Novel Shared Input Based LSTM for Semantic Similarity Prediction

D. Meenakshi and A. R. Mohamed Shanavas
Department of Computer Science, Jamal Mohamed College (Autonomous), Affiliated to Bharathidasan University, Trichy, Tamil Nadu, India
Email: {rschlrmeena, drarmshanavas}@yahoo.com

***Abstract*—Automated similarity detection in text is a core component for several applications. This work is an independent component of the formative assessment architecture being developed by the authors. The proposed model is constructed using a Shared Input based Long Short Term Memory (SI-LSTM) model. The model is composed of a preprocessing phase, an embedding layer creation phase and the SI-LSTM model. The SI-LSTM model is composed of two sections. The first section has been designated for building features. The input for the model is composed of two distinct inputs, whose level of similarity is to be identified. Feature matrices are created for the inputs using the embedding layer and the LSTM layer. The resultant features are integrated and passed through a deep learning model for duplicate identification. Experiments were performed using the Quora Question Pairs dataset. Comparisons with the existing state-of-the-art model indicate an improvement in accuracy of 1.7%, improvements in recall of 11.9% and improvements in F-Score of 5.7%.**

***Index Terms*—duplicate identification, semantic similarity identification, deep learning, LSTM, text processing, online education, automated test**

I. Introduction

Text based searches are increasing due to the availability of a huge amount of information available online. This has been accentuated by search engines such as Google that perform text matching to recommend web pages to users [1]. Further, websites based on Community Question Answering (CQA) also rely on text based searches. Several websites. like Quora or Stack Overflow, are solely based on providing solutions to users' queries [2], [3]. Such sites require identifying text similarities to identify similar questions and also to eliminate duplication of questions. These require understanding the semantic relatedness of content, rather than analysis of the content alone. These are the mainstream uses of content duplicate detection. However, several other applications exist for duplicate detection, avoiding duplicates in questions and integrating questions that are similar in nature, identifying similar data from corpus, and identifying web pages that are similar in nature to the user's query. With the recent advent of online-based teaching and assessment, another major application of duplicate identification has been identified in the education domain [4].

Early learning in education domain models was based on behavioral, cognitive and constructivist techniques. With the advent of the new connectivism model, learning and performance assessments have moved to the next higher level [5], [6]. These models require regular assessment of the performance of students. Tests play a vital role in the analysis of the performance of students. Summative and formative assessments are currently used for testing and assessment in the education domain. Summative assessments are formal and structured forms of assessments. They concentrate on standardizing performance for effective comparison [7]. Summative performance analysis has been the standard until quite lately, when it was identified that this type of assessment is not effective. Adaptive tests are the current norm, and are now in the main stream with the aid of automation techniques.

Formative assessment is an ongoing technique that is flexible and more informal than other diagnostic tools to be used in the education domain. Iterative feedback-based tests are required for formative assessments [8]. The feedbacks are to be obtained and processed during the course of the tests. Hence, the entire process is required to be automated. Question selection process should be automated to enable adaptive testing. The automated question generation model is required to provide similar questions with slightly higher or reduced complexity depending on the answer provided by the student [9]. This requires the application to analyze the questions and categorize them based on their similarity and complexity levels. Complexity levels are provided by the users, while similarity levels are required to be automatically identified. Similarity or duplicate level identification forms the core part of this process [10].

This work is a component of the formative assessment architecture that is being developed by the authors for automated adaptive testing. A part of this architecture requires analyzing the input questions to identify the similarity levels between them. Topic level associations are identified based on the semantic similarity levels of the questions. This work forms the similarity identification component, that enables question grouping. Similarity levels are also used to select questions based

Manuscript received December 13, 2021; revised May 9, 2022.

doi: 10.12720/jait.13.4.387-392

on the answers provided by a student to the previous question presented to them.

II. Related Work

Identifying the semantic relatedness of text has been an interesting and highly researched domain. Initial techniques utilized word corpus and used statistical measures with logical inferences to solve this problem. With the advancement in machine learning models, identifying semantic similarities to a better extent has become possible. A word embedding and cosine similarity based model for duplicate detection has been proposed by Babu *et al.* [11]. The model is based on questions from Stack Overflow. The model creates numerical vectors based on the cosine similarity values of the vectors obtained from the word embedding matrix. Statistical measures are applied in the numerical results to obtain the similarity levels. Other similar methods that use transformation based techniques for duplicate detection include works by Zhang *et al.* [12] and Bogdanova *et al.* [13].

An ANN based model that uses Manhattan Distance and LSTM to perform duplicate pair detection was proposed by Imtiaz *et al.* [14]. This technique has been designed for duplicate detection in the Quora question pair data. The model analyzes three different embedding techniques and applies them to the MaLSTM model to derive predictions. A multi-perspective model that has been designed to identify sentence similarity using Natural Language Sentence Matching (NLSM) was proposed by Wang *et al.* [15]. This technique also incorporates feature engineering techniques to identify sentence features. Deep Learning and LSTM have been the go-to techniques to solve text based problems. Some effective deep learning models used for this task include Siamese Recurrent Neural Network (RNN) models by Mueller *et al.* [16] and non-linear similarity identification models by Tsubaki *et al.* [17]. LSTM networks have been proven to exhibit exemplary performance in text processing problems. Techniques using LSTM as their major components in duplicate detection include the semantic similarity identification model by Sravanthi *et al.* [18], the supervised and semi-supervised LSTM models by Johnson *et al.* [19] and document modelling technique by Tang *et al.* [20].

Intent based analysis is the core component of duplicate detection. Identifying the text with the same intent signifies high levels of match between texts. A duplicate detection model that is based on intent analysis was proposed by Shankar *et al.* [21]. The model extracts various features such as; string based measures, stylistic and structural measures, semantic similarity measures, and fuzzy string matching measures. A Bayesian based approach to identify semantic similarity levels was proposed by Chergui *et al.* [22]. This work uses Case Based Reasoning (CBR) to identify similarities. Duplicate detection has been handled by either creating enhanced features from the text data, or by using Deep Learning models or LSTM. However, an aggregation of both the techniques is not handled by any of the existing models.

III. Dataset Description

For duplicate detection, the Question Pairs dataset from Quora [23] is used. The major intent behind the publication of this data is to avoid duplicated questions. The scope of the data is not only intended for content duplication, but also based on semantic equivalence. The training data has been designed with 400,000 distinct question pairs, each with a binary value indicating if the questions are distinct or duplicates. Descriptions of the data are provided in Table I.

TABLE I. Dataset Description

Attribute	Datatype	Description
ID	Integer	Question pair ID
QID 1	Integer	ID of question 1
QID 2	Integer	ID of question 2
Question 1	String	Text representing question 1
Question 2	String	Text representing question 2
Is_duplicate	Integer (1/ 0)	1 if the two questions are duplicates, 0 otherwise

Question 1 and Question 2 are string representations and are textual in nature. These serve as the base data for analysis. QID 1 and QID 2 are the ID values. Is_duplicate is a binary value that represents the semantic similarity status of the two questions.

IV. Duplicate Question Prediction Using Shared Input LSTM (SI-LSTM)

The proposed architecture is composed of the input preprocessing and tokenization phases, embedding matrix creation phase and the proposed Shared Input LSTM (SI-LSTM) for duplicate prediction. The initial phases perform data preparation that aids in the conversion of textual data into numerical vectors. Feature generation and duplicate identification are performed in the SI-LSTM phase.

A. Data Preprocessing

Input data for duplicate prediction is textual in nature. Text needs to be analyzed both in terms of content and context to identify duplication. Duplicated content alone does not correspond to contextual similarity. For e.g. “Do you have a dog?” and “You do have a dog” contain same tokens, hence are considered to be the same in terms of content, however, contextually they convey different meanings. One is a question, while the other represents a fact. Effective preprocessing of text plays a vital role in differentiating such entities.

Text preprocessing deals with tokenization, stemming and normalization. Tokenization is the process of dividing the text into smaller components; words. Token generation is based on delimiters such as space and all the special characters. The delimiters can be designated based on the domain. The tokenized components are passed to the stemming algorithm. This work uses the Porter Stemmer algorithm for this purpose. Stemming is

the process of removing the affixes (prefix and suffix) from tokens to obtain the root word. The next process is token normalization. Token standardisation is achieved using text normalization. This process eliminates the inflections contained in the text. Completion of these stages completes the text preprocessing phase. The resultant list of tokens is passed to the next phase for duplicate detection.

B. Embedding Layer Creation

Tokens generated from the preprocessing phase are textual in nature. Machine learning requires numerical data. This phase converts textual tokens into numerical vectors. This is done by creating the embedding matrix. Although text datasets differ based on their domain, the basic components, i.e. the tokens, are mostly similar in nature. Hence, standard word representations can be used for word vector creation. This work uses GloVe (Global Vectors) for word representation. GloVe is an unsupervised learning algorithm that contains vector representations for words based on co-occurrence statistics. This work uses the GloVe 200d dataset which contains 400,000 word vectors. The embedding matrix can be created prior to the actual embedding process. This acts as the training phase for the embedding layer. The embedding matrix that was built in the embedding layer is used to construct word vectors for the input data.

C. Shared Input LSTM (SI-LSTM) Model for Duplicate Prediction

Duplicate detection is the process of identifying whether two given texts are semantically equivalent. Hence, unlike general machine learning models with a single input, this work contains two different inputs that are to be processed independently. The resultant vectors from the model are concatenated and analyzed to verify their similarity levels. Data flow and the network architecture of the proposed SI-LSTM Model are shown in Fig. 1.

Input for the duplicate detection architecture is composed of two text entities. The entities are preprocessed and the word vectors are passed as input to the model. Each of the inputs is passed to the embedding layer to determine the embedding matrix for the input data. This results in the creation of numerical vectors for the word vectors created by the preprocessing component. Each input word vector is converted to a distinct numerical vector representing the textual component.

The next phase is the feature extraction phase using the LSTM module. LSTM, also known as Long-Short Term Memory is the type of network that facilitates data persistence. Deep learning ANN models operate of data to generate features. Features generated in a layer are passed to the next layer. Prior values are not retained by any of the layers. The domain of text processing requires persistence of prior data to enable semantic correlation between components. This is provided by the LSTM model.

LSTM nodes are composed of three major components; the input gate, output gate and the forget gate. The node maintains dependencies between tokens with the help of these three components. Activation vectors for the input, output and forget gates are provided below.

$$gate_t = \sigma_{gate}\left(W_{gate}x_t + U_{gate}h_{t-1} + b_{gate}\right)$$

All three gates operate based on the same equation. Any gate at time t is determined by its activation function, which is the sigmoid function, weight matrices of the gate W and U, bias vector b, input vector x, and the number of hidden units h.

The LSTM node is followed by a sequence of dense layers with progressively reducing the number of nodes in each layer. This concentrates the features effectively to enable better prediction in the further layers. The dense layers are followed by a dropout layer, with a dropout level of 20%. This marks the end of the feature extraction phase. Features are extracted independently for each of the inputs. Until this phase, the model operates as two independent branches. Each branch contains features extracted from a single text component.

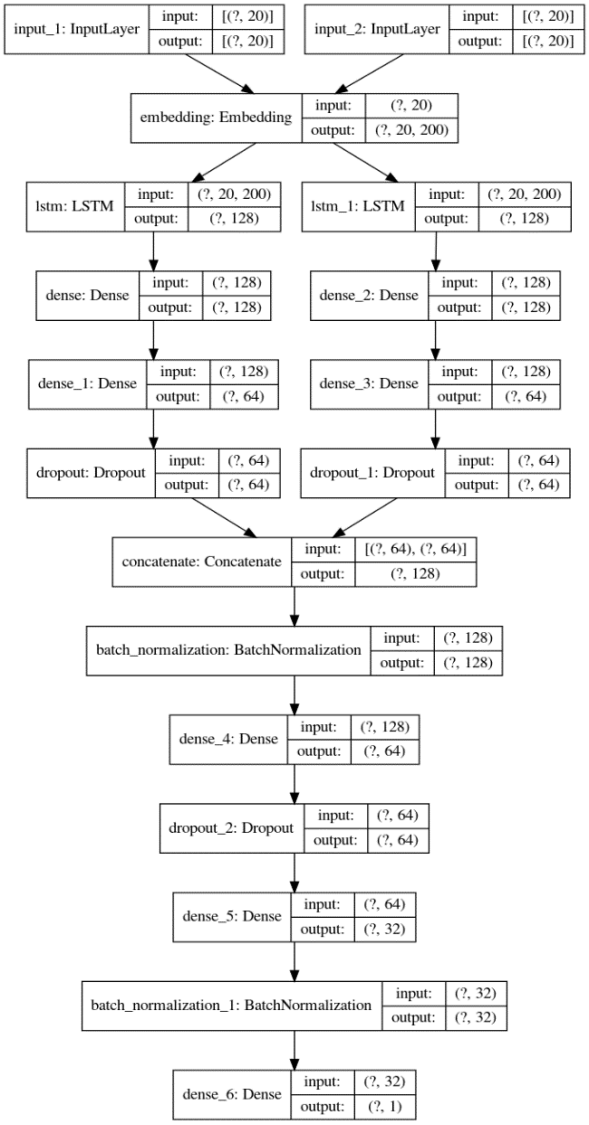

Figure 1. Data flow and SI-LSTM network architecture.

The next phase performs feature aggregation to identify if the two texts are contextually and semantically

equivalent or not. Feature aggregation is performed by concatenating both the feature vectors. The concatenated vectors are batch normalized and passed through a series of dense layers to obtain the final prediction. Batch normalization is used in deep networks to perform input standardization for layers. Batch normalization stabilizes the learning process and ensures effective handling of the internal covariate shift. It also aids in accelerated training and reduced epochs during the training process. The final dense layer uses sigmoid activation and has an output of one neuron that outputs a binary value indicating if the texts are duplicates or not.

V. Results and Discussion

Duplicate pair detection is performed using the Quora Question pair dataset. The proposed model has been implemented using Python and the SI-LSTM model is built using the Keras library. Hardware configurations include Tesla P100-PCIE-16GB GPU, 13GB RAM and Intel Xeon CPU. The hyper-parameters used for the network are provided in Table II.

TABLE II. Network Hyper Parameters

Parameter	Value
Loss	Binary Cross Entropy
Optimizer	Adam
Batch Size	1024
Epochs	100
Validation Data	Provided
Analyzing Metrics	Loss, Accuracy

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 20)]         0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 20)]         0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 20, 200)      4000000     input_1[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 128)          168448      embedding[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 128)          168448      embedding[1][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          16512       lstm[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 128)          16512       lstm_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           8256        dense[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 64)           8256        dense_2[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 64)           0           dense_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 64)           0           dense_3[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 128)          0           dropout[0][0]
                                                                 dropout_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 128)          512         concatenate[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 64)           8256        batch_normalization[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 64)           0           dense_4[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 32)           2080        dropout_2[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 32)           128         dense_5[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 1)            33          batch_normalization_1[0][0]
==================================================================================================
Total params: 4,397,441
Trainable params: 397,121
Non-trainable params: 4,000,320
__________________________________________________________________________________________________
```

Figure 2. Network description.

A detailed description of the layers, input dimensions, number of parameters obtained in each layer and the layers to which they are connected are presented in Fig. 2. It could be observed that the proposed architecture uses ~ 4 million non-trainable and 397K trainable parameters. Accuracy, Precision, Recall and F-Score are used as metrics to measure the performance of the SI-LSTM model.

$$Accuracy = \frac{TP + TN}{FP + TP + FN + TN}$$

$$Precision = \frac{TP}{FP + TP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

where *TP*, *FP*, *TN* and *FN* are the True Positive, False Positive, True Negative and False Negative respectively.

ROC and PR plots for the SI-LSTM model are presented in Figs. 3 and 4. ROC, also called Receiver Operating Characteristics. It is plotted with True Positive Rate (TPR) and False Positive Rate (FPR) on its axes (Fig. 3).

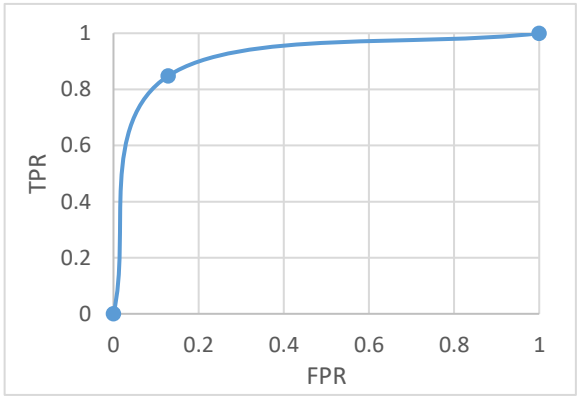

Figure 3. ROC plot for SI-LSTM.

The plot shows the level of true positives to false positives. High TPR levels and low FPR levels indicate high performance in models. The ROC plot begins from the point (0,0) and ends at (1,1). The intermediary points exhibit the performance exhibited by the models. Points towards the top left of the chart are indicative of high performance. It could be observed that the SI-LSTM model exhibits low FPR levels and high TPR levels, indicated by the point on the top left of the chart.

The PR plot presented in Fig. 4 represents the precision and recall levels of the SI-LSTM model. High precision and recall levels are expected of a high performing model. The graph shows high precision levels at ~80%, and high recall levels of >80%. This indicate that the model can effectively identify semantic similarity levels from text for effective duplicate detection.

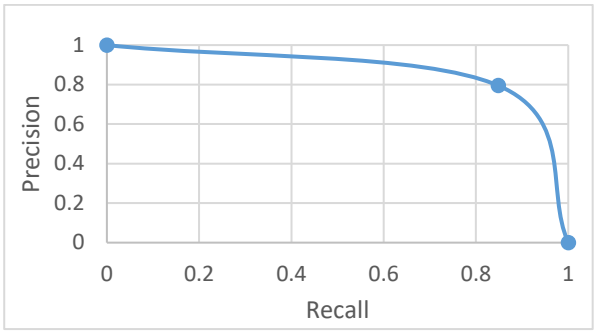

Figure 4. PR plot for SI-LSTM.

A comparison of performance of the proposed SI-LSTM model has been done with the MaLSTM model proposed by Imtiaz *et al.* [14] and is shown in Table III. The Comparison in terms of accuracy, precision, recall and F-Score indicates that the SI-LSTM model exhibits better performance in most of the metrics, except for Precision. Precision exhibits a 0.4% reduction when compared to the MaLSTM model. Accuracy exhibits an improvement of 1.7%, recall exhibits an improvement of 11.9% and F-Score exhibits an improvement of 5.7%. The exhibited performance indicates that the duplicate detection levels of SI-LSTM are much more effective compared to the existing models.

TABLE III. PERFORMANCE COMPARISON OF SI-LSTM AND MALSTM [14]

	SI-LSTM (Proposed)	**MaLSTM [14]**
Accuracy	86.2 %	84.5 %
Precision	79.5 %	79.9 %
Recall	84.8 %	72.9 %
F-Score	82 %	76.3 %

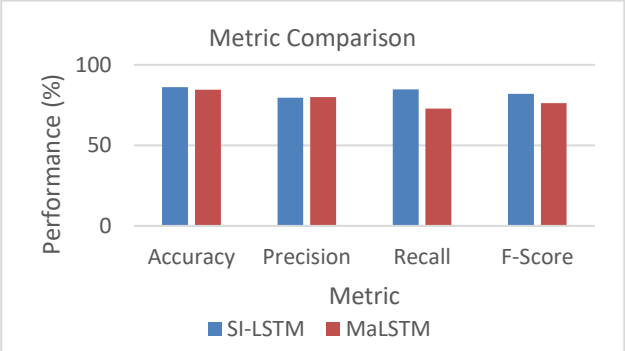

Figure 5. Metric comparison with MALSTM.

A comparison of standard performance metrics is shown in Fig. 5. It could be observed that precision level of both the models are almost similar, while the SI-LSTM model exhibits better performance in accuracy, recall and F-Score.

VI. CONCLUSION

Identifying semantic similarities in text has vast applications, especially in the current world where automation is at its peak. Education has been identified as one of the major domains that currently requires the aid of this process. With the advent of formative assessment based models in educational institutions, adaptive testing strategies are being researched widely. This work is a part of the automated formative assessment architecture that is being proposed by the authors. The deep learning based architecture has been designed to analyze two textual components to identify if they are semantically similar. The feature building phase of the proposed SI-LSTM model is constructed using the embedding layer and the LSTM layer. This process is performed individually for each of the inputs. The obtained features are integrated and passed to the deep learning model to identify the semantic similarity levels. The SI-LSTM model was analyzed using the Quora Question pair dataset. Results obtained were compared with the MaLSTM model and it was identified that the proposed model exhibits improvements in accuracy at 1.7%, improvements in recall of 11.9% and improvements in F-Score at 5.7%. Although the proposed SI- LSTM model exhibits better performance, the accuracy and precision levels still have scope for improvement. Future enhancements will be aimed towards improving the performance by using enhanced feature engineering techniques to improve the quality level of the training data.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

D. Meenakshi, the author made a big contribution to the article's concept, data collection, design and analysis of data. She also wrote the entire manuscript in accordance with the research supervisor Dr. Mohamed Shanavas's guidelines. The results were evaluated by all authors and the final version of the manuscript was approved by all of them.

REFERENCES

[1] L. Sharma, L. Graesser, N. Nangia, and U. Evci, "Natural language understanding with the quora question pairs dataset," arXiv preprint arXiv:1907.01041, 2019.

[2] E. Dadashov, S. Sakshuwong, and K. Yu. (2018). Quora question duplication. [Online]. Available: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2761178.pdf

[3] Z. Chen, H. Zhang, X. Zhang, and L. Zhao. (2018). Quora question pairs. [Online]. Available: http://static.hongbozhang.me/doc/STAT_441_Report.pdf

[4] T. B. Lalitha and P. S. Sreeja, "Personalised self-directed learning recommendation system," *Procedia Computer Science*, vol. 171, pp. 583-592, 2020.

[5] J. Xu, T. Xing, and M. V. D. Schaar, "Personalized course sequence recommendations," *IEEE Transactions on Signal Processing*, vol. 64, no. 20, pp. 5340-5352, 2016.

[6] S. Wan and Z. Niu, "An e-learning recommendation approach based on the self-organization of learning resource," *Knowledge-Based Systems*, vol. 160, pp. 71-87, 2018.

[7] Á. Tejeda-Lorente, J. Bernabé-Moreno, C. Porcel, P. Galindo-Moreno, and E. Herrera-Viedma, "A dynamic recommender system as reinforcement for personalized education by a fuzzly linguistic web system," *Procedia Computer Science*, vol. 55, pp. 1143-1150, 2015.

[8] A. Garrido, L. Morales, and I. Serina, "On the use of case-based planning for e-learning personalization," *Expert Systems with Applications*, vol. 60, pp. 1-15, 2016.

[9] D. Herath and L. Jayaratne, "A personalized web content recommendation system for E-learners in E-learning environment," in *Proc. National Information Technology Conference*, 2017, pp. 89-95.
[10] A. Klašnja-Milićević, B. Vesin, and M. Ivanović, "Social tagging strategy for enhancing e-learning experience," *Computers & Education*, vol. 118, pp. 166-181, 2018.
[11] J. Babu and S. Thara, "Finding the duplicate questions in stack overflow using word embeddings," *Procedia Computer Science*, vol. 171, pp. 2729-2733, 2020.
[12] Y. Zhang, D. Lo, X. Xia, and J. Sun, "Multi-factor duplicate question detection in stack overflow," *Journal of Computer Science and Technology*, vol. 30, no. 5, pp. 981-997, 2015.
[13] D. Bogdanova, C. D. Santos, L. Barbosa, and B. Zadrozny, "Detecting semantically equivalent questions in online user forums," in *Proc. the Nineteenth Conference on Computational Natural Language Learning*, 2015, pp. 123-131.
[14] Z. Imtiaz, M. Umer, M. Ahmad, S. Ullah, G. S. Choi, and A. Mehmood, "Duplicate questions pair detection using Siamese MaLSTM," *IEEE Access*, vol. 8, pp. 21932-21942, 2020.
[15] Z. Wang, W. Hamza, and R. Florian, "Bilateral multi-perspective matching for natural language sentences," arXiv preprint arXiv:1702.03814, 2017.
[16] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proc. Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
[17] M. Tsubaki, K. Duh, M. Shimbo, and Y. Matsumoto, "Non-linear similarity learning for compositionality," in *Proc. AAAI*, 2016, pp. 2828-2834.
[18] P. Sravanthi and B. Srinivasu, "Semantic similarity between sentences," *International Research Journal of Engineering and Technology*, vol. 4, no. 1, pp. 156-161, 2017.
[19] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embeddings," arXiv preprint arXiv:1602.02373, 2016.
[20] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proc. Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1422-1432.
[21] S. Shankar and A. Shenoy, "Identifying quora question pairs having the same intent," 2017.
[22] O. Chergui, A. Begdouri, and D. Groux-Leclet, "Integrating a Bayesian semantic similarity approach into CBR for knowledge reuse in community question answering," *Knowledge-Based Systems*, vol. 185, 2019.
[23] Question Pairs Dataset. [Online]. Available: https://www.kaggle.com/quora/question-pairs-dataset

D. Meenakshi is currently a Research Scholar, in the Department of Computer Science, Jamal Mohammed College (Autonomous), Trichy, India. She received her M.Phil Degree in Bharathidasan University, Trichy, India in 2014 and also she is pursuing Ph.D (Computer Science) in Bharathidasan University, Trichy. Her area of interest includes Big Data, Cloud Computing and so on.

A. R. Mohamed Shanavas is working as an Associate Professor, in the Department of Computer Science, Jamal Mohamed College (Autonomous), Trichy, India. He completed his PhD (Computer Science)Bharathidasan University, Trichy. His area of interest include IoT, AI, cloud computing, Data Analytics, Image processing and Data Mining.