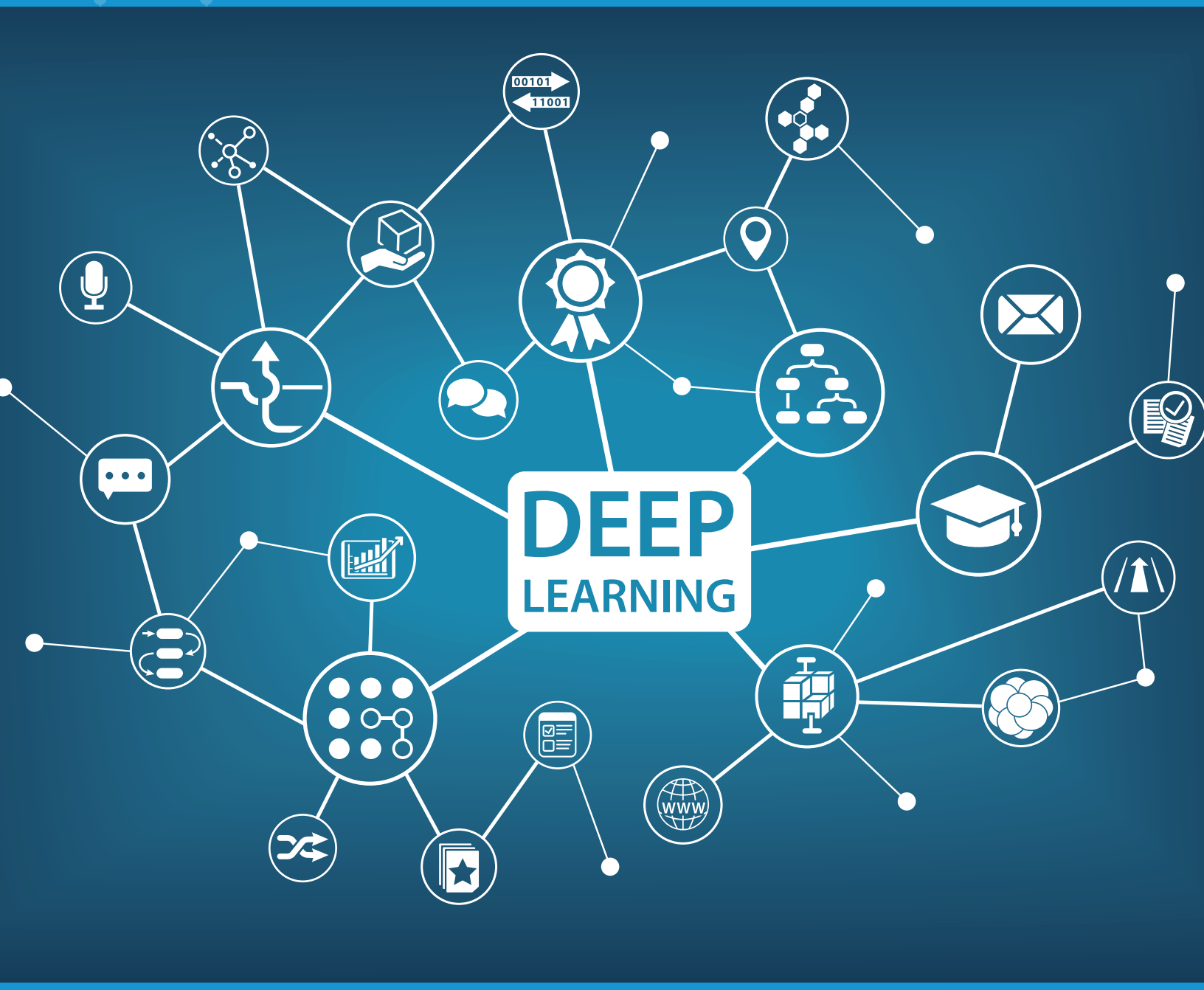
Mphasis
Unleash the Next



Deep Learning in Natural Language Processing

Ashutosh Vyas
Assistant Manager, Mphasis NEXTlabs

NEXT labs

Introduction

In recent years multi-layer neural networks are gaining back their importance in the field of Natural Language Processing (NLP), as they are becoming capable of generating state-of-the-art results with a relatively decent speed due to utilization of Graphical Processing Units. The introduction of different methods in multi-layer neural network, such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Longest State Time Memory Neural Networks (LSTM) has proved to be very efficient in generating better results.

In this paper, we will discuss different methods that can be adopted to perform NLP using deep learning methodologies. We will start our discussion with representation methods available to present the data, then we move to feed forward NN, RNN, CNN and see how to use these methods for NLP.

Abstract

Deep learning has emerged as a highly efficient technique in machine learning to perform text analytics, which includes text classification, text sequence learning, sentiment analysis, question-answer engine, etc.

This paper presents the concept and steps of using deep learning in NLP, ways to pull out the features from the text, ways to prepare data for deep learning and popular methods that are being used for NLP, including RNN, CNN, LSTM.

We also present the results of a working example based on a movie review shared by IMDB.

Feature Representation

When feeding textual data to Artificial Neural Networks (ANN), we need to consider few points about the way we create the data. We input the data vector x in-dimensions and generate the output in out-dimensions. Instead of creating an individual dimension for each and every feature, we need to embed each feature into a D-Dimensional space and represent as a dense vector in the space.

The main advantage of dense vector representation is that if there exists a similarity sequence which we can learn, that can be captured best in dense vector representation.

In the example presented, we used continuous-bag-of-words (CBOW) approach to build our dense vectors.

Feature Extraction

Let us now look at some of the methods used to extract features from the given text:

CBOW: Continuous bag of words is a dense vector based feature extraction approach. Neural networks assume a fixed input. Thus for training the NN we need to extract features from textual data. CBOW approach enables feature extraction by expressing each word in the training data set as unique feature. This creates a vector representation for each sentence to be used in training NN. For specific importance association we provide a weighted vector as an input to NN. This weight association is done majorly using two approaches – summing or averaging of embedded features and weighted averaging of embedded features. An example is tf-idf of collection of bag of words and giving weight to different dimensions based upon importance. **Note:** We can adopt other methods also as discussed in section titled 'Word Embedding'.

Distance and Position Features: The distance can be calculated by subtracting the feature vectors of the corresponding words, and this information can be used to train the Neural Networks.

Dimensionality: Decision needs to be made on the dimensionality of the embedded vectors. It is a trade-off between processing speed and task accuracy.

In our example, we used the context window of *ten words.*

Training Functions

Given below are the types of functions that are used in modeling deep learning neural networks. The selection of these functions depends majorly on the problem statement.

Non-linear Functions

Neural networks are universal approximators and thus can approximate any non-zero linear or non-linear function. The two basic things that should be kept in mind before selecting a non-linear function are:

1) It should cover the entire input range, from negative infinity to positive infinity, as its domain.

2) Its output should be bounded in a set range.

The popular non-linear functions used are: Sigmoid, Hard Tanh, Hyperbolic tangent, Rectifier, Relu.

Output Transformation Functions

In many cases we use a transformation function in the outer most layer of neural network.

The most popular function used here is softmax, to represent multi-class classification.

Loss Function

When training the neural network, we require a loss function which acts as an indicator to identify how far is the network trained output from the actual output.

Selection of lost function depends upon the type, requirement and extent of tolerance of loss function.

Most popular loss functions are: Hinge-binary, Hinge-multiclass, log-loss, categorical cross entropy loss, ranking loss.

Word Embedding

Let us now look at the methodology to represent each of the features into a low dimensionality space. This technique is popularly used for sequential pattern training of the sentences in the documents.

Continuous-Bag-of-Words

This approach is used to capture the sequential information in the sentences. The process is as follows:

Total number of unique words are collected and all the words are given a vector representation, this is called **one hot vector representation.** Depending upon the window size, words that are sequential part of a sentence are collected and a training model is developed. E.g.:

A quick brown fox jumps right over a lazy dog.

In the above sentence, suppose we represent a word with one word to its left and one to its right, then the training data would look like:

[(a, brown),quick],[(quick,fox),brown] etc.

Then a neural network is taken with input layer's size = C(Context)*V(vocabulary size) and a hidden layer's size = N(heuristic) and output layer's size = V(vocabulary size).

An optimization technique like gradient descent, is used to train the model.

Figure 1 represents single word context CBOW.

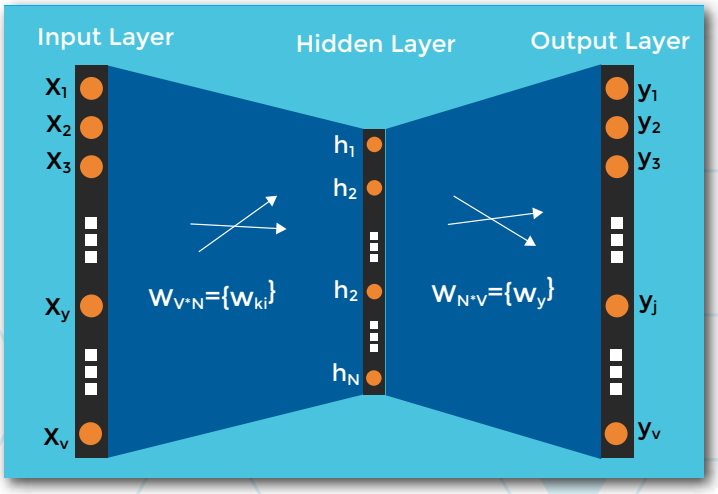


Figure 1

Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of neural network where connections between units form a directed cycle. This allows NN to create an internal state of the network which allows it to exhibit dynamic temporal behavior. RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unregimented connected handwriting recognition or speech recognition.

RNNs are neural nets that can deal with sequences of variable length. They are able to perform this by defining a recurrence relationship over timestamps which is typically the following formula:

$S_{k=f}(S_{k-1}xW_{rec}+X_kxW_x)$

Above equation forms the basic structure of RNN. The previous state result is used to calculate the current state output.

Figure 2 represents a fully connected recurrent neural net.

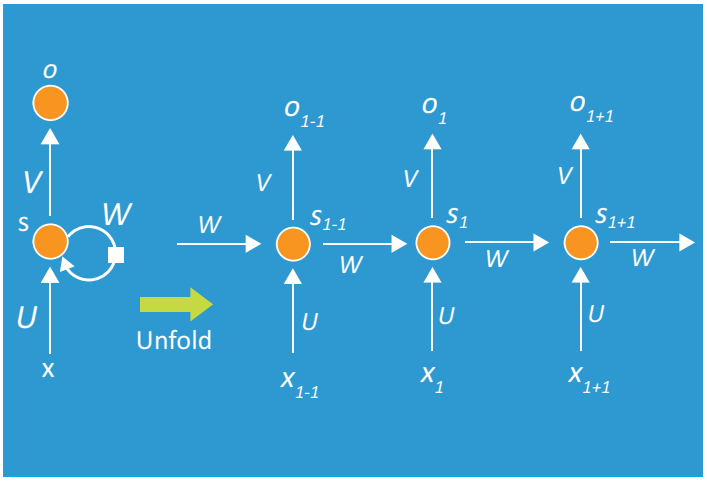


Figure 2

In our example, we used LSTM, a version of RNN due to the presence of *vanishing gradient problem* in RNN.

Convolutional Neural Networks

Convolutional neural networks are a type of artificial neural networks that use convolution methodology to extract the features from the input data to increase the number of features.

The basic steps are as follows: Considering the input data set, we decide and select N number of convolution functions, also known as filters, where N = maximum number of features we want. After the selection of filter functions we convolute the input with the filters. The output from the convolution is a D-Dimensional vector, which is then sent for pooling.

Pooling is a way of selecting most relevant features from the set of given features. Mostly for text analytics we select the features with max or average values.

Once the best features are selected, the normal architecture of ANN is followed.

Convolutional NN are majorly used for image classification, text classification, etc.

Figure 3 demonstrates a fully connected, multilayered filtering CNN.

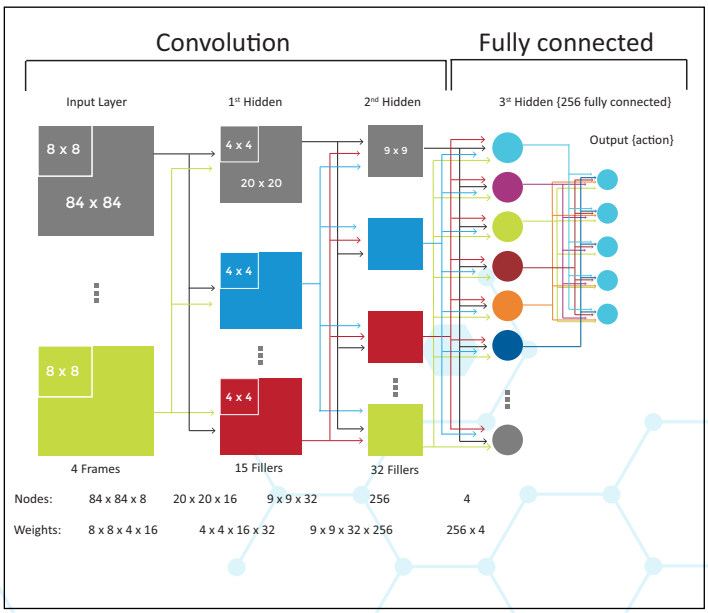


Figure 3

CNN are good performers for text classification, but poor in learning the sequential information from the text. Hence, LSTM was used in the given example.

LSTM

Long-short-term-memory (LSTM) is to deal with the concept introduced for Recurrent Neural Networks because of the problem of **vanishing gradient** in RNN.

Vanishing gradient problem is faced in machine learning while training the neural nets using the back propagation algorithm. In back propagation, each of the weights receives an update proportional to gradient of the error function. As the number of layers increase, the gradients of the error functions multiply. This slows down the training process of NN terribly.

IN RNN, this problem exacerbates when the feedback component is also added to the training of weights.

To handle this, LSTM was introduced by Hochreiter and Schmidhuber (1997).

Figure 4 represents the blog diagram of LSTM.

LSTM selects the parameters and decides when to and when not to update the memory based on the log-logistic function, working on the input vector and the latest learnt weight.

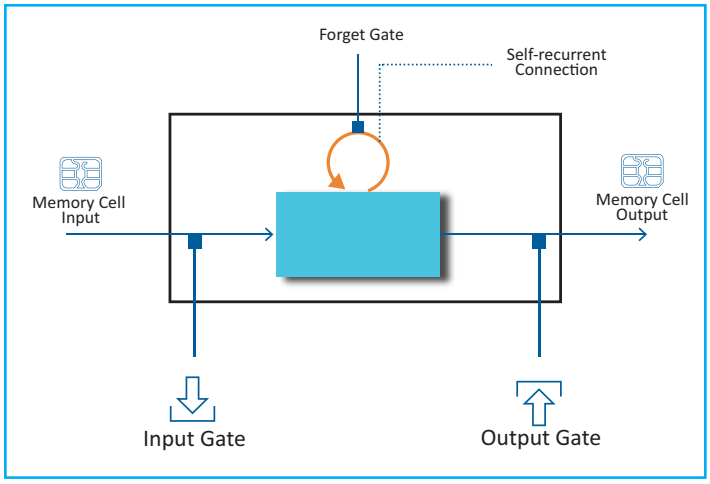


Figure 4

Forget gate helps LSTM to forget the learnt weight if the effect of carrying it forward is negligible, thus reducing the long term dependencies.

Working Example

We performed sentiment analysis of movie review data taken from IMDB.

The data has 5000 total sentences including positive and negative sentiments.

We used RNN – LSTM as we need to train the sequential information to our neural nets.

Data was split into 75% training and 25% testing data by random sampling.

Deep Learning module was constructed as mentioned below:

Word embedding: **skip-gram** (similar to CBOW)

RNN LSTM layer = **1**

Number of LSTM nodes = **100**

Neural Nodes = **5**

Output Node = **1** (as there exist only 2 classes to classify)

Cost function: **binary cross entropy**

Optimizer: **RMS prop**

Non-linearity = **tanh**

After creating the neural network using Python as the programming language, following result was obtained:

Validation Accuracy: 82.30%

This accuracy signifies that out of 25% of the entire data (test set), having 0 - 1 tags , 82.30% was correctly classified as positive or negative sentiment.

References

[1]. Goldberg, Yoav. "A primer on neural network models for natural language processing." arXiv preprint arXiv:1510.00726 (2015).

[2]. Rong, Xin. "word2vec parameter learning explained." arXiv preprint arXiv:1411.2738 (2014).

Ashutosh Vyas

Assistant Manager, Mphasis NEXTlabs

Mr. Ashutosh Vyas is an Assistant Manager at Mphasis NEXTlabs. He completed his B.Tech from SKIT, Jaipur Rajasthan Technical University, followed by M.Tech from IIITB Bangalore in Information Technology. He pursued his interest in analytics and machine learning by joining NEXTLabs as a Sr. Analyst, where he worked on different analytical and technical aspects and designed an agent based simulation model for Mphasis proprietary product – InfraGraf. He designed a machine learning based information retrieval algorithm from documents and sentiment analysis module using deep learning. Ashutosh also published a paper on Life Event Detection in InnovEx Asia 2016 conference.

His technical interest resides in machine learning, agent based modelling, optimization theory, graph theory, stochastic based analysis, and time series analytics.

About Mphasis

Mphasis is a global technology services and solutions company specializing in the areas of Digital, Governance, Risk & Compliance. Our solution focus and superior human capital propels our partnership with large enterprise customers in their digital transformation journeys. We partner with global financial institutions in the execution of their risk and compliance strategies. We focus on next generation technologies for differentiated solutions delivering optimized operations for clients.