IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 5, Issue 9, September 2018

Toxic Comment Classification Using Neural Networks and Machine Learning

Revati Sharma[1], Meetkumar Patel[2]

Computer Engineering, Ahmedabad Institute of Technology, Ahmedabad, India[1,2]

Abstract: A cornucopia of data is developed through conversations, interactions of humans online. This scenario has contributed considerably well to the quality of human life but it also involves prodigious dangers as online text communications with high toxicity quality cause individual assaults, online provocation and harassing practices. This has activated both industrial and research network over the most recent couple of years while there are a few attempts to distinguish a proficient model for online toxic comment classification and prediction. Be that as it may, these means are still in their earliest stages and new methodologies and structures are required. On parallel, the information blast that shows up always, makes the development of new machine learning computational apparatuses for overseeing this data, a basic need. Gratefully progresses in big data management, hardware and cloud computing administration permit the advancement of Deep Learning approaches showing up exceptionally encouraging execution up until now. Recently the use of Convolutional Neural Networks and Recurrent Neural Networks have been approached for computational purposes for the text classification systems. In this work, we utilize this way to deal with finding toxic comments, remarks in an extensive pool of records given by a current Kaggle's competition with respect to Wikipedia's talk page edits which has divided the level of toxicity into 6 labels: toxicity, severe toxicity, obscenity, threat, insult or identity hate.

Keywords: Long-Short Term Memory, Convolutional Neural Network, Text mining, Word Embedding, Toxic text classification, Text classification

I. INTRODUCTION

Every day, we get a tremendous amount of short content data from the blast of online correspondence, web-based business and the utilization of advanced gadgets [1]. This volume of data requires text mining apparatuses to carry out the various report tasks in an opportune and suitable way. Text Classification is a great point for NLP(Natural Language Processing) and a basic segment in numerous applications, for example, web seeking, data filtering, topic categorization, and opinion or sentiment analysis [2]. Text classification can be characterized essentially as: Given a set of documents D and a set of classes (or labels) C, define a function F that will assign a value from the set of C to each document in D [3]. Accordingly, a tremendous pool of machine learning strategies have been connected for text classification in different data types with acceptable results. These days, data is normally and usually in short messages such as interpersonal organizations, news in website pages, gatherings et cetera. Be that as it may, short messages accumulations, having the impediment of short length documents, end up represented by sparse matrices, with minimal co-occurrence or shared context.

As a result, defining efficient similarity measures is not straightforward, especially regarding the most popular word-frequency based approaches, resulting in degrading performance [4].

Text arising from online interactive communication hides many hazards such as fake news, online harassment and toxicity [5]. The obscene, nasty, obnoxious kind of comments or texts which do not only result in online verbal violence viz., sexual harassment, personal attacks of one reputed organization to other organizations or personalities to denigrate or defame them in a disrespectful and malignant way, bullying and also racisms in hostile language and are even sometimes life threats which can be found on blogs, hate sites and comment sections etcetera online social platforms. For instance, cyber-bullying affects an individual feeling intimidated and embarrassed about oneself.

The online disinhibition effect can have an effect on one's job security and future employment opportunities. Indicatively, the Wikimedia foundation found that 54% of those who had experienced online harassment expressed decreased participation in the particular project which occurred [6]. Also, a 2014 Pew Report highlights that 73% of adult internet users have seen someone harassed online, and 40% have personally experienced it [5]. Although, there are efforts to enhance the safety of online environments based on crowd-sourcing voting schemes or the capacity to denounce a comment, in most cases these techniques are inefficient and fail to predict a potential toxicity [7]. Automatic virulent comment identification and prediction in real time is of preponderant importance since it'd enable the hindrance of many adverse effects for net users.

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 5, Issue 9, September 2018

Towards this direction, the work of Wulczyn et al. [6] planned a methodology that blends crowdsourcing and machine learning to research personal attacks at scale. Recently, Google and Jigsaw propelled a venture known as Perspective [7], which uses machine learning to mechanically find on-line insults, harassment, and abusive speech. Perspective is an API (www.perspectiveapi.com) that permits the developers to use the unhealthful detector running on Google's servers, to spot harassment and abuse on social media or a lot of expeditiously filtering vituperation from the comments on a news website. The API uses machine learning models to get the perceived impact a comment might need on a voice communication. Developers and publishers will use this score to convey real-time feedback to commenter or facilitate moderators do their job, or enable readers to a lot of simply realize relevant data.

A main limitation of those models is that they're not as reliable because it ought to which sometimes the percentage of toxicity is not ascertained. The solution presented here is based upon the Kaggle competition which i held as a platform globally for modeling through predictions and analysis for a particular problem sector where data scientists, mathematicians especially statisticians, data miners come together to compete by performing analysis on the data or information models provided by genuine companies and users as an open source to find out the best working systems of highest accuracy ratings.

Natural Language Processing (NLP) has profited significantly from the resurgence of deep neural systems (DNNs), because of their high performance with less need of designed highlights. There are two fundamental DNN structures: convolutional neural networks (CNN) (LeCun et al., 1998) and recurrent neural system (RNN) (Elman, 1990). Gating systems have been produced to ease a few confinements of the fundamental RNN, bringing about two winning RNN composes: long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) as well as gated recurrent unit (GRU) (Cho et al., 2014). As a rule, CNNs are various leveled i.e hierarchical and RNNs successive i.e sequential designs. By what means should we pick between them for preparing language or the textual conversation? In view of the portrayal "various leveled (CNN) versus successive (RNN)", it is enticing to pick a CNN for classification tasks like sentiment analysis and classification since opinion is normally controlled by some key expressions; and to pick RNNs for a sequence demonstrating undertaking like language displaying as it requires adaptable displaying of context reliances. Be that as it may, current NLP writing does not bolster such a reasonable conclusion. For instance, RNNs perform well on sequence modeling tasks (Tang et al., 2015); and Dauphin et al. (2016) as of late demonstrated that gated CNNs beat LSTMs on speech demonstrating undertakings, even in spite of the fact that LSTMs had for some time been viewed as more qualified.[8] (i) CNNs and RNNs provide complementary information for text classification tasks. Which architecture performs better depends on how important it is to semantically understand the whole sequence. (ii) Learning rate changes performance relatively smoothly, while changes to hidden size and batch size result in large fluctuations.[8]

II. STATE-OF-ART OF THE EXISTING SYSTEMS

One received approach is the one model and numerous yield layers. Otherwise called multi-assignment learning, this approach would have one input layer, one arrangement of hidden layers, and one yield layer for each name. It is not quite the same as the One Model Per Label (OMPL) [9]approach, where for each label i.e. the mark, we prepare one model to recognize if a perception has a place with that label or not (e.g. foul or not revolting). This approach would require huge preparing time for each mark. It is additionally unique in relation to OMPL with exchange learning, which is, as its name uncovers it, like OMPL. Be that as it may, rather than preparing each model starting with no outside help, one could prepare a constructed display in light of a particular name, and clone it as the reason for future models. This approach would require huge preparing time for the first show, however moderately small preparing time for extra names. In any case, conveying this model would in any case require taking care of numerous model pipelines.

Recently, Convolutional Neural Networks (CNN) are being applied to text classification or natural language processing both to distributed as to discrete embedding of words [10, 12], without using syntactic or semantic knowledge of a language [11]. Also, a recurrent CNN model was proposed recently for text classification without human-designed features [13] by succeeding to outperform both the CNN model as well as other well-established classifiers. Their model captures contextual information with the recurrent structure and constructs the representation of text using a convolutional neural network. Meanwhile, CNN has been shown an alternative mechanism for effective use of word order for text categorization [11]. An effective CNN based model using word embeddings to encode texts is published recently [14]. It uses semantic, embeddings, sentiment embeddings and lexicon embeddings for texts encoding, and three different attentions including attention vector, LSTM (Long Short Term Memory) attention and attentive pooling are integrated with CNN model. To improve the performance of three different attention CNN models, CCR (Cross-modality Consistent Regression) and transfer learning are presented. It is worth noticing that CCR and transfer learning are used in textual sentiment analysis for the first time. Finally, some experiments on two different datasets demonstrate that the proposed attention CNN models achieve the best or the next-best results against the existing state-of-the-art models[16].

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 5, Issue 9, September 2018

III. NEURAL NETWORKS

Whenever Neural Networks are applied to the NLP depictions of text, it only illustrates the occurrence of the words within a document and for this purpose, it involves only the vocabulary of known words and how many times they have occurred. Any information about he what is the position or the structure of these words or which or where in the document they occur is not taken into account. A BoW (Bag-of-Words) display is an elective method for extracting features from content that can later be utilized for any sort of investigation for example, classification. This portrayal of text depicts the event of words inside a document and for this reason it just includes a vocabulary of known words and their relating proportion of the nearness. Any data about the order or structure of words in the report is disposed of in this case. The model is just worried about the event of words in the report and not where in the document they happen. the models or systems which use these techniques and approaches for text classification and categorization give out a great execution. Albeit the hypothetical basic and commonsense proficiency, a BoW show includes a few specialized difficulties. The initial phase in an average content investigation strategy is to build the Document-Term-Matrix (DTM) from input documents. This is finished by vectorizing documents making a guide from words to a vector space. In the following stage a model for either directed or unsupervised learning is connected. Following, we give the points of interest of the DTM development for the toxic comment classification issue at hand.

IV. NEURAL NETWORKS FOR TEXT CLASSIFICATION

We start the strategy by creating the vocabulary of words. Here we pick interesting words showing up in all documents disregarding case, punctuation, numbers and regular words that don't contain much data (called stop words). At that point as opposed to scoring words in view of the quantity of times each word shows up in a report we utilize the Term Recurrence - Inverse Document Frequency (TF-IDF) approach [15]. Along these lines we stay away from the issue of high scoring by commanding words that as a rule don't contain 'informing content'. To accomplish this, the recurrence of words is rescaled by how frequently they show up in all reports by punishing most visit words over all documents. The subsequent scores are a weighting showing significance or intriguing quality of words. TF-IDF have turned out to be exceptionally fruitful for classification tasks specifically by uncovering the distinctions among regular gatherings. In the last advance of preprocessing we bargain with the sparsity issue. sparse representations are typically more hard to display both for computational reasons and likewise for data reasons, as so little data need to be separated from such a huge illustrative space. Here we dispose of terms with higher that 99% sparsity overseeing likewise to decrease the dimensionality of the DTM altogether.

As a result following steps were performed for development for text classification: (i)embedding layer which is efficient for better results is developed that maps vocab indices to dimensions. (ii)a max pooling which will extract features from the embeddings GlobalMaxPooling1D() on the comments which were embedded. (iii)the final result being project onto 6-unit output layer squashing it with sigmoid layer.

V. CONVOLUTIONAL NEURAL NETWORK

Delineating the Convolutional Neural Networks for the classification bringing in light the process description for particular text classification. These are multi-staged Neural Network architectures for the task of classification:

1. Convolutional Layers, are major components of the CNNs. A convolutional layer consists of a number of kernel matrices that perform convolution on their input and produce an output matrix of features where a bias value is added. The learning procedures aim to train the kernel weights and biases as shared neuron connection weights[16].
2. Pooling Layers, being also integral part of the CNNs, its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layers make a sub sampling to the output of the convolutional layer matrices combing neighbouring elements. Pooling layer operates on each feature map independently. The most common approach used in pooling is max pooling which takes the maximum value of the local neighborhoods.
3. Embedding Layer is a unique segment of the CNNs for text classification issues. The motivation behind an embedding layer is to change the content contributions to a appropriate frame for the CNN. Here, each expression of a text report is changed into a dense vector of fixed size.
4. Fully-Connected Layer is a classic Feed-Forward Neural Network (FNN) hidden layer. It can be interpreted as a special case of the convolutional layer with kernel size 1×1. This type of layer belongs to the class of trainable layer weights and it is used in the final stages of CNNs.[16]

The training of CNN relies on the Back Propagation (BP) training algorithm [17]. The requirements of the BP algorithm is a vector with input patterns x and a vector with targets y, respectively. The input xi is associated with the

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 5, Issue 9, September 2018

output oi . Each output is compared to its corresponding desirable 2 target and their difference provides the training error. Our goal is to find weights that minimize the cost function[16]

$$Ew = 1\ n\ P\ p{=}1\ NL\ j{=}1\ o\ L\ j,p - yj,p\ 2$$

where P the number of patterns, $o\ L\ j,p$ the output of j neuron that belongs to L layer, NL the number of neurons in output layer, yj,p the desirable target of j neuron of pattern p. To minimize the cost function Ew, a pseudo-stochastic version of SGD algorithm, also called mini-batch Stochastic Gradient Descent (mSGD), is usually utilized [18].

VI. CNN FOR TEXT CLASSIFICATION

The CNN have been widely applied to image classification problems due to their inner capability to exploit the two statistical properties that characterize image data, namely 'local stationarity' and 'compositional structure' [19].Local stationarity structure can be translated as the trait of a picture to show reliance between neighbouring pixels that is sensibly steady in nearby picture areas. Local stationarity is abused by the CNNs' convolution administrator.
For text classification issues the unique crude information likewise introduces the previously mentioned factual properties in view of the way that neighboring words in a sentence display reliance, be that as it may, their processing isn't clear or straightforward. The parts of a picture are just pixels represented to by integer values qualities inside a particular run. Then again, the parts of a sentence (the words) must be encoded before encouraged to the CNN [15]. For this purpose we may utilize a vocabulary. The vocabulary is built as an index containing the words that show up in the set of document texts, mapping each word to an integer value in the vicinity of 1 and the vocabulary size. The fluctuation in reports length (number of words in a report) should be tended to, as CNNs require a steady information dimensionality. For this reason, the padding procedure is received, loading up with zeros document matrix keeping in mind the goal to achieve the most extreme length among all documents in dimensionality.

In the subsequent stage the encoded reports are changed into matrices for which each line corresponds to single word. The created matrices go through the inserting layer where each word (row) is changed into a low-dimension portrayal by a dense vector [1]. The strategy at that point keeps following the standard CNN philosophy. At this point, it merits specifying that there are two methodologies for the low-dimension portrayal of each word. The first approach called 'randomized' which is accomplished by setting a distribution over the word, creating a dense vector with settled length. The values of the vectors are tuned by means of the training procedure of the CNN. The other extremely well known approach additionally assessed here is to utilize settled dense vectors for words, which have created in view of word implanting techniques such as the word2vec [22] and GloVe [23]. All in all the word implanting techniques have been prepared on a vast volume dataset of words creating for each word a dense vector with a particular dimension and fixed qualities. The word2vec implanting strategy for instance, has been prepared on 100 billion words from Google News creating a vocabulary of 3 million words. The embedding layer coordinates the information words with the fixed dense vector of the pre-prepared embedding strategies that have been chosen. The estimations of these vectors don't change amid the preparation procedure, except if there are words not effectively incorporated into the vocabulary of the embedding strategy in which case they are instated haphazardly.

VII. RECURRENT NEURAL NETWORKS

RNN takes words in a sentence in a sequential order and is able to learn the long-term dependencies of texts rather than local features[24].

The long-term dependencies learned by RNN can be viewed as the sentence level representation. The sentence-level representation is taken to the fully connected network and the softmax output reveals the classification result. Unlike feed forward neural networks, RNNs are able to handle a variable-length sequence input by having a recurrent hidden state whose activation at each time is dependent on that of the previous time[24].

To train the RNNs we update the weights bby figuratively moving backwards in time changing hence updating the weights which is called the Back Propagation Through Time (BPTT). For example, in order to calculate the gradient at t=4 we would need to back propagate 3 steps and sum up the gradients. As information length develops, RNNs trained through BPTT move toward becoming unfit to figure out how to interface data which results in problem called as Vanishing/Exploding gradient problem. This is precisely where Long Short-Term Memory neural networks (LSTMs) go ahead stage. They are expressly intended to keep away from the long haul reliance issue. The center thought behind LSTMs is the cell state, which is pretty much like a transport line. It goes from end to end of the whole chain, with just a few minor straight communications. Subsequently, it is simple for data to simply stream along it without being changed.

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 5, Issue 9, September 2018

VIII. LSTMS FOR TEXT CLASSIFICATION

Sequence modelling being essential for better training of model through RNN algorithms-LSTM can be built by the framework keras sequential model with Tensor Flow running at the backend. The sole reason for using the Kers is that it is built fully in python codes and hence easy to debug and modify if there is a need to combine various standalone modules such as neural layers, activation functions, optimizers, regularization schemes into one to develop new modules. For developing the sequential model , pre-processed and tokenized data is required which ia again handled by the APIs provided by Keras Framework. Pre-processing the text data is done in some successive steps with keras:(i)first the given texts or the sentences are split into words in a list for a meaningful sequence by text_to_word_sequence which by default converts text to all lower case splitting out the punctuations for further ease.(ii)once these sequence of words is generated, the function provided by keras : hashing_trick() frees the limitation of word tracking and counting by tokenizing the text and hence generating an integer encoded version of the document.

The word embedding strategies used here is word2Vec being one of the top techniques coded in python to provide words as inputs into the neural networks which majorly overcomes or outweighs the shortcomings simple vectorization of bag of words model which is just production of vectors to given words, but the word to Vec technique helps maintain a semantic relationship between words of text i.e. how a word can be related to other by generating dense matrices or vectors of integer encoded texts based on the distance between most related word being the least and words with least or no semantic relationship being farther. This relationship however is lost in the simple vectorization techniques. We load this embedding matrix into an Embedding layer using embedding index dictionary and word index matrices.

An Embedding layer should be fed sequences of integers, i.e. a 2D input of shape (samples, indices). These input sequences should be padded so that they all have the same length in a batch of input data (although an Embedding layer is capable of processing sequence of heterogeneous length, if you don't pass an explicit input_length argument to the layer).

All that the Embedding layer does is to map the integer inputs to the vectors found at the corresponding index in the embedding matrix, i.e. the sequence [1, 2] would be converted to [embeddings[1], embeddings[2]]. This means that the output of the Embedding layer will be a 3D tensor of shape (samples, sequence_length, embedding_dim).

LSTMs having the cell states for controlling the flow of information i.e which to accept and sink in completely and which to let go or get rid off that are controlled and protected by the three gates composed of the sigmoid layer whose outputs 1 and 0 depicts the herenow mentioned instructions respectively. For understanding and predicting the next word based on the previous ones, the gender pattern must be learned by the cell states so that relevent and correct pronouns are used. For this the cell states with the help of the sigmoid layer also the input gate layer, decided what values need to be updated to the new subject replacing the old ones. Tanh layer updates the memory.

Using the keras framework, we will be developing : (i)a bidirectional LSTM layer, (ii)one-dimensional global-max pooling layer,(iii) dense 50 neuron layer for better memory powers, (iv) two 0.1 dropouts to prevent overfitting, (v)a last dense 6-neuron output layer. Simple recurrent neural networks' additive operations lag in terms of the flexibility and results when compared to the multiplicative operations between word embedding through gate structures.

IX. OVERALL RESULTS

Table I Accuracy performance of different algorithms (in %)

Epochs	NN	CNN	LSTM
3	0.9884	0.9801	0.9840
5	0.9860	0.9803	0.9870
7	0.9891	0.9804	0.9877

Table II Time performance of different algorithms (in s)

Epochs	NN	CNN	LSTM
3	2.753	235.8	168.5
5	2.800	211.1	192.3
7	9.703	235.0	198.0

DOI 10.17148/IARJSET.2018.597

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 5, Issue 9, September 2018

CONCLUSION

There have been ceaseless trials experimenting and computing the presence of toxicity of various kinds on the online platforms including the micro and macro blogging sites by the industries as well as the research communities for an efficient model that detects and predicts the online toxic comments. This holds importance in the research field due to the tremendously growing online interactive communication among users. This work is dedicated to finding the best possible optimum solutions for online toxic comment classification which further classifies the toxic comments into 6 labels provided by the datasets on kaggle platform. These datasets are from the wikipedia talk page edits. Using the word embedding techniques and also comparing the primary level neural network algorithms results with intricate Convolutional Neural Networks and Recurrent Neural Network compose: LSTM (Long-Short Term Memory) results , the obtained analysis show that the LSTM perform better than the CNNs in terms of both the accuracy and time performance given the same number of epoch and hence are preferable to use rather than CNN with word-level embeddings. More accurate and promising results can be obtained by further improving the word embeddings by using more finely pre-processed data as well as advancements in developing the proposed LSTM systems.

REFERENCES

[1]. Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, and Shifu Bie. 2014. Short Text Classification: A Survey. Journal of Multimedia 9, 5 (2014).
[2]. Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In Mining text data. Springer, 163–222.
[3]. M Ramakrishna Murty, JVR Murthy, and Prasad Reddy PVGD. 2011. Text Document Classification based-on Least Square Support Vector Machines with Singular Value Decomposition. International Journal of Computer Applications 27, 7 (2011).
[4]. Xiaojun Quan, Gang Liu, Zhi Lu, Xingliang Ni, and Liu Wenyin. 2010. Short text similarity based on probabilistic topics. Knowledge and information systems 25, 3 (2010), 473–491.
[5]. Maeve Duggan. 2014. Online harassment. Pew Research Center.
[6]. Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 1391–1399.
[7]. Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving Google's Perspective API Built for Detecting Toxic Comments. arXiv preprint arXiv:1702.08138 (2017).
[8]. Comparative Study of CNN and RNN for Natural Language Processing Wenpeng Yin, Katharina Kann, Mo Yu and Hinrich Schutze ¨ CIS, LMU Munich, Germany IBM Research, USA. arXiv:1702.01923v1 [cs.CL] 7 Feb 2017.
[9]. B. Herger, "Detecting toxic comments with multi-task deep learning," 2018. https://www.hergertarian.com/ detecting-toxic-comments\protect\ discretionary{\char\hyphenchar\ font}{}{}with-multitask-deep-learning/
[10]. Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. 69–78.
[11]. Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. arXiv preprint arXiv:1412.1058 (2014).
[12]. Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014).
[13]. Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification.. In AAAI, Vol. 333. 2267–2273.
[14]. Zufan Zhang, Yang Zou, and Chenquan Gan. 2018. Textual sentiment analysis via three different attention convolutional neural networks and cross-modality consistent regression. Neurocomputing 275 (2018), 1407–1415.
[15]. Anand Rajaraman and Jeffrey David Ullman. 2011. Mining of Massive Datasets. Cambridge University Press. https://doi.org/ 10.1017/CBO9781139058452.
[16]. Convolutional Neural Networks for Toxic Comment Classification,Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, Vassilis P. Plagianakos. arXiv:1802.09957v1 [cs.CL] 27 Feb 2018
[17]. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradientbased learning applied to document recognition. Proc. IEEE 86, 11 (Nov 1998), 2278–2324.
[18]. LÃľon Bottou. 1998. On-line learning and stochastic approximations. In In On-line Learning in Neural Networks. Cambridge University Press, 9–42.
[19]. Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. CoRR abs/1506.05163 (2015). arXiv:1506.05163 http://arxiv.org/abs/ 1506.05163.
[20]. Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. J. Mach. Learn. Res. 12 (Nov. 2011), 2493–2537. http://dl.acm.org/citation.cfm?id=1953048. 2078186
[21]. Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. 1019–1027.
[22]. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In Advances in Neural Information Processing Systems 26, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119.
[23]. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In In EMNLP.
[24]. Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts Xingyou Wang1 , Weijie Jiang2 , Zhiyong Luo3. http://www.aclweb.org/anthology/C16-1229

BIOGRAPHIES

Revati Sharma is currently pursuing an undergraduate degree of computer engineering from Ahmedabad Institute of Technology of Gujarat Technological University. She has a great interest in machine learning, deep learning, Natural Language Processing, artificial intelligence and Data Science and Analytics.

Meetkumar Patel is graduated from Ahmedabad Institute of Technology of Gujarat Technological University. He has profound understanding of the machine learning, deep learning and Natural Language Processing.