Illinois Institute of Technology

Recurrent Convolutional Neural Network for Text Classification

Submitted by:

Bharath Elangovan (A20344918)

Guru Prasad Natarajan (A20344932)

Date: 04/18/2016

Abstract

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned processing the given set of corpus based on the NLP task. The NLP has so many applications like Automatic summarization, Discourse analysis, Part-of-speech tagging, Sentiment analysis, Speech recognition etc.

Sentiment analysis is a sub category of Text classification which is a foundational task in many NLP applications. There are lots of ways of handling text classifiers. It can be done without the use of Machine Learning but the hybrid of Machine Learning and NLP has proved to show better results than the conventional methods. We have worked on different models to compare their accuracy on different models. We have implemented Naïve Bayes, Logistic Regression and we have also implemented Recurrent Neural Network, Convolution Neural Network and hybrid of Recurrent and Convolution Neural Network.

We conducted experiments on the IMDB dataset. Logistic Regression model was modeled using Pyhton's inbuilt packages. Recurrent Neural Network with three hidden layers where implemented with a feedback loop. We experimented on recurrent convolution neural network for text classification. In our model, we apply a recurrent structure to capture contextual information as far as possible when learning word representations, which may introduce considerably less noise compared to traditional window-based neural networks. We developed a hybrid method which is a combination of the Convolutional Neural Networks and the Recurrent Neural Networks and we call that network to be Recurrent Convolutional Neural Network (RCNN). We stack up the layers in sequential manner so that the output from the above neural network can be passed as input to the subsequent layers. From our model the recurrent structure and convolutional neural network the experiment demonstrates that our model outperforms CNN and RecursiveNN for the given dataset

Introduction

In Natural Language Processing many challenges in involve natural language understanding, enabling computers to derive meaning from human or natural language input, and others involve natural language generation. Modern NLP algorithms are based on machine learning, especially statistical machine learning. The paradigm of machine learning is different from that of most prior attempts at language processing. In this project we have experimented text classification through conventional Logistic Regression, Naïve Bayes and also implemented Recurrent, Convolution and the hybrid of both neural networks. We have experimented to see if our implementation is more accurate and robust than the conventional method. A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. The experiment on Recurrent Neural Networks is done with three hidden layers. Convolutional neural networks are used widely in the field of natural language processing. CNN models have subsequently been shown to be effective for various NLP problems and have achieved excellent results in semantic parsing, search query retrieval, sentence modeling, classification, prediction, and other traditional NLP tasks.

Data Preprocessing

The dataset that we used is a sequence of texts with the tab separated labels. The dataset is well organized so we had to only take care of the cleaning and tokenization part.

We read each and review posted by the users and cleans the sentences by removing the unwanted characters that do not aid much with the classification process. Once the data cleaning process is done, we tokenize the sentences and obtain the individual words in the sentence.

Post tokenization, we make all the sentences in the dataset to be of equal size by appending a constant word. This is done by taking the sentence that is the longest of all the sentences as the maximum size of the features and sentences less than the maximum are appended with the number of constant word which is equal to (max_size – actual_size). This proved to be very effective in creating the feature vectors for the deep learning techniques and provides a solution for the data sparsity problem which is faced by traditional machine learning approaches such as Logistic Regression, etc.,

Following the above process, we create a dictionary of the vocabulary used in the sentences. In order to get rid of the stop words, we just take into account the words that are most common and we index the words followed by the feature vector creation.

We have also experimented on the Counvectorizer in python to perform analysis over bigram and tri-gram features. This data has also been experimented on the implemented Recurrent Neural Networks.

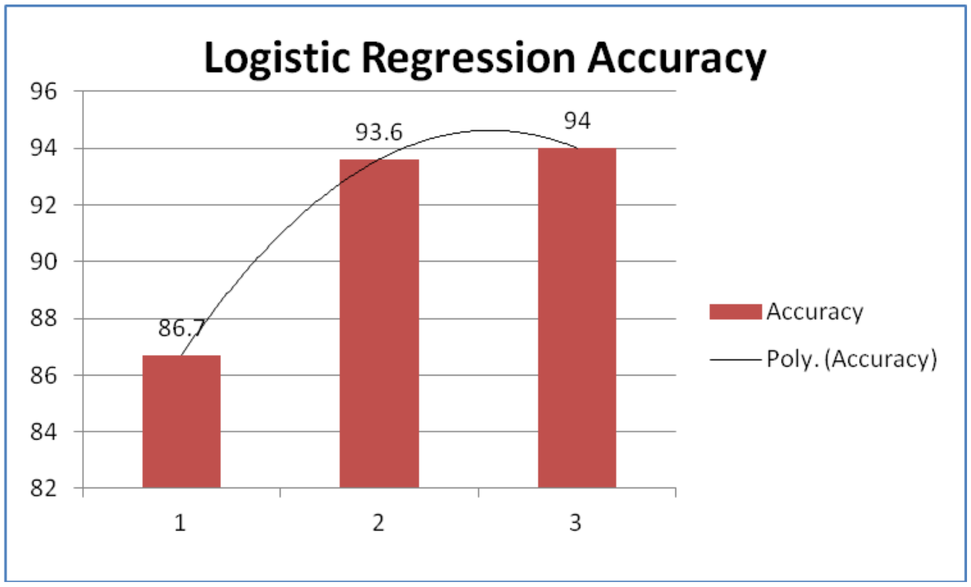
Models

BoW model with Naïve Bayes

We developed a Naïve Bayes model by binarizing the feature vectors. We used the same dataset that will be used in the neural networks that we build. Though this model did pretty well with the data still it suffers from a major setback which is data sparsity problem. We address a solution to the problem while constructing the deep learning techniques.

Logistic Regression

We have built a Logistic Regression model which uses the features from the Countvectorizer object. This is basically the bag of words method of classifying the text and this has the least percent of accuracy of all the other models we have implemented on. The highest accuracy among the experiments was by Trigram analysis of the text classification.



Recurrent Neural Network

We have implemented RNN in two different ways. One is the manual implementation of RNN with python functions and by using Keras which is a library written in python for developing complex neural networks, which helps us to compare and contrast the hybrid model that we are going to build in terms of time and accuracy.

In the manual implementation of RNN we have used three hidden layers with a cyclic network as a feedback loop. We experimented on implementing a basic model of the RNN and evaluate its performance over the other model. For this implementation the gradient for the third has been derived with at most care. The data from the preprocessing and the Countvectorizer object was experimented along with the variation in the number of iterations and the number of features. When accuracy of the model dropped when there was feature reduction from the original number of features.

$$\frac{\partial E}{\partial v} = \sum(y' - y)^2 Z$$

$$\frac{\partial E}{\partial y'} = \sum\sum(y' - y)VZ(1 - Z)Q$$

$$\frac{\partial E}{\partial s} = \sum\sum\sum(y' - y)VW(1 - W)Q(1 - Q)QX$$

The above derivation was implemented for each sample of data and below is the model of simple RNN we have implemented.

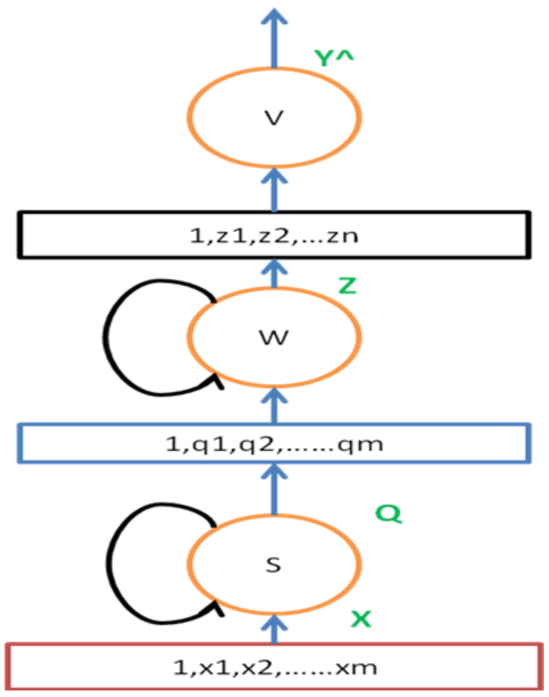


Figure 1. Implemented Recurrent Neural Network Model

Convolutional Neural Network

Convolutional neural networks are used widely in the field of natural language processing. CNN models have subsequently been shown to be effective for various NLP problems and have achieved excellent results in semantic parsing, search query retrieval, sentence modeling, classification, prediction, and other traditional NLP tasks.

So,we developed a convolutional neural network(CNN) using Keras. The convolutional network is a model of type random that contains single dimensional convolution layers. We build different layers by varying the filter size and a fixed number of filters. Since we use single dimensional layers we used the filters to be single dimension and the stride is set to its default. With the default setting, we will have much more overlapping fields, which would be ideal for Natural Language Processing applications.

Also, we use the border mode which helps in dealing with the input on the borders by applying zero padding. This is done to preserve the spatial size of the output. Above all the hyper parameters, we also used dropout as the regularization method. Since the model that we have built is a fully connected network. It is vulnerable to overfitting of the data. By using the dropout, which is a probability, we drop the output of the nodes after the training stage with a probability of 'p'.

Finally, we use ReLU(Rectified Linear Unit) as the activation function for the convolution layers and sigmoid as the activation function for all output units.

Below are the steps used to build the convolutional network:

Step 1: Select the cnn model

Step 2: Define the hyperparameters of the model (such as n_gram filters, stride, dropout, embedding dimension, number of hidden units)

Step 3: For each gram specified in the filter, build a 1-D convolutional layer followed by the max-pool layer and apply activation function to create a fully connected layer.

Step 4: Repeat Step 3 for all the grams and create a subnet and concatenate the outputs.

Step 5: Create the word embedding by using the vocabulary built which gives representations of the words and add the subnet below, so the subnet can utilize the embeddings.

Step 6: Add activations to the output layer.

Step 7: Compile the model built with customized or default objectives and optimizers.

Step 8: Train and test the data to observe the results.

Since the problem that we are trying to solve is a classification problem, we used binary_crossentropy as the model objective and stochastic gradient descent as the optimizer with its default settings.

Recurrent Convolutional Neural Network

We developed a hybrid method which is a combination of the Convolutional Neural Networks and the Recurrent Neural Networks and we call that network to be Recurrent Convolutional Neural Network (RCNN). We stack up the layers in sequential manner so that the output from the above neural network can be passed as input to the subsequent layers.

We use the same approach that we used to build the convolutional neural network that forms a layer for the new model and the output from the layer before reaching the output layer is fed into the recurrent convolutional layer which is Long Short Term Memory (LSTM) in our case. We made the model simple just by adding a single layer of both the models.

Approach:

We model a deep neural network approach to capture the semantics of the text. The model will have a Document D as its input which is a sequence of words w1,w2....wn. The output of the model is the class labels. Below are the steps:

We pre-process the data by tokenizing the texts. We either use tokenizer of the nltk package or a custom-built tokenizer.

1) Word Representation Learning: In this step, we combine a word and its context to present a word. The context helps us to obtain a more precise meaning of the word.

2) Text Representation Learning: Once all the word representations are calculated, we apply a max-pooling layer. It converts the texts with various lengths into a fixed-length vector with which information throughout the entire text can be captured.

3) Pre-training: In this step, we perform pre-training of the word embedding which is a distributed representation of a word and is a suitable input for neural networks. In this, we use skip-gram model to pre-train the word embeddings which trains the embeddings of words w1,w2....wn by maximizing the average log probability.

4) Training: We define all the parameters to be trained as θ. The parameters are word embeddings, bias vectors, initial contexts and the transformation matrices. The training of the network is to maximize the log-likelihood with respect to θ.
   We have chosen commonly used hyper-parameters for the neural networks. We use stochastic gradient descent to optimize the training target. In each step, we randomly select an example and make a gradient step.
   Moreover, we set the default learning rate of the stochastic gradient descent provided by Keras, the hidden layer size as H =50 , the vector size of the word embedding as |e| = 20.

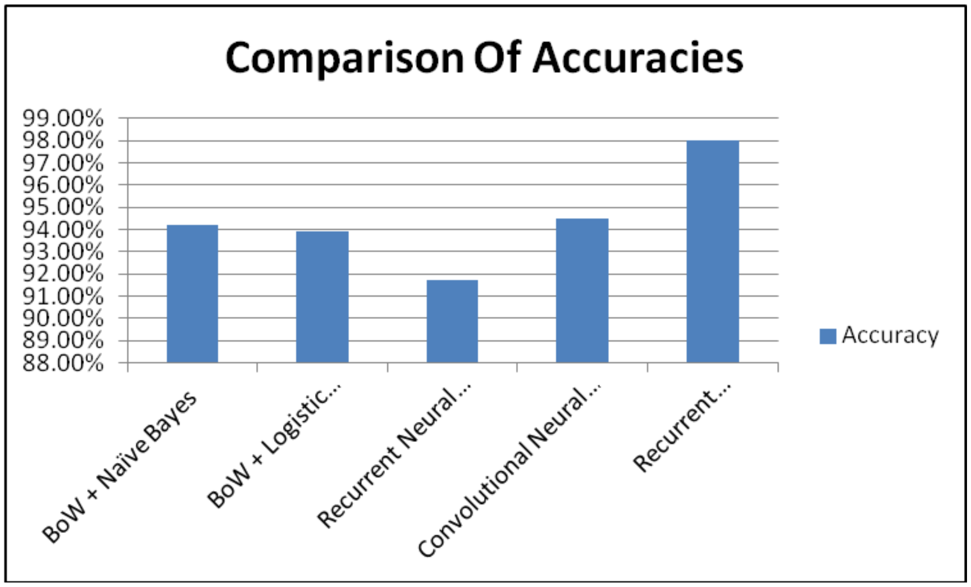
Comparison of methods

We compare the model that we built with some of the approaches that is being widely used such as Logistic Regression and Naïve Bayes using the BoW approach and other neural network approaches such as recurrent neural network and convolutional neural network.

Results and Discussion

The experimental results are shown in the table below. For the neural networks models we used the same number of epoch to determine which one does best. The results are obtained by varying the hyperparameters and selecting the one that produces the best results.

Model	**Accuracy**
BoW + Naïve Bayes	94.2%
BoW + Logistic Regression	93.9%
Recurrent Neural Network (RNN)	91.7%
Convolutional Neural Network (CNN)	94.5%
Recurrent Convolutional Neural Network (RCNN)	**98.0%**



Neural Networks vs Traditional Approach

When we compare neural network approaches (RecursiveNN, CNN, and RCNN) to the widely used traditional methods (e.g., BoW+LR), the experimental results show that the neural network approaches outperform the traditional methods for all four datasets. It proves that neural network based approach can effective compose the semantic representation of texts. Neural networks can capture more contextual information of features compared with traditional methods based on BoW model, and may suffer from the data sparsity problem less.

RCNN vs CNN and RNN

When we compare RCNN with CNN as well as RNN, we found that RCNN out performs both the other techniques and provided better results. This illustrates that convolution based framework is more suitable for constructing the semantic representation of texts compared with the other neural networks technique and the main advantage lies in the usage of the max-pooling layer that captures more discriminative features through it.

Also it is to be noted that RCNN produced superior results on the dataset with a small number of iterations. For eg., CNN took almost 15 iterations to produce the best accuracy, whereas RCNN produced extreme results within 10 iterations.

From the above discussions, we can clearly say that RCNN is best suitable for text classification.

Conclusion

We experimented recurrent convolutional neural networks to text classification. Our model captures contextual information with the recurrent structure and constructs the representation of text using a convolutional neural network. The experiment demonstrates that our model outperforms CNN and RecursiveNN for the given dataset.

References

[1] Siwei Lai; Liheng Xu; Kang Liu and Jun Zhao, Recurrent Convolutional Neural Networks for Text Classification, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.
https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/download/9745/9552

[2] Yoon Kim, Convolutional Neural Networks for Sentence Classification.
http://emnlp2014.org/papers/pdf/EMNLP2014181.pdf

[3] http://keras.io/models/

[4] Nitish Srivastava, Geofferey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdin, Dropout: A Simple Way to Prevent Neural Networks from Overfitting
https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

[5] http://cs231n.github.io/convolutional-networks/

[6] http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

[7] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[8] https://github.com/kjw0612/awesome-rnn
[9] http://www.fit.vutbr.cz/research/groups/speech/servite/2010/rnnlm_mikolov.pdf