate, next step is to find its cosine similarity with sample words and assign the POS-tag of the word to the token that is most similar to it. Here we are using concept of cosine similarity because it will find the nearest word in the vector space correspond to the each given token [1, 12].

## 4 Machine Learning Using DNN

One issue with the above method is that it can't handle the words which belong to more than one category according to their use in sentence. For example, book can be noun or verb. This will be depended on neighbouring words because tags of neighbouring words are dependent. Some tags are very unlikely to be followed by other tags (e.g. verbs never follow the determinant) and some tags represents the word chunks (e.g. proper noun can consist more than one word) (Fig. 4).



**Fig. 4.** POS-tagging through RNN

To resolve these problems, we require a well-trained system that can provide the most suitable tag to a word by using the information of its neighbour. RNN has the capability to maintain the history of the words in a given sentence and then it uses this information in decoding phase [2, 3]. That is the reason, it is used in building a machine learning system for POS-tagging [5].

## 5 Methodology

We have trained a 3 GB wiki file with word2vec. These word vectors required in neural network processing. In our experiment, semi-supervised learning method train a machine for POS-tagging of words.

First task is applying pre-processing on input that includes sentence separation, contraction removal, and tokenization.

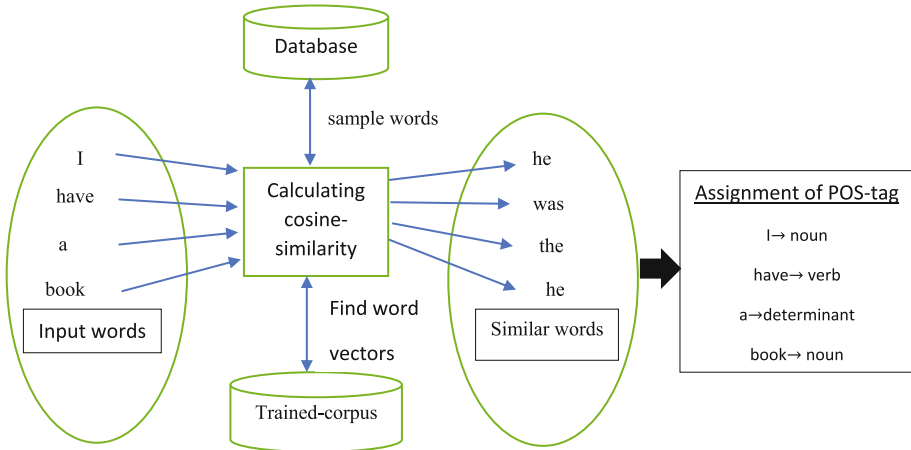
After the pre-processing, next step is to find vector values of words using trained corpus. Suppose we have an input sentence  $S_1$  that consists words  $w_1, w_2 \dots w_n$ . Trained wiki file is used to find the corresponding vectors of the words  $v_1, v_2 \dots v_n$ , respectively. To learn the system for POS-tagging, a small database is required that have some sample words with their POS-category.

Let us take an example 1, I have a book (Table 1).

**Table 1.** Sample words in database table

Words	POS-category
The	Determinant
Bad	Adjective
He	Noun
Was	Verb

We can represent this whole process with the help of Fig. 5.

**Fig. 5.** POS-tagging of words with the help of word vectors

If  $\langle S_1, V_1 \rangle$  is the pair of set, where  $S_1$  is the set words of input sentence  $w_1, w_2, \dots, w_n$  and  $V_1$  contains their corresponding vector values  $v_1, v_2, \dots, v_n$ . Similarly,  $\langle S_2, V_2 \rangle$  is the pair of set, where  $S_2$  is the set of sample words.

$w'_1, w'_2, \dots, w'_m$  that exist in the database and  $V_2$  contains their corresponding vector values  $v'_1, v'_2, \dots, v'_m$ .

$$S_1 = \{w_1, w_2, \dots, w_n\}$$

$$V_1 = \{v_1, v_2, \dots, v_n\}$$

$$S_2 = \{w'_1, w'_2, \dots, w'_m\}$$

$$V_2 = \{v'_1, v'_2, \dots, v'_m\}$$

Where  $v_i, v'_j \in \mathbb{R}$  for all  $i, j \in \mathbb{N}$ .

Find the similarity of each input word vectors with all sample word vectors and find out the most similar sample word for each input word using following equation

$$C = \sum_{k=1}^m f(v_i, v'_k) \text{ where } C \in \mathbb{R} \quad (3)$$

Here the function

$$f(v_i, v'_k) = \frac{k_1}{k_2}$$

$$K_1 = V_i \cdot V'_k \text{ and } k_2 = \|v_i\| \cdot \|v'_k\|.$$

We have  $\langle P_1, P_2 \rangle$ , pair of set that contains category values of sample words and input words, respectively.

Initially,

$$P_1 = \{p'_1, p'_2, \dots, p'_m\} \text{ and } P_2 = \{p_1, p_2, \dots, p_n\}$$

Suppose  $v'_j$  is the most similar to  $v_i$  then assign the POS tag value of  $w'_j$  to the  $w_i$  POS category as shown in the following equation

$$p_i = p'_j \quad (4)$$

When we use recurrent neural network to deal with the disambiguity issue then we find out the POS tag using the given equation

$$p_{i+1} = f(p_i, v_{i+1}) \quad (5)$$

Here  $p_i \in P_2$  and  $v_i \in V_1$ .

When we apply only cosine similarity concept to find out the POS tags of the words then wrong values of the words can be obtained because some words come under more than one POS category. For e.g., book can be used as noun (as shown in above example 1) or as a verb (as shown in following example 2) Book my tickets for the evening show.

Hence, we are using RNN to assign correct tag to the word. In example 1, book follows the word “a” which is a determinant. According to English grammar rule, a verb follows a determinant in more unlikely cases. RNN assign the tag “noun” to the book because it has the information about previous word i.e. “a”.

## 6 Testing

We can calculate the accuracy of the POS tag system with the help of following formula,

$$A = \frac{N_C}{N_T} * 100 \quad (6)$$

where  $N_C$  is the number of words to which the system has assigned correct POS tags and  $N_T$  is the total number of words in the given input sentence.

## 7 Conclusion

The recent advancement of Deep Neural Network would help us to achieve better result in POS tagging of words and phrases so we have opted the Machine leaning using DNN,

RNN and word2vec. For the testing and analysis purpose we have used 3 GB wiki data/file to do machine learning for identifying the POS tagging of the words and phrases. RNN has the capability of feature learning and sense the accurate meaning based on previous history. In this task RNN provide better results in POS tagging as it can handle the issues related to word sense disambiguity.

## 8 Future Work

Result obtained from the above approach can be used in various NLP applications like speech reorganization, semantic analyses, machine translation etc. We can implement this idea on Indian languages as well as on foreign languages. It is difficult to train the large data file on CPU and deep neural network also need a powerful hardware support for training. GPUs can be used to improve the system performance by reducing training time period.

## References

1. Nogueira dos Santos, C., Zadrozny, B.: Learning character-level representation for part-of-speech tagging. In: Processing of the 31st International Conference on Machine Learning, Beijing, China, vol. 32 (2014). JMLR: W&CP
2. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. *J. Mach. Learn. Res.* **1** (2009)
3. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to Sequence Learning with Neural Networks. Google
4. Zhang, J., Zong, C.: Deep Neural Network in Machine Translation. Institute of Automation, Chinese Academy of Sciences
5. Perez-Ortiz, J.A., Forcada, M.L.: Parts-of-speech tagging with recurrent neural network. *IEEE* (2001)
6. Deng, L., Yu, D.: Deep Learning: Methods and Applications, vol. 7. Microsoft Research, Redmond (2013)
7. Chen, M.: Efficient vector representation for documents through corruption. In: ICLR. Criteo Research, Palo Alto (2017)
8. Mikolov, T., Sutskever, I., Chen, K.: Distributed representations of words and phrases and their compositionality. Google Inc.
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representation in vector space. [arXiv:1301.3781](https://arxiv.org/abs/1301.3781) v3, 7 September 2013
10. Zheng, X. Chen, H., Xu, T.: Deep learning for Chinese word segmentation and POS tagging. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013
11. <http://Www.Deeplearningbook.Org/>
12. <http://Nlp.Cs.Tamu.Edu/Resources/Wordvectors.Ppt>
13. <http://Www.Minerazzi.Com/Tutorials/Cosine-Similarity-Tutorial.Pdf>
14. <http://www.cs.upc.edu/~nlp/wikicorpus/>
15. <https://deeplearning4j.org/word2vec>