Deep Learning-Based Correct Answer Prediction for Developer Forums

HAFIZ UMAR IFTIKHAR1, AQEEL UR REHMAN2, OLGA A. KALUGINA3, QASIM UMER2, AND HARIS ALI KHAN2
1Research Center for Modelling and Simulation, National University of Sciences and Technology, Islamabad 44000, Pakistan
2Computer Science Department, COMSATS University Islamabad, Vehari Campus, Vehari 61100, Pakistan
3Department of English Language for Professional Communication, Financial University Under the Government of the Russian Federation, 125167 Moscow, Russia
Corresponding author: Qasim Umer (qasimumer@cuivehari.edu.pk)

ABSTRACT
Developer forums are essential for software engineers to solve their problems with the assistance of experts on such forums. However, sometimes the solutions (answers) of a problem are not satisfactory or challenging to select the potential answer. Information seekers usually browse all the answers within the question thread to get the potential answer. The manual selection of correct answers is a tedious and time-consuming task. In this paper, we propose an automatic classification approach to predict the correct answers for developer forums. We first extract the metadata and combination of Q/A for each thread of the developer community (Stack Overflow). Then, the natural language processing techniques are applied to preprocess the Q/A combinations of the given dataset. After that, a keyword ranking algorithm is leveraged to extract keywords and their ranking scores for each Q/A combination. Based on keywords and their ranking scores for each Q/A combination, a keywords-based feature vector is constructed. Subsequently, word embedding is leveraged to convert each preprocessed Q/A combination into a text-based feature vector. Finally, we pass the metadata, keywords-based features, and text-based features to the ensemble deep learning model for training to predict correct answers. The results of 10-fold cross-validation specify that the proposed approach is accurate and surpasses the state-of-the-art. On average, it improves the accuracy, precision, recall, and f-measure up to 1.72%, 24.96%, 6.57%, and 16.62%, respectively.

INDEX TERMS
Developer forums, community questing answering, correct answer prediction, stack overflow, software maintenance, deep learning, reliability.

I. INTRODUCTION
The popularity of mailing list forums and web-based discussion is decreased with the extensive use of modern community-based question answering (Q/A) sites, e.g., Stack Overflow. In this modern era, the complexity of software is rising due to the advancement and rapid growth in the software industry. Therefore, Q/A forums became an essential need of developers [1]. The increasing trend of Q/A forums has also been noticed as developers require more technical support, information, and expert opinions. Previously, the mailing list and web-based discussions were helpful. Both allow developers to store and search the generated knowledge, but web-based discussions prevent users from asking similar questions. However, identifying an accurate resolution of requested assistance is a manual, time-consuming, and challenging task. Highlighting such identification in the first position is the main factor in the success of Q/A forums, e.g., Quora and Yahoo Answer [2], [3].

Lately, developer forums are moving their support from traditional forums to Stack Overflow [1]. The largest Q/A site of the Stack Exchange network (forum) allows programmers to ask software development-related questions and helps them by providing answers daily. Stack Overflow further facilitates and encourages users by allowing them to assign custom tags to threads that highlight them from specific developer forums. Moreover, the inability of the archival support of content is another reason for disassembling the traditional forums, i.e., 6 out of 21 developer forums [1] are entirely dismantled due to the inability of the archival support of their content. Consequently, the content storage of developer forums is essential for their success. However,
the storage of unwanted content, e.g., incorrect answers, is an overhead for developer forums due to their popularity and information overload. Therefore, the selection of correct answers is a valuable step for the communities. Recently, developer forums, e.g., Stack Overflow, are performing this step manually, which is a tedious and time-consuming task. Therefore, automatic identification of correct answers could be beneficial for developer forums.

In this paper, we propose an automatic classification approach to predict the correct answer for developer forums. We first extract the metadata, e.g., votes (m-features) and combination of Q/A for each thread of Stack Overflow [4]. Then, the natural language processing techniques are applied to preprocess the Q/A combinations of the given dataset. After that, a keyword ranking algorithm (TextRank) [5] is leveraged for the extraction of keywords and their ranking scores for each Q/A combination, and a keywords-based feature vector (k-features) is constructed based on the keywords and their ranking scores for each Q/A combination. Subsequently, word embedding is leveraged to convert each preprocessed Q/A combination into a text-based feature vector (t-features). Finally, we pass the m-features, k-features, and t-features to the ensemble deep learning model for training to predict correct answers. The 10-fold cross-validation technique is leveraged to evaluate the proposed approach. The evaluation results specify that the proposed approach is accurate and surpasses the state-of-the-art.

The contributions of the paper are as follows:

- An automatic classification approach is proposed for correct answers prediction for developer forums. To the best of our knowledge, we are the first to exploit keyword ranking algorithms with an ensemble deep learning model in the prediction of correct answers for developer forums.
- The evaluation results of 10-fold cross-validation specify that the proposed approach is accurate and surpasses the state-of-the-art. The average accuracy, precision, recall, and f-measure are 1.72%, 24.96%, 6.57%, and 16.62%, respectively.

The rest of the paper is organized as follows: Section II presents the basic knowledge. Section III presents the related work. Section IV introduces the proposed approach. Section V highlights the research questions, evaluation criteria, and findings. Section VI and Section VII inform the threats to validity and conclude the paper, respectively.

II. BASIC KNOWLEDGE

This section provides the essential background of developer communities and how they work.

A. DEVELOPER COMMUNITIES

Developer communities are generally referred to as places where developers share their programming-related problems and seek the correct answer from developers worldwide who post their answers according to their expertise. These communities are the center point of interaction and knowledge sharing and have become an on-demand knowledge base for fostering knowledge that improves the workflow and decreases support costs. There are numbers of developer communities available, e.g., Toptal, Developers Forum, Mozilla Developer Network, Experts-exchange, and Stack Overflow. However, Stack Overflow is the largest and absolute must community for the developers who seek answers to their programming related questions. Since its inception from 2008 Stack Overflow is growing rapidly and as of in 2019 it has more than 4.7 million developers who assured top-notch advice anytime.

B. STACK OVERFLOW ANSWER SELECTION PROCESS

For the last decade, Stack Overflow is the most popular platform for finding programming related problems. It is written by many and read by many. It is often among the first search results for programming related queries. To ensure the visibility of good content, Stack Overflow provides a voting system where users can contribute high quality answers by assigning positive votes and thematic tags [6], e.g., Python and Machine Learning. Thereby a reputation score is formed that identifies the most knowledgeable users for specific fields and provides privileged roles to high-ranked users, i.e., up-voting, editing, and moderating the community [6]. Moderation of the community is quite strict; only well-documented, on-topic, and correctly tagged questions/answers are accepted. These features ensure the quality of the content and provide a rich resource for the social analysis of the platform.

On Stack Overflow, only a super user (who posted the question/query) can accept the answer. Consider a case where a super user and other users have accepted the answer and up-voted the answer, respectively. Later, another answer is posted by the super user which is more reliable and efficient as compared to the accepted/up-voted answer. In that case, the more accurate answer should be replaced by the super user. However, the most accurate answer is often ignored in such cases. Therefore, Stack Overflow requires an automatic solution for finding the accurate answer to avoid the manual process of producing correct answers by exploiting the voting, badges, and reputation.

III. RELATED WORK

The questioning and answering system of Stack Overflow worked on voting and badges system, Roy et al. [7] claimed that the voting system is not a good way to check the quality of an answer; sometimes, a better answer posted later did not get high votes and placed at the bottom. They used the stack exchange open-source dataset for classification and labeled the data into three classes based on voting. Answers that received votes below 1 labeled as low-quality answers, those who received 1 or 2 votes are labeled as a moderate-quality answer, and the answers obtained votes greater than 2 labeled as high-quality answers. They extracted 26 features including wrong words, code snippet, user reputation, readability, activity/topic entropy, topical reputation, question-answer
similarity, and answer-answer similarity. They trained three classifiers: naïve Bayes, random forest, and gradient boosting for classification, and their results showed that gradient boosting gives the best performance.

Users’ reputation is also a very important characteristic that increases the chances of an answer to be accepted on Stack Overflow [8], [9]. Bosu et al. [10] analyzed the ways of how a user can quickly earn a good reputation on Stack Overflow. The findings include answering questions with lower expertise, density tags, answering questions promptly, becoming the first answerer, becoming the active member in peak hours, and contributing to diverse areas. Zheng and Li [11] took three features under consideration (i.e., textual, code snippet, and answerer background) and performed AdaBoost-based learners for the prediction of correct answers on the Stack Overflow dataset. Their classifier model achieved 63%, 59%, and 61% of precision, recall, and f-measures, respectively.

To assess the quality of questions, Ponzanelli et al. [12] proposed an automated approach that detects the low-quality posts from Stack Overflow to refine the review queue. They used different datasets consisting of almost 5 million questions and separated them into two classes based on scores. The high-quality content class contains questions greater than zero, whereas the other questions belong to the low-quality content class. Their proposed model is mainly based on two features, i.e., textual features consisting of contents of the post and community-based features related to the user’s popularity.

DelPredictor model is presented by Xia et al. [13] that predicts the deleted questions based on two sets of features, i.e., meta-features (community, profile, and syntactic features) and textual features (question title, body, and tags). They implemented two classifiers: multinomial naïve Bayes and random forest and evaluated the model on 417685 deleted-questions of Stack Overflow. The results showed that multinomial naïve Bayes achieves good accuracy than baseline approaches. Similarly, Ponzanelli and Xia worked on the prediction of quality questions.

There are thousands of questions on Stack Overflow which are unanswered by users. To evaluate the reason, Treude et al. [14] performed analysis on 15 days Stack Overflow’s data and categorized the types of questions. They exploited 200 most frequently used tagged-keywords from the data and found that these keywords cover 60-193 tags. They identified that such tags cover most of the instances.

Calefato et al. [15] investigated how users can increase the chances of their answers to be accepted on Stack Overflow. For this purpose, they found four factors that highly influence the success of the answers: 1) presentation, which includes URLs, code snippet, length, uppercase ratio; 2) emotional effect either positive or negative; 3) time of answer posted; and 4) reputation score of askers. They built dataset from official Stack Overflow dump posted in 30 days having 348,618 total answers. They implemented logistic regression and achieved 64% of accuracy.

To check the effect of sentiments when writing questions, Calefato et al. [16] provided a guideline to write good questions. They shared the factors such as affect (either positive or negative), presentation-quality (code snippet, title, body length, uppercase character ratio, presence of multiple tags, and presence of URLs), time (posting time), and user reputation (asker reputation) that potentially influence the success of the questions. They used 82k questions from the official dump of Stack Overflow. They implemented logistic regression and achieved accuracy up to 65%.

Hart and Sarma [17] discussed the role of social cues for novice users to select answers on Stack Overflow and reputation earned by users. They performed a survey through Amazon Mechanical Turk on the sample set of java related questions and answers. The purpose of this survey was to check how social factors influence technical forums. The results suggested that the presentation style affects the quality of the answer instead of a social reputation as novice users usually judge the answer’s quality. Barua et al. [18] worked on Stack Overflow dataset, analyzed its textual content to discover the main topics that are in discussion of developers, and discovered the relationship between these topics and their trends over time. They used Latent Dirichlet Allocation for topic modeling and found that web development, mobile application, Git, and MySQL are the most popular topics discussed by developers.

Adamic et al. [19] presented the first study related to correct answer prediction using data from Yahoo Answers. They analyzed the categories of forums and clustered them according to the pattern of interaction and content categories among users. They characterized the entropy of users’ interests and mapped the related categories by analyzing the users’ participation across them. Moreover, they combined both features such as answer characteristics and user attributes to predict the correct answer. They utilized a logistic regression classifier and achieved 73% of accuracy for programming related answers. Shah [20] proposed a Bayesian network model on Yahoo Answers for correct answer prediction by using user-related features and textual features that are extracted from answers. Their model achieved 89%, 97%, 86%, and 98% of accuracy, precision, recall, and accuracy, respectively. As compared to other studies, the model proposed by Shah is less challenging.

Shah and Pomerantz [21] also considered the dataset of Yahoo Answers that contained non-technical questions to predict the quality of the answer. Their approach is based on 13 different criteria where the feature set contains user-related features. They selected a small set of questions, where each set contains at least 5 answers, and each answer is manually rated by 5 different people based on 13 predefined criteria. They compared the rated answers with the asker’s rating. They utilized logistic regression, and their evaluation results showed 85% of accuracy.

Tian et al. [22] presented a model that does not rely on user-related features as compared to others. First, they measured three key factors: 1) quality of answer content;
2) contribution of the answer to solve new question; and 
3) how it compares with other answers by designing the features, especially contextual information. Second, they predicted the correct answer by evaluating and designing a learning approach from extracted features. They applied two-fold cross-validation with the random forest algorithm on the Stack Overflow dataset, and their classifier achieved 72% of accuracy. Cai and Chakravarthy [23] also presented a temporal features model and argued that these features work better than traditional approaches. They used three different datasets to predict correct answers and applied 10-fold cross-validation with a support vector machine. Their classifier achieved only 55% of precision.

Burel et al. [24] proposes a model to predict the correct answer on two different types of communities, i.e., Stack Exchange and SAP Network. They experimented on three different datasets; two are extracted from the sub-communities of Stack exchange (Server Fault, cooking community), and the third dataset is taken from SAP Community (SCN forums). They used three kinds of features to train their classifier, i.e., user, thread, and content features. After the comparison of different decision tree algorithms (random forest, J48, ADT), they found the ADT classifier performed best and achieved 85% f-measure and 92% of accuracy.

Gkotsis et al. [25] also proposes a novel system named ACQUA that can be used to predict the correct answer. Answer score and user ratings are not used in this model as compared to previous approaches. They tested their approach on 21 websites of Stack exchange having 4 million questions and 8 million answers. They used the ADT learning algorithm to predict the correct answer and achieved 84% of average precision and 70% recall.

Calefato et al. [26] used two datasets: 1) Stack Overflow for the training of the classifier; and 2) Docusign for the testing of the trained classifier. User related features are not used in their model, i.e., the number of accepted answers, badges, and user reputation, as these features do not exist in old developer forums. In their research work, four main categories of feature sets are proposed: 1) linguistic features such as length in character, word count, average characters per word, sentences count, average words per sentence, and URLs; 2) vocabulary such as normalized log-likelihood, i.e., frequency of a word is divided by the number of unique words occurring in a sentence and Flesch-Kincaid grade; 3) metafeatures such as age (time difference between an answer and posted question), rating score (up-votes minus down-votes of each question); and 4) thread features such as answer count (number of answers to a question). They used random trees, random forest, J48 and alternating decision tree algorithms and achieved 63%, 78%, 74%, and 83% of accuracy, respectively. Notably, the proposed approach differs from the existing approaches for correct answer prediction in that we are the first to exploit keyword ranking algorithms with a deep learning based ensemble model in the prediction of correct answers for developer forums.

Elalfy et al. [27] proposed a hybrid model for predicting the best answer. The proposed model consists of two modules. The first module examined the content features (question-answer features, answer content features, and answer-answer features). However, they examined the non-content feature in the second module by using a novel reputation score function. They merged both modules to predict the best answer. They exploited Naive Bayes, Logistic Regression, and random Forest to train and test the proposed model and found Random Forest as the best classifier among them.

In conclusion, researchers have proposed many approaches for the automation of the process of developer forums. However, only two studies [26], [27] focus on the correct answer prediction of developer forum similar to our approach. Moreover, the studies exploit machine learning algorithms for the correct answer prediction of developer forums. Our proposed approach differs from the existing approaches in that we apply a deep learning algorithm for the correct answer prediction of the developer forums by leveraging the ranking algorithm for feature identification.

IV. PROPOSED APPROACH

A. OVERVIEW

The overview of the proposed approach is illustrated in Fig. 1. The steps of the proposed approach for correct answer predictions are following:

1) Metadata (m-features) and the combination of Q/A for each thread are collected from Stack Overflow.

2) The natural language preprocessing techniques are exploited to clean the Q/A combinations of the given dataset.

3) Keywords (k-features) of each Q/A combination are constructed by extracting the keywords and their scores from each Q/A combination using the text ranking algorithm (TextRank).

4) Word embedding (t-features) of each Q/A combination are constructed by leveraging word2vec.

5) Given the m-features, k-features, and t-features of each Q/A combination, a deep learning based ensemble model is trained and tested for correct answer prediction.

B. PROBLEM DEFINITION

An answer \(a\) from a set of \(A\) can be defined as,

\[ a = < m, t > \]  (1)

where \(m\) and \(t\) are m-features and textual information of the thread, respectively. Notably, m-features include length in characters, word count, sentence count, longest sentence, average words per sentence, average character per word, and contained hyperlink (yes/no) in answers posted by users.

An automatic correct answer prediction of a new answer \(a\) could be defined as a mapping function \(f\),

\[ f : a \rightarrow c \]  (2)

\[ c \in \text{correct}, \text{incorrect}, \quad a \in A \]  (3)
where \( c \) is prediction against answer from a classification set (correct; incorrect).

**C. PREPROCESSING**

It is important to apply preprocessing to reduce/remove the noise from the textual information. We exploit Python Natural Language Toolkit (NLTK)\(^1\) for the text preprocessing. The step by step process to reduce noise is as follows:

**Data Cleaning:** The extracted data contains the unwanted information, e.g., HTML tags. In this step of preprocessing, we remove the HTML tags, code snippets, punctuation, and URLs from the textual information of the given dataset.

**Spell Correction and Lowercase Conversion:** The extracted dataset contains textual information and is populated manually by users, i.e., Q/A. Therefore, there is a high probability of spell mistakes. Consequently, we perform spell-check for the textual information to mitigate this threat and convert all text into lowercase.

**Tokenization:** Natural language processing treats each word independently. Therefore, we separate text into separate words where each word represents a token.

**Stop-word Removal:** Stop-words are those words in the text document that occur frequently and do not have actual meaning, e.g., \( is \) and \( am \). Therefore, we remove such words from the textual information of the given dataset.

**Stemming:** In this step of preprocessing, each word is converted into its root word, e.g., \( shouting \) is converted into its root word \( shout \).

**Lemmatization:** Each word is converted into its dictionary form in lemmatization, e.g., the words \( best \), \( good \), and \( better \) have similar dictionary meaning, therefore we transform them into \( good \).

After preprocessing, an answer \( a \) can be defined as,

\[
a = < m, t' >
\]

where \( m \) and \( t' \) are \( m \)-features and preprocessed textual information of the thread, respectively.

**D. KEYWORD EXTRACTION AND RANKING**

To consider the semantics within the textual information, we leverage text ranking algorithm to extract the important words and their ranking score. It is reported that semantic representation of the text can be used to extract and score keywords [5], [28]. There are a number of tools available for the keywords extraction that define the significance of words using statistical methods (e.g., \( TF/IDF \) [29], \( Yake \) [28], \( Rake \) [30], and \( KP-Miner \) [31]), supervised machine learning methods (e.g., \( MAUI \) [32] and \( KEA \) [33]), and unsupervised machine learning methods (e.g., \( ExpandRank \) [34], \( SingleRank \) [34], \( TopicPageRank \) [35], \( PositionRank \) [36], \( TopicRank \) [37], \( MultipartiteRank \) [38], and \( TextRank \) [5]). However, we leverage \( TextRank \) as it has significant performance for short text [39].

\( TextRank \) is a graph-based keyword extraction model in text processing that first considers all words as vertices and identifies the importance of vertex with a graph. Then, it draws edges between words to specify their relationship based on the co-occurrence of words in a separate window and generates the undirected graph from a sequence of text to capture the important information from the graph recursively. \( TextRank \) model assigns a score to each vertex (word) based on the voting system using the corresponding vertexes. Furthermore, the higher the casted votes for a vertex, shows higher the importance for that vertex (word). The score computation method for a vertex can be defined as,

\[
S(V_i) = (1 - d) + d \sum_{j \in \text{in}(V_i)} \frac{1}{\text{Out}(V_j)} S(V_j)
\]

\(^1\)https://www.nltk.org/
where $V_i$ represents the set of vertices of a directed graph, 
$ln(V_i)$ represents the set of vertices that point to it (predecessors of $V_i$), $Out(V_i)$ represents the set of vertices that point to the successors of vertex $V_i$, and $d$ is a damping factor having a value in between [0-1].

We pass each Q/A combination as an input to TextRank to extract and score the keywords. Notably, Python NLTK packages ($spacy$, $pytextrank$, and $en_core_web_sm$) are exploited to extract and score the keywords. Each Q/A combination is passed to $textrank nlp pipe$ that returns the keywords and their scores as an output. Based on the extracted keywords and their scores, we construct a vector of $k$-features. The length of the vector is variable as the extracted keywords against each Q/A combination is different. After keywords extraction, an answer $a$ can be defined as,

$$a = < m, t', k > \quad (6)$$

where $m$, $t'$, and $k$ are $m$-features, preprocessed textual information of the thread, and $k$-features extracted from $t'$, respectively.

E. WORD EMBEDDING

To input the textual information of each Q/A combination (other than $k$-features) to the proposed model, the preprocessed text is converted into the fixed length (300) numerical vectors. We leverage a pre-trained skip-gram model (word2vec), proposed by Mikolov et al. [40], due to its significant efficiency to learn high-quality distributed vector representation. It returns the $n$-dimensional vectors by capturing the syntax and semantic relationships among words. We exploit the hidden layer of the given trained network to convert words into numerical vectors. The overview of the model is presented in Fig. 2.

After word embedding, an answer $a$ can be defined as,

$$a = < m, t'', k > \quad (7)$$

where $m$, $t''$, and $k$ are $m$-features, $t$-features extracted from $t'$, and $k$-features extracted from $t''$, respectively.

F. THE ENSEMBLE DEEP LEARNING MODEL

The composition of the ensemble deep learning model is presented in Fig. 3. We construct the ensemble deep learning model by exploiting $Keras$ (the Python deep learning API) that includes three deep learning classifiers: the convolutional neural network (CNN) based classifier and two long short term memory (LSTM) based classifiers. Note that different combinations of deep learning classifiers are tested to construct the ensemble model. However, the proposed model yields the best among them. The model receives vectors of $m$-features, $k$-features, and $t$-features against each Q/A combination as input and predicts the correct answer.

We pass the vectors of $m$-features, $k$-features, and $t$-features to the proposed model into three parts. We feed $m$-features to $CNN$, and $k$-features, and $t$-features to LSTM classifiers, respectively. We use CNN with settings:

- $filter = 128$, $kernel size = 1$, and $activation = tanh$, where, $filter$, $kernel size$, and $activation$ are the number of neurons, size of the filter, and final value of neuron, respectively. Each neuron acts as a different convolution on the input to the layer. After the convolutions and dropouts the output is forwarded to a dense layer. Moreover, we use LSTM model with settings: $dropout = 0.2$, recurrent dropout = $0.2$, and $activation = sigmoid$. The flatten layer [41] is added in LSTM where textual features ($t$-features) are passed. Finally, the outputs of all three classifiers are passed to a merge layer [41] that performs concatenation [42] to combine them. Notably, we use $binary-crossentropy$ as the loss function for the ensemble architecture after concatenation of the extracted features. The final dense layer maps all three outputs into a single output (prediction).

V. EVALUATION

This section presents the research questions, dataset, metrics, and process used to evaluate the proposed approach.

A. RESEARCH QUESTION

The proposed approach is evaluated by investigating the following research questions:

- Q1: Does the proposed approach outperform state-of-the-art approaches in the prediction of correct answers for developer forums?
- Q2: Does text ranking influence the performance of the proposed approach?
- Q3: Does the preprocessing of textual information influence the performance of the proposed approach?
- Q4: Does re-sampling help to improve the performance of the proposed approach?
- Q5: Does the proposed approach outperform the other traditional machine/deep learning algorithms?

The RQ1 investigates the performance of the proposed approach by comparing the evaluation results with the state-of-the-art approach. We select the correct/best
answer prediction (BAP) approach [26] proposed by Fabio Calefato et al. and a hybrid approach to predict correct/best answers (HAP) [27] proposed by Dalia Elalfy et al. because these are the most related study to our approach. The proposed approach differs from BAP and HAP. To the best of our knowledge, we are the first to exploit keyword ranking algorithms with an ensemble deep learning model to predict correct answers for developer forums.

The RQ2 compares the evaluation results of the proposed approach on different input settings to compute the impact of TextRank on the performance of the proposed approach.

The RQ3 compares the evaluation results of the proposed approach with and without preprocessing to find out the impact of preprocessing on the performance of the proposed approach.

The RQ4 investigates the influence of re-sampling to mitigate the threat of an imbalanced dataset. To this end, we apply under-sampling and over-sampling to balance our dataset and compare the proposed approach’s evaluation results.

The RQ5 compares the evaluation results of different machine/deep learning algorithms to highlight the significant performance of the proposed approach for correct answer prediction.

**B. DATASET**
We exploit the Stack Overflow open source Q/A dataset for the evaluation of the proposed approach. We download the dataset from the Stack Overflow released on September, 2019. The dataset contains 17 M multi-tagged questions and 26 M million answers. Notably, the dataset is skewed towards incorrect answers, i.e., only 10% are marked as correct. Therefore, we apply filter of five tags (java,.net, php, ruby, and misc) and reduce the dataset that contains 236,000 answers and 91,500 question threads, where 70,800 (30%) and 165,200 (70%) answers are correct and incorrect, respectively.

**C. PROCESS**
The performance of the proposed approach is evaluated as follows: We first collect and reuse the Stack Overflow dataset. Then, we apply natural language preprocessing techniques to clean the noise from the textual information of the dataset. Then, we construct three input vectors of each Q/A combination. After that, we divide A into ten segments notated as $A_i (1, 2, \ldots, 10)$ to carry out the 10-fold cross-validation. For each $i^{th}$ cross-validation, we take $i^{th}$ segment as testing set ($S_t$) and the rest of the segments as training set ($S_r$). The evaluation process for each cross-validation is as follows,

1) We collect $S_r$ by taking a union of all segments but $S_i$.
2) Different machine learning algorithms (multi-nomial naive Bayes (MNB), logistic regression (LR), random forest (RF), and decision trees (DT)) and deep learning classifiers (proposed ensemble model, BAP, and HAP) are trained using $S_r$. Notably, we employ the same dataset for the replication/evaluation of BAP and HAP.
3) Each $S_i$ from $S_r$ is predicted with the trained machine learning and deep learning classifiers.
4) Evaluation metrics are computed for each machine/deep learning classifier to compare their performances.

**D. METRICS**
The performance of the proposed approach is evaluated using the well-known and most adopted metrics (accuracy, precision, recall, and f-measure) for the machine/deep learning based classifiers. The selected metrics can be defined as,

$$\text{Acc} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\text{Pre} = \frac{TP}{TP + FP}$$

$$\text{Rec} = \frac{TP}{TP + FN}$$

$$\text{FM} = \frac{2 \cdot \text{Pre} \cdot \text{Rec}}{\text{Pre} + \text{Rec}}$$

where, Acc, Pre, Rec, and FM represent the accuracy, precision, recall, and f-measure, respectively. Moreover, true positive (TP) are the correct answers that are correctly predicted, true negative (TN) are the incorrect answers that are correctly predicted, false positive (FP) are the correct answers that are incorrectly predicted, and false negative (FN) are the incorrect answers that are incorrectly predicted.
TABLE 1. Performance of the Proposed Approach, BAP and HAP.

| Approach | Accuracy | Precision | Recall | F-Measure |
|----------|----------|-----------|--------|-----------|
| Proposed | 84.39%   | 96.16%    | 84.52% | 89.97%    |
| BAP      | 82.96%   | 76.95%    | 79.31% | 77.15%    |
| HAP      | 72.43%   | 74.35%    | 78.69% | 76.46%    |

E. RESULTS

RQ1: COMPARISON OF THE PROPOSED APPROACH AGAINST STATE-OF-THE-ART

To investigate RQ1, we compare the performances of the proposed approach, BAP, and HAP. Note that we exploit the transformer based approach, e.g., BERT (a pre-trained model) [43] for the comparison of the proposed result, but the experimental results were not satisfactory. Therefore, we ignore the transformer based approaches for the evaluation of the proposed approach. One possible reason for this performance decrease is that BERT does not work well with software engineering datasets.

The evaluation results of the both approaches are presented in Table 1. The performance results of the proposed approach specify that average accuracy, precision, recall and f-measure are 84.39%, 96.16%, 84.52%, and 89.97%, respectively. Similarly, the performance results of BAP and HAP specify that the average accuracy, precision, recall, f-measure are (82.96%, 76.95%, 79.31%, and 77.15%) and (72.43%, 74.35%, 78.69%, and 76.46%), respectively.

The f-measure distribution of 10-fold evaluation of the proposed approach, BAP, and HAP is presented in Fig. 4. Each bean is plotted for the comparison of the approaches. Each small horizontal line in each bean represents the f-measure of cross-validation, and the long horizontal line in each bean represents the average f-measure of 10-fold cross-validation.

From Table 1 and Fig. 4, we notice the following:

- The proposed approach surpasses BAP and HAP in correct answer prediction.
- The improvement of the proposed approach in contrast to BAP in accuracy, precision, recall, and f-measure is 1.72% = (84.39% - 82.96%) / 82.96%, 24.96% = (96.16% - 76.95%) / 76.95%, 6.57% = (84.52% - 79.31%) / 79.31%, and 16.62% = (89.97% - 77.15%) / 77.15%, respectively.
- The improvement of the proposed approach in contrast to HAP in accuracy, precision, recall, and f-measure is 16.51% = (84.39% - 72.43%) / 72.43%, 29.33% = (96.16% - 74.35%) / 74.35%, 7.41% = (84.52% - 78.69%) / 78.69%, and 17.67% = (89.97% - 76.46%) / 76.46%, respectively.
- The f-measure distribution of the proposed approach does not dramatically fluctuate in contrast to BAP and HAP that ensures the reliability of the proposed approach.

Moreover, we examine the significant difference between the proposed approach and BAP by performing a one-way analysis of variance (ANOVA) and the Wilcoxon test. Note that we select the best two approaches for these statistical tests. We apply and evaluate both approaches on the same dataset. Therefore, we select ANOVA to investigate the significant difference between approaches, whereas we perform the Wilcoxon test to verify the significant difference. The ANOVA and Wilcoxon test results check whether the only difference (single factor, i.e., different approaches) leads to the difference in performance. Note that we perform ANOVA on Excel and Wilcoxon test on Stata using their default settings of parameters. The f-ratio value and p-value of ANOVA are 273.71 and 2.47E-12, whereas the p-value of Wilcoxon test is 8.36E-11. The results of both ANOVA and Wilcoxon test specify that the factor (using different approaches) has a significant difference at p < 0.05.

Although the proceeding analysis specifies that the proposed approach is accurate, we notice many false positives and false negatives. For example, the proposed approach falsely predicts the incorrect answer “You need to release the LocationManager. Make sure that you …find your location accurately enough.” as correct and correct answer “I don’t think there is a correct size, Since the iPhone really is running OSX …the icon looks pixel perfect on the iPhone screen.” as incorrect, for the question “I need to get my current location using GPS programatically. How can I achieve it?”.

To investigate the rationale of false positives ad false negatives, we randomly select 2000 samples of Q/A combinations and manually check their correctness. We observe that some software engineering words, e.g., robust have low significance and are not selected as keywords by TextRank. Note that, TextRank is not designed for software engineering text that could be the reason for false-classifications. However, we have not fully understood the rationale for false classifications. In future, we would like
to investigate the rationale for false classifications to reduce them.

The preceding analysis concludes that the proposed approach surpasses the state-of-the-art for correct answer prediction.

**RQ2: INFLUENCE OF TEXT RANKING**

To investigate RQ2, we compare the performances of the proposed approach with different input settings. The evaluation results of the proposed approach against different input settings are presented in Table 2. The first column of Table 2 presents the preprocessing settings. Columns 2-5 of Table 2 present evaluation results of performance metrics, respectively. Each row of Table 2 presents the evaluation results of the proposed approach for each input setting, respectively.

From Table 2, we notice the following:

- Text alone (i.e., without metadata and keywords) is insufficient for correct answer prediction. Compared to the default setting (i.e., enabling metadata, keywords, and text), disabling metadata and keywords decreases accuracy significantly from 84.39% to 73.52% and f-measure from 89.97% to 80.64%.
- Keywords alone (i.e., without metadata and text) is insufficient for correct answer prediction. Compared to the default setting, disabling metadata and text decreases accuracy significantly from 84.39% to 76.33% and f-measure from 89.97% to 80.64%.
- Metadata alone (i.e., without keywords and text) is insufficient for correct answer prediction. Compared to the default setting, disabling metadata and text decreases accuracy significantly from 84.39% to 73.52% and f-measure from 89.97% to 80.64%.
- Disabling text (i.e., metadata and keywords) results in significant reduction in performance. It significantly reduces accuracy and f-measure up to 2.34% = (84.39% - 82.46%) / 82.46% and 4.66% = (89.97% - 85.96%) / 85.96%, respectively.
- Disabling keywords (i.e., metadata and text) also results in significant reduction in performance. It significantly reduces accuracy and f-measure up to 1.52% = (84.39% - 83.13%) / 83.13% and 3.82% = (89.97% - 86.66%) / 86.66%, respectively.
- Disabling metadata (i.e., keywords and text) also results in significant reduction in performance. It significantly reduces accuracy and f-measure up to 4.65% = (84.39% - 80.64%) / 80.64% and 9.81% = (89.97% - 81.97%) / 81.97%, respectively.

The preceding analysis concludes that all inputs (metadata, keywords, and text) are significantly important for the proposed approach.

**RQ3: INFLUENCE OF PREPROCESSING**

To investigate RQ3, we compare the performances of the proposed approach by enabling and disabling preprocessing. The evaluation results of the proposed approach against different preprocessing settings are presented in Table 3. The first column of Table 3 presents the preprocessing settings. Columns 2-5 of Table 3 present evaluation results of performance metrics, respectively. Each row of Table 3 presents the evaluation results of the proposed approach for each preprocessing setting, respectively.

From Table 3, we notice the following:

- The evaluation results of the proposed approach by enabling preprocessing specify the significant improvement in accuracy, precision, recall, and f-measure.
- The performance improvement in accuracy, precision, recall, and f-measure is up to 2.95% = (84.39% - 81.97%) / 81.97%, 1.20% = (96.16% - 95.02%) / 95.02%, 2.92% = (84.52% - 82.12%) / 82.12%, and 2.12% = (89.97% - 88.10%) / 88.10%, respectively.

The preceding analysis concludes that the preprocessing of the textual information is critical for the proposed approach. The noise (i.e., disabling preprocessing) affects the performance of the proposed approach.

**RQ4: INFLUENCE OF RE-SAMPLING**

To investigate RQ4, we re-sample the imbalanced dataset (as mentioned in Section V-B) to make it balance and compare the performances of the proposed approach with and without re-sampling. Note that we use Synthetic Minority Over-sampling Technique (SMOTE) [45] for minority class which finds k-nearest neighbors to randomly choose samples for minority class, whereas we select n random samples from majority class for under-sampling to balance both classes (incorrect/correct) completely. The evaluation results of the proposed approach with and without a re-sampled dataset are presented in Table 4. The first column of Table 4 presents the re-sampling methods. Columns 2-5 of Table 4 present evaluation results of performance metrics, respectively. Each row of Table 4 presents the evaluation results of the proposed approach for each re-sampling method, respectively.

Table 4 shows that the results of under-sampling and over-sampling of average accuracy, precision, recall, and
f-measure are (73.14%, 74.56%, 73.38%, and 73.97%) and (87.95%, 89.11%, 87.70%, and 88.40%), respectively. We notice the significant difference in the performance of the proposed approach on balanced and imbalanced datasets. The results specify that over-sampling only improves the accuracy and recall of the proposed approach, whereas under-sampling does not increase the performance of the proposed approach. The possible reason for the decrease in performance is that the negative data (incorrect answer) are reduced with re-sampling. Consequently, the machine contributes a few negative predictions (more positive predictions) and results in performance decrease of the proposed approach, i.e., the average accuracy, precision, recall, and f-measure for the correct and incorrect labels are posed approach, i.e., the average accuracy, precision, recall, and f-measure for the correct and incorrect labels are (80.34%, 90.77%, 78.12%, and 83.95%) and (65.94%, 58.35%, 68.64%, and 63.02%), respectively.

**TABLE 5: Influence of Classification Algorithms.**

| Classifiers | Accuracy | Precision | Recall | F-Measure |
|-------------|----------|-----------|--------|-----------|
| Proposed    | 84.39%   | 96.16%    | 84.52% | 89.97%    |
| CNN         | 82.20%   | 95.71%    | 82.25% | 88.47%    |
| LSTM        | 82.53%   | 93.29%    | 83.67% | 88.22%    |
| MNB         | 67.88%   | 65.88%    | 68.92% | 67.37%    |
| LR          | 81.15%   | 76.33%    | 81.19% | 78.69%    |
| RF          | 72.43%   | 74.35%    | 78.69% | 76.46%    |
| DT          | 71.56%   | 72.61%    | 71.22% | 71.91%    |

RQ5: COMPARISON AGAINST CLASSIFICATION ALGORITHMS

To investigate RQ5, we compare the performances of the proposed approach with traditional machine learning (MNB, LR, RF, and DT) and deep learning classifiers (CNN and LSTM). Note that we select these classifiers because they have significant performances for software engineering text classification [46]–[50]. The evaluation results of all classifiers are presented in Table 5. The first column of Table 5 presents the classification algorithms. The columns 2-5 of Table 5 present the evaluation results of performance metrics, respectively. Each row of Table 5 presents the evaluation results of the proposed approach of each classifier, respectively. The average accuracies of the classifiers are 84.39%, 82.20%, 82.53%, 67.88%, 81.15%, 72.43%, and 71.56%, respectively.

From Table 5, we make the observations as follows:

- The proposed approach surpasses the deep learning classifiers. The performance improvement in accuracy and f-measure varies from 2.25% = (84.39% - 82.53%) / 82.53% to 2.66% = (84.39% - 82.20%) / 82.20% and varies from 1.70% = (89.97% - 88.47%) / 88.47% to 1.98% = (89.97% - 88.22%) / 88.22%, respectively.
- The proposed approach surpasses the machine learning classifiers. The performance improvement in accuracy and f-measure varies from 3.99% = (84.39% - 81.15%) / 81.15% to 24.32% = (84.39% - 67.88%) / 67.88% and varies from 14.33% = (89.97% - 78.69%) / 78.69% to 33.55% = (89.97% - 67.37%) / 67.37%, respectively.
- The performance of the proposed approach is better than LR (the best machine learning classifier). The possible reason is that LR does not work for the variable input dimensions.
- The performances of the proposed approaches and the deep learning classifier are very close. However, we have not fully understood the rationale for such similarity. In future, we would like to investigate the rationale for better understanding.

The preceding analysis concludes that the proposed approach surpasses other machine/deep learning classifiers to predict correct answers.

VI. THREATS

A. THREATS TO VALIDITY

The first threat to construct validity is the adoption of evaluation metrics: accuracy, precision, recall, and f-measure. Note that we select these metrics because of their popularity (as mentioned in Section V-D) to evaluate the proposed solution of the text classification problems in machine/deep learning [51].

Another threat to construct validity is the leverage of the keywords extraction tool TextRank to extract and rank keywords from the textual information of the given dataset. Note that we leverage TextRank because it is reported best among other keywords extraction tools especially for short text as mentioned in Section IV-D. However, the usage of different keyword extraction tools may influence the performance of the proposed approach.

The first threat to internal validity is the implementation of the proposed and baseline approaches. We double-check the implementation of the approaches and their results to mitigate the threat. Nevertheless, the code may have some unseen errors.

The first threat to external validity is related to the abstraction of the proposed approach. Note that we use Stack Overflow dataset for the evaluation of the proposed approach as mentioned in Section V-B. Other datasets may influence the performance of the proposed approach.

Another threat to external validity is related to the optimization of hyper-parameters for the ensemble deep learning model. The tuning and adjustment of hyper-parameters may influence the performance of the proposed approach.
VII. CONCLUSION AND FUTURE WORK

This paper proposes an ensemble deep learning approach to predict the correct answers for developer forums. We extract the dataset from the developer community (Stack Overflow) and preprocess using natural language preprocessing techniques. After that, we leverage the keywords extraction tool to extract and rank keywords. Subsequently, we leverage word embedding to convert the textual information. Given the vectors (metadata, keywords, and text), we train the proposed ensemble deep learning based classifier. The evaluation results of the 10-fold cross-validation specify that the proposed approach is accurate and surpasses the state-of-the-art.

The broader impact of our work is to show that the combination of Q&A could be a rich source to predict the correct answer prediction. Our results encourage future research on correct answer prediction. The results’ closeness of the proposed ensemble model and other deep learning models is still unclear. In future, we would like to investigate the rationale behind that and figure out how the machines learn and deduce conclusions.

REFERENCES

[1] M. Squire, “Should we move to stack overflow? Measuring the utility of social media for developer support,” in Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng., vol. 2, May 2015, pp. 219–228.
[2] S. Grant and B. Betts, “Encouraging user behaviour with achievements: An empirical study,” in Proc. 10th Work. Conf. Mining Softw. Repositories (MSR), May 2013, pp. 65–68.
[3] L. Manykina, B. Cao, J. Mittal, G. Hripscak, and B. Hartmann, “Design lessons from the fastest QA site in the west,” in Proc. SIGCHI Conf. Hum. Factors Comput. Syst., May 2011, pp. 2857–2866.
[4] Stack Exchange. (Nov. 2021). Stackoverflow. [Online]. Available: https://stackoverflow.com/
[5] R. Mihalceas and P. Tarau, “TextRank: Bringing order into text,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2004, pp. 404–411.
[6] I. Mourtizis and H. T. P. Williams, “Community evolution on stack overflow,” PLoS ONE, vol. 16, no. 6, pp. 1–23, Jun. 2021, doi: 10.1371/journal.pone.0253010.
[7] P. K. Roy, Z. Ahmad, J. P. Singh, M. A. A. Alryalat, N. P. Rana, and Y. K. Dwiwedi, “Finding and ranking highest-quality answers in community question answering sites,” Global J. Flexible Syst. Manage., vol. 19, no. 1, pp. 53–68, Mar. 2018.
[8] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Stenkenste, and C. Faloutsos, “Analysis of the reputation system and user contributions on a question answering website: StackOverflow,” in Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM), New York, NY, USA, Aug. 2013, pp. 886–893, doi: 10.1109/292517.2500242.
[9] A. Anderson, D. Hutterlocher, J. Kleinberg, and J. Leskovec, “Discovering value from community activity on focused question answering sites: A case study of stack overflow,” in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), New York, NY, USA, 2012, pp. 850–858, doi: 10.1145/2399350.2336965.
[10] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft, “Building reputation in StackOverflow: An empirical investigation,” in Proc. 10th Work. Conf. Mining Softw. Repositories (MSR), May 2013, pp. 89–92.
[11] W. Zheng and M. Li, “The best answer prediction by exploiting heterogeneous data on software development Q&A forum,” Neurocomputing, vol. 269, pp. 212–219, Dec. 2017.
[12] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, and D. Fullerton, “Improving low quality stack overflow post detection,” in Proc. IEEE Int. Conf. Softw. Maintenance Evol., Sep. 2014, pp. 541–544.
[13] X. Xia, D. Lo, D. Correa, A. Sureka, and E. Shihab, “It takes two to tango: Deleted stack overflow question prediction with text and meta features,” in Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC), vol. 1, Jun. 2016, pp. 73–82.
[14] C. Treude, O. Barzilay, and M.-A. Storey, “How do programmers ask and answer questions on the web (NIER track),” in Proc. 33rd Conf. Softw. Eng., May 2011, pp. 804–807.
[15] F. Calefato, F. Lanubile, M. C. Marasciulo, and N. Novielli, “Mining successful answers in stack overflow,” in Proc. 12th Work. Conf. Mining Softw. Repositories, May 2015, pp. 430–433.
[16] F. Calefato, F. Lanubile, and N. Novielli, “How to ask for technical help? Evidence-based guidelines for writing questions on stack overflow,” Inf. Softw. Technol., vol. 94, pp. 186–207, Feb. 2018.
[17] K. Hart and A. Sarma, “Perceptions of answer quality in an online technical question and answer forum,” in Proc. 7th Int. Workshop Cooperat. Hum. Aspects Softw. Eng., Jun. 2014, pp. 103–106.
[18] A. Barua, S. W. Thomas, and A. E. Hassan, “What are developers talking about? An analysis of topics and trends in stack overflow,” Empirical Softw. Eng., vol. 19, no. 3, pp. 619–654, 2014.
[19] L. A. Adamic, J. Zhang, E. Bakshy, and S. M. Ackerman, “Knowledge sharing and Yahoo answers: Everyone knows something,” in Proc. 17th Int. Conf. World Wide Web, 2008, pp. 665–674.
[20] C. Shah, “Building a parsimonious model for identifying best answers using interaction history in community Q&A,” in Proc. 78th ASIS&T Annu. Meeting, Inf. Sci. Impact. Res. Community, Medford, NJ, USA: American Society for Information Science, 2015, p. 51.
[21] C. Shah and J. Pomerantz, “Evaluating and predicting answer quality in community QA,” in Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., 2010, pp. 411–418.
[22] Q. Tian, P. Zhang, and B. Li, “Towards predicting the best answers in community-based question-answering services,” in Proc. 7th Int. AAAI Conf. Weblogs Social Media, 2013, pp. 1–4.
[23] Y. Cai and A. Chakravarty, “Predicting answer quality in Q/A social networks: Using temporal features,” Dept. Comput. Sci. Eng., Univ. Texas Arlington, Arlington, TX, USA, Tech. Rep. 721353, 2011.
[24] G. Burel, Y. He, and H. Alani, “Automatic identification of best answers in online enquiry communities,” in Proc. Extended Semantic Web Conf. Crete, Greece: Springer, May 2012, pp. 514–529.
[25] G. Gkotsis, “ACQUA: Automated community-based question answering through the discretization of shallow linguistic features,” J. Web Sci., vol. 1, no. 1, pp. 1–15, Jun. 2015.
[26] F. Calefato, F. Lanubile, and N. Novielli, “Moving to stack overflow: Best-answer prediction in legacy developer forums,” in Proc. 10th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas., Sep. 2016, p. 13.
[27] D. Elaffy, W. Gao, and R. Ismail, “A hybrid model to predict best answers in question answering communities,” Egyptian Inform. J., vol. 19, no. 1, pp. 31–39, Mar. 2013.
[28] R. Campos, V. Mangaravite, A. Pasqualli, A. M. Jorge, C. Nunes, and A. Jatowt, “A text feature based automatic keyword extraction method for single documents,” in Proc. Eur. Conf. Inf. Retr. Grenoble, France: Springer, 2018, pp. 684–691.
[29] J. Ramos, “Using TF-IDF to determine word relevance in document queries,” in Proc. 1st Instruct. Conf. Mach. Learn., vol. 242, Piscataway, NJ, USA, 2003, pp. 137–142.
[30] S. Rose, D. Engel, N. Cramer, and W. Cowley, “Automatic keyword extraction from individual documents,” Text Mining, Appl. Theory, vol. 1, pp. 1–20, Mar. 2010.
[31] S. El-beltagy, “KP-miner: A simple system for effective keyphrase extraction,” in Proc. Innow. Inf. Technol., Nov. 2006, pp. 1–5.
[32] O. Medelyan, V. Perrone, and I. H. Witten, “Subject metadata support powered by Maui,” in Proc. JCDL, 2010, pp. 407–408.
[33] I. H. Witten, E. Frank, and M. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed. Burlington, MA: Morgan Kaufmann, 2011.
[34] C. Florescu and C. Caragea, “PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents,” in Proc. 55th Annu. Meeting Assoc. Comput. Linguistics, 2017, pp. 1105–1115.
[35] Z. Liu, W. Huang, Y. Zheng, and M. Sun, “Automatic keyphrase extraction via topic decomposition,” in Proc. Conf. Empirical Methods Natural Lang. Process. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 366–376.
[36] C. Florescu and C. Caragea, “PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents,” in Proc. 55th Annu. Meeting Assoc. Comput. Linguistics, 2017, pp. 1105–1115.
[37] A. Bougouin, F. Boudin, and B. Daille, “TopicRank: Graph-based topic ranking for keyphrase extraction,” in Proc. Int. Joint Conf. Natural Lang. Process., 2013, pp. 543–551.
G. Forman, “Unsupervised keyphrase extraction with multipartite graphs,” 2018, arXiv:1803.08721. [Online]. Available: http://arxiv.org/abs/1803.08721

W. Li and J. Zhao, “TextRank algorithm by exploiting Wikipedia for short text keywords extraction,” in Proc. 3rd Int. Conf. Inf. Sci. Control Eng., Jul. 2016, pp. 683–686.

T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013, arXiv:1301.3781. [Online]. Available: http://arxiv.org/abs/1301.3781

Keras. (Nov. 2021). Merge Layer. [Online]. Available: https://github.com/keras-team/keras/blob/master/keras/layers/merge.py

Q. Umer, H. Liu, and Y. Sultan, “Sentiment based approval prediction for bug reports,” IEEE Access, vol. 6, pp. 35743–35752, 2018.

H. Batra, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, “BERT based sentiment analysis: A software engineering perspective,” CoRR, vol. abs/2106.02581, pp. 1–11, Jun. 2021. [Online]. Available: https://arxiv.org/abs/2106.02581

Q. Umer, H. Liu, and I. Illahi, “CNN-based automatic prioritization for language understanding,” CoRR, vol. abs/1810.04805, pp. 1–16, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

S. J. Sohrawardi, I. Azam, and S. Hosain, “A comparative study of text classification algorithms on user submitted bug reports,” in Proc. 9th Int. Conf. Digit. Inf. Manage. (ICDIM), Sep. 2014, pp. 242–247.

Q. Umer, H. Liu, and I. Illahi, “Sentiment based approval prioritization for enhancement reports,” J. Syst. Softw., vol. 155, pp. 57–69, Sep. 2019.

Q. Umer, H. Li, and Y. Sultan, “Emotion based automated priority prediction for bug reports,” IEEE Access, vol. 6, pp. 35743–35752, 2018.

F. Boudin, “Unsupervised keyphrase extraction with multipartite graphs,” 2018, arXiv:1803.08721. [Online]. Available: http://arxiv.org/abs/1803.08721

T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013, arXiv:1301.3781. [Online]. Available: http://arxiv.org/abs/1301.3781

J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” CoRR, vol. abs/1810.04805, pp. 1–16, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

H. U. Iftikhar

HAFIZ UMAR IFTIKHAR received the B.S. degree in computer science from COMSATS University Islamabad, Pakistan, and the M.S. degree in computational science and engineering from the National University of Sciences and Technology, Pakistan. He is currently an Artificial Intelligence Engineer having more than two years of industrial experience in solving computer vision problems. His major research interests include computer vision, machine learning, and natural language processing.

AQEEL UR REHMAN received the M.Sc. degree in computer science from The Islamia University Bahawalpur, the master’s degree in computer engineering from UET Taxila, CASE Campus, Islamabad, Pakistan, and the Ph.D. degree in computer science and technology from Chongqing University, Chongqing, China. He worked as an Assistant Professor with the Computer Science Department, COMSATS University Islamabad, Vehari Campus, Pakistan. He is currently on study leave with the Computer Science Department, COMSATS University Islamabad. He is also working as a Senior Research Fellow with Southwest University, Chongqing. He has published more than 15 research articles in impact factor journals. His primary areas of research interests include non-linear dynamics and cryptography. He is a reviewer of reputed journals, such as Optics and Laser Technology, Optics and Lasers in Engineering, and International Journal Engineering Science and Technology.

OLGA A. KALUGINA received the degree from Mordovian State Pedagogical Institute named after M. E. Evseev, Russia, in 2005. She is currently an Associate Professor in the area of professional cross-cultural communication and economics with the Financial University Under the Government of the Russian Federation. She is interested in environmental economics and sustainable development. Her current research interest includes development of practical tools for software engineers.

AQEEL UR REHMAN

AQEEL UR REHMAN received the B.S. degree in computer science from Punjab University, Pakistan, in 2006, the M.S. degree in .NET distributed system development and the M.S. degree in computer science from the University of Hull, U.K., in 2009 and 2013, respectively, and the Ph.D. degree from Beijing Institute of Technology, China. He is currently an Assistant Professor with the Computer Science Department, COMSATS University Islamabad, Vehari Campus, Pakistan. His research interests include machine learning, data mining, software maintenance, and developing practical tools to assist software engineers.

HAFIZ UMAR IFTIKHAR

HAFIZ UMAR IFTIKHAR received the B.S. degree in computer science from COMSATS University Islamabad, Pakistan, and the M.S. degree in computational science and engineering from the National University of Sciences and Technology, Pakistan. He is currently an Artificial Intelligence Engineer having more than two years of industrial experience in solving computer vision problems. His major research interests include computer vision, machine learning, and natural language processing.

AQEEL UR REHMAN

AQEEL UR REHMAN received the B.S. degree in computer science from COMSATS University Islamabad, Vehari Campus, Pakistan, in 2016, and the M.Phil. degree in computer science from the National College of Business Administration and Economics (NCBA&E), Pakistan, in 2019. He is currently a Research Associate with the Computer Science Department, COMSATS University Islamabad. His research interests include software product line applications and quality assurance.