Paraphrase Detection using Dependency Tree Recursive Autoencoder

Deepak Singh Rana, and Pankaj Kumar Mishra

***Abstract*—An architecture is proposed based on recursive autoencoder for paraphrase detection. The proposed architecture embeds the semantic information by using word representations generated from the neural network language model and the syntactic information by implementing the dependency tree over the recursive autoencoder, where dependency tree reveals the syntactic information of the given sentence in recursive form. The proposed architecture is tested on the MSRP dataset for paraphrase detection and the results are above the baseline. The proposed system reached a moderate accuracy and F1 score for the paraphrase detection test on MSRP dataset.**

***Index Terms*—Autoencoder, Recursive Autoencoder, Paraphrase Detection, Dependency tree.**

I. Introduction

THE paraphrase is a process of changing or arranging the words in a given sentence without changing its meaning. Paraphrase detection is a task of identifying whether the given pair or more sentences are paraphrase or not. Paraphrase detection plays an important role in many tasks that require semantic analysis related to the real world applications as given in ([16] [18] [29]). Paraphrase detection is a task which requires computation and comparison of semantic information of the sentences. Therefore, there is a requirement of such a method or model that can extract the semantic information from the given sentences. Semantic analysis basically deals with word and sentence meaning. The main focus of Semantic analysis is on combining the meaning of words to get a sentence or phrase meaning. Most of the problems in semantic analysis require the sentence or phrase meaning for solving them. Now the problem is to get an appropriate representation of words that encapsulates the semantic information of words in itself and combining the semantic information of words in phrases of the sentence, so that the combined representations be able to represent the semantic information of phrases. The problem of getting an appropriate representation of words is solved by the language model.

Language model encapsulates the semantic information of the word in an appropriate representation. The language model that trains themselves on raw text in unsupervised are mostly Neural Network Language models (NNLM). The idea of the neural language model was first introduced by Yoshua Bengio [1]. NNLM learns word embedding into an n-dimensional vector space and to use these vector in predicting the likelihood of a word in a given context. Paraphrase detection is the task of similarity identification, where it can be very useful in various fields of NLP, like similar information retrieval where for a given input text we have to find similar text out of the given document, Comprehensive question answering is the task of finding answers of questions based on the given passages or etc. The Paper is organized as follows. Section II shown the related works and the base models for the proposed model. Section III presented the discussion on the proposed model. In section IV the setup and various components used for experimental and section V presents the results and discussions for the experiment. Finally, section VI presents the conclusions.

This paper is submitted for review on 24/02/2018.

Deepak Singh Rana is Postgraduate (M. Tech.) from Department of Computer Engineering, G.B.P.U.A.T, Pantnagar, Uttarakhand, India (e-mail: deepaksinghrana049@gmail.com).

Pankaj Kumar Mishra Assistant professor from Department of Computer Engineering, G.B.P.U.A.T, Pantnagar, Uttarakhand, India (e-mail: pkmishra.cs@gmail.com).

II. Related work and Base model

Here we outline various works done for paraphrase detection on MSRP dataset [5] [24] and the base models, i.e. Recursive Autoencoder and Recursive neural network. We also showed the point of exploitation in the Recursive Autoencoder model.

A. *Related work*

Work on paraphrase detection is majorly started when the MSRP dataset for testing paraphrase detection system was developed. There were many systems developed for performing paraphrase detection testing. The Standard Machine translation (MT) evaluation methods (BLEU, NIST, WER, and PER) was used for paraphrase identification [8]. The Support Vector Machine, k-Nearest Neighbor, and Maximum Entropy based machine uses POS tag and semantic similarity for paraphrasing identification [16]. In [18] by using two corpus-based and six knowledge-based measures of similarity measure were used to identify semantic similarity. Semantic similarity is a good parameter while [23] used two-phase framework. The first phase identifies the sentence similarity and the second phase identifies the dissimilarity (if any), in the parts of pair sentence left after removing the similar parts. The judgment of this model was based on both similar and dissimilar parts identification. A machine learning approach that filters out the false or not paraphrase pair of sentences using N-gram Overlap, Dependency Relation Overlap, Dependency Tree-Edit Distance and Surface Features

[29]. A method that uses two metric, i.e. semantic similarity and string similarity for the text similarity identification [9]. Fernando and Stevenson in 2008 [7] proposed a system which adopted the similarity matrix approach that takes the advantage of all similarity scores between all possible word pairs formed in-between the given pair of sentences. A generative model for the generation of the similar sentence and a probabilistic method to infer whether the given two sentences hold the paraphrase relationship or not [32]. Blacoe and Lapata in 2012 presents that the problem of paraphrase detection was resolved by using the compositional meaning of phrases and sentences [2]. The compositional meaning for phrase and sentences was generated by using word representations and distributional methods. Automated metrics developed for machine translation evaluation tasks were useful for paraphrase recognition [17]. They used the meta-classifier trained using machine translation metrics. Wang and others 2016 proposed a model to consider both the similarities and dissimilarities by breaking down and forming lexical semantics over sentences. As most methods mainly focused on the similar part while ignoring the non-similar parts, where dissimilar parts also contain some information about sentences [30].

The introduction NNLM [1] for generation of word vector representation used in semantic role labelling, Part of Speech, Chunking and Name Entity Recognition was shown the state-of-art results [4], which results in the increase of unsuperviousness and accuracy of various NLP related problems. The NNLM results in development of various word vector representation based models for paraphrase detection ([25] [14] [20] [13] [3] [31]). A term-weighting metric called TF-KLD [10] to convert high-dimensional bag-of-words representation to low-dimensional latent representation, these low-dimensional latent representation used for semantic similarity detection. The TF-KLD based model result into the highest accuracy of paraphrase detection test on MSRP dataset. But this model used traditional feature extraction. Socher and other in 2011 proposed a Recursive Auto Encoder (RAE) [26] for learning the vector representation for phrases by embedding the semantic information of words together using dependency tree and used these phrase vectors for semantic similarity measure.

The system described in this paper is based on the Recursive autoencoder (RAE) [22] [25] [26] for generation of phrase vectors and these phrase vectors are used in the generation of similarity matrix which in turn used for paraphrase detection. The early RAE-based [25] model used the binary dependency tree for phrase vector generation while here in this paper, we extend the approach by using a variable dependency tree (variable dependency was defined in DT-RNN [27]. Next two sections describe the base methods used in development of proposed model.

B. Recursive Autoencoder

Recursive autoencoder (RAE) [25] aims to generate the vector representation of variable-sized phrase by semi-supervised training and these generated phrase vectors then used for various tasks for semantic analysis. For generation of these phase vectors RAE uses the word vector representation generated by unsupervised neural language model CBOW and Skip-gram [19]. These word vector representations are stacked together into a word vector matrix $M \epsilon R^{n \times |V|}$, where n is the vector dimension and |V| is the vocabulary (consists of distinct words).

Now a given sentence of *m* words is represented as a word vector set ($x_1, x_2, x_3, ..., x_m$), where x_i is the vector represent of i^{th} word. For each word there is an index value k which associates the word to its vector representation in the word vector matrix. To extract a vector representation for each word from the word vector matrix we use a lookup operation. This look-up operation can be presented as a projection layer which takes a binary vector b of size |V| consists of zero at all positions except k^{th} (consist one),

$$x_i = Lb_k \epsilon R^n \tag{1}$$

We have a sentence word vector representation and its binary tree form [25] [26] which can be presented as a branching dependency of parent and child like $y \rightarrow x_1x_2$, where x_1 and x_2 are input word vector and y is the nonterminal hidden node. For example, in Fig. 1 we have followed branching dependency: $h_2 \rightarrow x_2x_3$, $h_1 \rightarrow x_1h_2$.

Now for applying neural network on a pair of children, the hidden representation h_j should have same the dimension for word vector x_i.

Given, the tree structure Fig. 1 we compute the parent vector representation h from children x_1, x_2:

$$h = f(w_e[x_1; x_2] + b_e) \tag{2}$$

Where $w_e \epsilon R^{nx2n}$ is a weight matrix of multiplied to the concatenation of the pair of children, b_e is the bias term and f is the activation function.

To find how well the parent vector representing its children by reconstructing the children through reconstruction layers of the RAE:

$$[x_1'; x_2'] = f(w_d h + b_d) \tag{3}$$

During RAE training the goal is to minimize the reconstruction error of the input pair. We compute the Euclidean distance between the input vector and its reconstructed vector for every pair:

$$E([x_1; x_2]) = \frac{1}{2} |[x_1; x_2] - [x_1'; x_2']|^2 \tag{4}$$

This whole computing of parents and reconstruction of child process called encoding and decoding as presented in Fig. 1. This way encoding and decoding is done for all hidden vector nodes (parent node) in the tree. The goal of the RAE is to minimize the reconstruction error by changing the weight parameters i.e. w_e and w_d. Finally the values of hidden nodes are treated as a phrase vector representation.

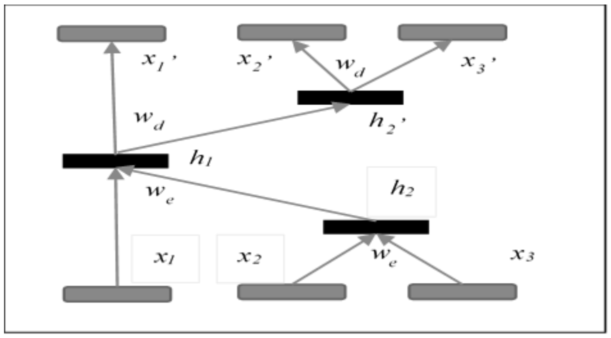

Fig. 1: Illustration RAE example.

C. Dependency-tree Recursive Neural Network

The DT-RNN [27] model, computes parent vector (phrase vector) for every child (word vector) using an activation function (*f*) and weight (*W*) in a bottom up fashion, for example fig 2 (b):

$$h_1 = f(w_v x_1),\ h_3 = f(w_v x_3) \quad (5)$$
$$h_4 = f(w_v h_4 + w_{l1} h_3) \quad (6)$$
$$h_2 = f(w_v x_2 + w_{l1} h_1 + w_{r1} h_4) \quad (7)$$

Where $W = (w_{lk}, \ldots, w_{l1}, w_v, w_{r1}, \ldots, w_{lk},)$ is the weight parameters of the DT-RNN model.

By using Dependency tree based model (DT-RNN), the vector generated for non-terminal node able to embed the syntactic details of sentence in the phrase vectors. Which in return may provide better performance on semantic tasks.

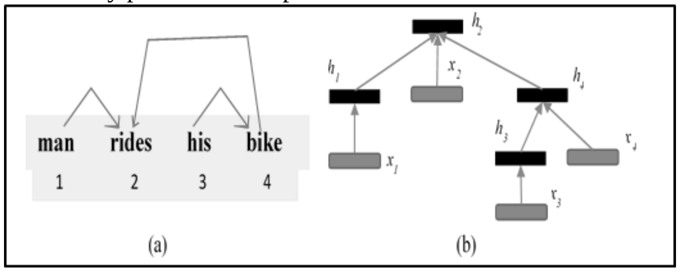

Fig. 2: a) Dependency of a sentence. b) Its DT-RNN tree structure.

D. Point of change

In the RAE-based approach the syntactic information of a sentence is incorporated in representation by using parsed tree of a sentence, where the child-nodes of any intermediate node is fixed i.e. two. This is the point of change where the fixed child-node structure is replaced by the variable (i.e. 1 to n) child-node structure in the proposed work by using the dependency tree structure of the DT-RNN.

III. Proposed model: Dependence-tree Recursive Autoencoder

The Dependence-tree recursive autoencoder (DT-RAE) is derived from RAE model and DT-RNN model both is explained above. In the DT-RAE model the implementation of dependency tree structure from DT-RNN over RAE is done to embed better syntactic information on the vector representation. The DT-RAE takes a dependency tree and the word vectors as input and generate the phrase vector. The structural of DT-RAE for example Fig. 2 (a) is given in Fig. 3.

DT-RAE is divided into two phases, i.e. encoding and decoding. Encoding phase encodes the tree like structure into a single compressed vector while decoding phase reconstructs the tree like structure from the compressed single vector example Fig. 3. The encoding and decoding phase of Fig. 3 DT-RAE structure is explained below.

Encoding phase: In this phrase the intermediate or phrase vectors are calculated using input vectors by using following:

$$h_1 = f(w_v x_1),\ h_3 = f(w_v x_3) \quad (8)$$
$$h_4 = f(w_v h_4 + w_{l1} h_3) \quad (9)$$
$$h_2 = f(w_v x_2 + w_{l1} h_1 + w_{r1} h_4) \quad (10)$$

Where x_1, x_2, x_3, x_4 are the input vectors, h_1, h_2, h_3, h_4 are intermediate vectors and w_{l1}, w_v, w_{r1} are weight matrixes of encoding phase.

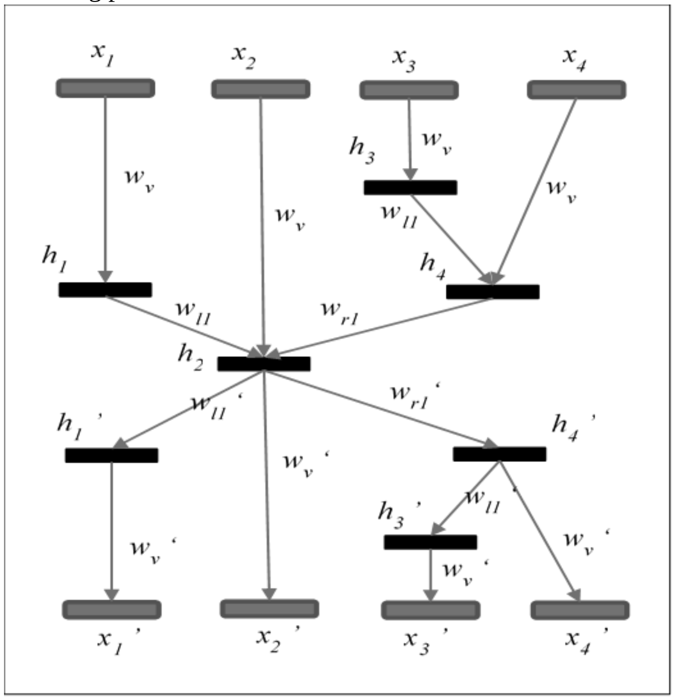

Fig. 3: Proposed DT-RAE structure with encoding and decoding phrase of Fig. 2(a).

Decoding phase: In decoding phrase the intermediate or phrase vector and input vector are reconstructed by using the following:

$$h_1' = f(w_{l1}' h_2),\ h_4' = f(w_{r1}' h_2), x_2' = f(w_v' h_2) \quad (11)$$
$$h_3' = f(w_{l1}' h_4'), x_4' = f(w_v' h_4'), x_1' = f(w_v' h_1') \quad (12)$$
$$x_3' = f(w_v' h_3') \quad (13)$$

Where x_1', x_2', x_3', x_4' are the reconstructed input vectors h_1', h_2', h_3', h_4' are reconstructed intermediate vectors and w_{l1}', w_v', w_{r1}' are transposes of encoding weight matrixes.

After encoding and decoding phase the input vectors and reconstructed input vectors are used for reconstruction error calculation.

$$E([x_1; x_2; x_3; x_4]) = \frac{1}{2} |[x_1; x_2; x_3; x_4] - [x_1'; x_2'; x_3'; x_4']|^2 \quad (14)$$

The reconstruction error is then back propagated by using BPTS algorithm [11] to update the weights of the DT-RAE.As the weights are updated the DT-RAE is able to learn a generalized vector representation for phrases of sentences. The trained DT-RAE is then used for the phrase vector generation of the given sentence.

IV. Experiment setup

This section consists of the description of dataset, models and model parameters used in the proposed work. The experiment setup is prepared for paraphrase testing using an MSRP dataset. This section also describes various features or parameters used in testing.

A. Dataset description

1) Wikidump:

Size = 13GB; wikidump used for training of word vector. [6]

2) BBC dataset:

Contain sentences equal to 37000 appproximate. BBC dataset used for training of word vector and DT-RAE. [12]

3) MSRP dataset:

MSRP is dataset consist of pair of paraphrase sentence. [5], [24]

B. Features set description

1) Stopword:

The Stopword [28] are frequently occurring words in text documents and appear to be of little semantic value, hence can be excluded example is, a, was, etc. These words are called Stopword. For experiment we use Stopword list provided by Stanford Natural Language processing group.

2) Number feature:

The number feature [25] adds three values to the fixed input produced by the proposed work using the condition given as:

a) First is 1 if the pair of sentences contains same numbers or no number else 0.

b) Second is 1 if both sentences contain the same numbers else 0

c) Third is 1 if the set of numbers in a sentence is a strict subset of the numbers in the other sentence, e.g. {12.3, 12, 20} and {12.3}.

If the feature value is 1 means that it is includes else if feature value is 0 means that it is not included.

C. Dependency-tree Recursive Autoencoder parameter description

1) DT-RAE is a recursive autoencoder trained using ADAM [15] and reconstruction error, i.e. Euclidean distance between the word vector and its reconstructed vector.

2) Use the fixed values of learning rate and regularization controller on L2 regularization in the weight update term.

3) The word vector size of 200 is used for training DT-RAE.

4) The dynamic pooling (for fixing matrix generation) pool size is 15 and min-pooling function is taken.

D. Dependency-tree Recursive Autoencoder model description in Table I

1) DT-RAE_h1_0:

DT-RAE autoencoder with one hidden and trained without Stopword presence.

2) DT-RAE_h1_1:

DT-RAE with one hidden and trained with Stopword presence.

3) DT-RAE_h2_0:

DT-RAE with two hidden and trained without Stopword presence.

4) DT-RAE_h2_1:

DT-RAE with two hidden and trained with Stopword presence.

TABLE I
Parameter of DT-RAE models

Model name	Stop word	No. of hidden lay.	Size of hidden lay.		Learning rate		Regularization Controller	
DT-RAE_h1_0	0	1	200		0.001		0.0001	
DT-RAE_h1_1	1	1	200		0.001		0.0001	
DT-RAE_h2_0	0	2	Lay1	Lay2	Lay1	Lay2	Lay1	Lay2
			150	200	0.005	0.0001	0.9	0.99
DT-RAE_h2_1	1	2	Lay1	Lay2	Lay1	Lay2	Lay1	Lay2
			150	200	0.005	0.0001	0.9	0.99

layi = i^{th} layer of model,
DT-RAE_hi_j = DT-RAE means dependency tree recursive autoencoder, hi means model contain i layers, j weather stopword include (1) or not (0)

E. MSRP dataset description

1) MSRP training set:

The MSRP training set used to train the classifier of the proposed work for paraphrase detection. The TABLE II gives content description of the MSRP training dataset.

2) MSRP testing set:

The MSRP testing set use to test the classifier of the proposed work for paraphrase detection. The TABLE II gives content description of the MSRP testing dataset.

TABLE II: MSRP dataset

Data	Total sentence pair	Paraphrase pair	Non-paraphrase pair
Training set	4076	2753	1323
Testing set	1725	1147	578

V. Results and Discussion

This section presents the results obtained from training and testing of the proposed work and shows the detailed evaluation of the proposed work performance by selecting or deselecting feature and using different model given in TABLE I. Various performance parameters like accuracy, etc. are used for evaluation of the proposed work.

A. Classifier selection results

The Classifier is selected on the base of maximum score, which means the maximum value of accuracy and f1 score from the given classifier. For classifier selection a simple para-phrasing detection test is done with the MSRP test dataset as input, DT-RAE_h1_1 model with Stopword and number feature set to 1 is used. The classification results are taken 10 times and maximum values are shown in TABLE III.

TABLE III: RESULT OF VARIOUS CLASSIFIERS ON MODEL DT-RAE_H1_1 WITH STOPWORD AND NUMBER FEATURE SET TO 1

Classifier	Minimum accuracy	Respective f1 score	Maximum accuracy	Respective f1 score
NN+log loss+lbfgs+tanh	60.058	69.561	61.275	71.4285
NN+log loss+adam+tanh	65.681	74.171	69.101	79.954
NN+log loss+adam+sigmoid	68.869	80.118	69.681	80.197
SVM + log loss + rbf	66.493	79.875	66.493	79.875
SVM+log loss+sigmoid	66.029	79.395	66.029	79.395
SVM+log loss+linear	68.696	79.405	68.695	79.405

The results in Table 3 show that the SVM+log loss+linear classifier produce steady accuracy and f1 score. So, for classification SVM+log loss+linear is used.

TABLE IV: COMPARATIVE RESULTS OF PROPOSED MODELS WITH DIFFERENT FEATURE. (codes : https://github.com/deepakrana47/dt-rae)

model	Stop word	Number feature	recall	preci sion	Accur acy	f1 score
DT-RAE_h1_0 + Dynamic pool	0	0	95.73	67.99	67.19	79.50
DT-RAE_h1_0 + Dynamic pool	0	1	94.24	69.61	68.81	80.07
DT-RAE_h1_1 + Dynamic pool	1	0	95.11	68.62	67.82	79.72
DT-RAE_h1_1 + Dynamic pool	1	1	93.55	70.31	69.44	80.28
DT-RAE_h2_0 + Dynamic pool	0	0	95.55	68.15	67.36	79.56
DT-RAE_h2_0 + Dynamic pool	0	1	94.59	69.37	68.64	80.04
DT-RAE_h2_1 + Dynamic pool	1	0	95.99	69.07	68.75	80.33
DT-RAE_h2_1 + Dynamic pool	1	1	94.51	70.03	69.44	80.44

B. Paraphrasing detection on MSRP dataset

The classification of MSRP testing dataset is done by using NN with sigmoid activation function and ADAM as weight optimization function. Classification is done on the output produced by using different DT-RAE models given in TABLE I, dynamic pooling and selection or de-selection of Stopword and number features. The results shown in TABLE VI.

From the TABLE IV it can be viewed that the DT-RAE_h2_1 model with Stopword set and number feature set to 1 gives the highest accuracy and f1 score. In the Table 4 it can also be seen that the use of Number feature increases the accuracy by 1% approx. and the Stopword feature provides a very little increment in the accuracy.

C. Comparison with other approach

In this subsection the results of proposed method is compared with the baseline and base method.

It can be seen from TABLE V that the proposed work is able to perform better than Baseline with accuracy value exceed by 4.04% and f1 score exceed by 5.14% but not able to perform better than base work [25], while the accuracy value is lagging behind by 7% approx. and the f1 score lags behind by 3.4% approx.

TABLE 5: COMPARISON WITH OTHER MODELS FOR PARAPHRASING DETECTION ON MSRP DATASET

Model	Reference	Accuracy	f1 score
Vector Based Similarity (Baseline)	[18]	65.4	75.3
DT-RAE + dynamic pooling	Proposed model	69.44	80.44
RAE + dynamic pooling(base)	[25]	76.8	83.6
human	[5] [24]	83	-

VI. CONCLUSIONS

From the result, it is concluded that the proposed RAE-based model developed to get vector representations of phrases for natural language (English), one should not focus on fast reduction of error while training. During training if the model parameters (i.e. Weights) cover a wide area of parameter dimensional space, it gets a more generalized view of phrases in natural language (English). In future the DT-RAE model can be tested with multiple sense word vector representation and L-BFGS for weight optimization.

VII. ACKNOWLEDGMENT

.We thank to Computer Engineering department of G.B.P.U.A.T. for providing infrastructure and support for this research work.

REFERENCES

[1] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. Journal of machine learning research, 1137-55.

[2] Blacoe, W., & Lapata, M. (2012). A comparison of vector-based representations for semantic composition. In Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, (pp. 546-556).

[3] Cheng, J., & Kartsaklis, D. (2015). Syntax-aware multi-sense word embeddings for deep compositional models of meaning. arXiv, preprint arXiv:1508.02354.

[4] Collobert, R., and Weston, J. (2008) A united architecture for natural language processing: Deep neural networks with multitask learning. In ICML, 2008.

[5] Dolan, B., Quirk, C., & Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In Proceedings of the 20th international conference on Computational Linguistics, (p. 350).

[6] English Wikipedia, 2017. Retrieved April 12, 2017, http://wikipedia.org/wiki/English_Wikipedia.

[7] Fernando, S., & Stevenson, M. (2008). A semantic similarity approach to paraphrase detection. In Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics, (pp. 45-52).
[8] Finch, A., Hwang, Y., S., & Sumita, E. (2005). Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In Proceedings of the Third International Workshop on Paraphrasing (IWP2005), pp. 17-24.
[9] Islam, A., & Inkpen, D. (2009). Semantic similarity of short texts. Recent Advances in Natural Language Processing V, 309:227-36.
[10] Ji, Y., & Eisenstein, J. (2013). Discriminative Improvements to Distributional Sentence Similarity. In EMNLP 2013, (pp. 891-896).
[11] Goller, C., & Kuchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. In Neural Networks, 1996, IEEE International Conference, (Vol. 1, pp. 347-352), IEEE.
[12] Greene, D. and Cunningham, P., 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In Proceedings of the 23rd international conference on Machine learning, pp. 377-384.
[13] He, H., Gimpel. K., & Lin, J. J. (2015). Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In EMNLP, pp. 1576-1586.
[14] Hu, B., Lu, Z., Li, H., & Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In Advances in neural information processing systems, (pp. 2042-2050).
[15] Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[16] Kozareva, Z., & Montoyo, A. (2006). Paraphrase identification on the basis of supervised machine learning techniques. In FinTAL, (pp. 524-533).
[17] Madnani, N., Tetreault, J., & Chodorow, M. (2012). Re-examining machine translation metrics for paraphrase identification. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (pp. 182-190).
[18] Mihalcea, R., Corley, C., & Strapparava, C. (2006). Corpus-based and knowledge-based measures of text semantic similarity. In AAAI, (Vol. 6, pp. 775-780).
[19] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv, preprint arXiv:1301.3781.
[20] Milajevs, D., Kartsaklis, D., Sadrzadeh, M., & Purver, M. (2014). Evaluating neural word representations in tensor-based compositional settings. arXiv, preprint arXiv:1408.6179.
[21] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In EMNLP, (Vol. 14, pp. 1532-1543).
[22] Pollack, J. B. (1990). Recursive distributed representations. Artificial Intelligence. 46(1):77-105.
[23] Qiu, L. and Kan, M.Y. and Chua, T.S. 2006. Paraphrase recognition via dissimilarity significance classification. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pp. 18-26.
[24] Quirk, C., Brockett, C., & Dolan, W. B. (2004). Monolingual Machine Translation for Paraphrase Generation. In EMNLP, (pp. 142-149).
[25] Socher, R., Huang, E., H., Pennin, J., Manning, C., D., & Ng, A. Y. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In Advances in Neural Information Processing Systems, (pp. 801-809).
[26] Socher, R., Lin, C., C., Manning, C, & Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In Proceedings of the 28th international conference on machine learning (ICML-11), pp. 129-136.
[27] Socher, R., Karpathy, A., Le, Q., V., Manning, C., D., & Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. Transactions of the Association for Computational Linguistics. 2:207-18.
[28] Stopword, 2017. Retrieved on February 1, 2017, https://github.com/stanfordnlp/CoreNLP/blob/master/data/edu/stanford/nlp/patterns/surface/stopwords.txt
[29] Wan, S., Dras, M., Dale, R., & Paris, C. (2006). Using dependency-based features to take the para-farce out of paraphrase. In Proceedings of the Australasian Language Technology Workshop, (Vol. 2006).
[30] Wang, Z., Mi, H., & Ittycheriah, A. (2016). Sentence similarity learning by lexical decomposition and composition. arXiv preprint arXiv:1602.07019.
[31] Yin, W., & Schütze, H. (2015). Convolutional neural network for paraphrase identification. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (pp. 901-911).
[32] Das, D. and Smith, N.A., 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, pp. 468-476.

Deepak Singh Rana received his B. Tech. from Department of Computer Science and Engineering of G. B. Pant Collage of Engineering in 2009 and his M. Tech. from department of Computer Engineering of G. B. Pant University of Agriculture and Technology in 2017.

Pankaj Kumar Mishra Assistant professor in Department of Computer Engineering of G. B. Pant University of Agriculture and technology.