Information Sciences 600 (2022) 144–154

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Multi-view approach to suggest moderation actions in community question answering sites

Issa Annamoradnejad, Jafar Habibi [*], Mohammadamin Fazli

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

A R T I C L E I N F O

Article history:
Received 16 May 2021
Received in revised form 27 March 2022
Accepted 30 March 2022
Available online 4 April 2022

Keywords:
Automatic moderation
User-generated content
Community question answering
Multi-view learning
Decision support system
Stack overflow

A B S T R A C T

With thousands of new questions posted every day on popular Q&A websites, there is a need for automated and accurate software solutions to replace manual moderation. In this paper, we address the critical drawbacks of crowdsourcing moderation actions in Q&A communities and demonstrate the ability to automate moderation using the latest machine learning models. From a technical point, we propose a multi-view approach that generates three distinct feature groups that examine a question from three different perspectives: 1) question-related features extracted using a BERT-based regression model; 2) context-related features extracted using a named-entity-recognition model; and 3) general lexical features derived using statistical and analytical methods. As a last step, we train a gradient boosting classifier to predict a moderation action. For evaluation purposes, we created a new dataset consisting of 60,000 Stack Overflow questions classified into three choices of moderation actions. Based on cross-validation on the novel dataset, our approach reaches 95.6% accuracy as a multiclass task and outperforms all state-of-the-art and previously-published models. Our results clearly demonstrate the high influence of our feature generation components on the overall success of the classifier.

1. Introduction

Question-answering (Q&A) websites include a variety of user-generated content, ranging from simple documented guides to highly experienced-based answers. While the content is not presented as step-by-step guidelines, these domain-specific systems are being considered as primary or auxiliary tools of learning and fault-finding for mid-level and high-level technicians.

Community Q&A websites must maintain their content quality by editing incomprehensible questions or removing spam and duplicate content in order to remain a citable source and trustworthy encyclopedia for the target audience. High-quality content will improve the site's reputation, attract more visitors, provide a better user experience, encourage experts to answer questions that challenge their expertise, and finally, boost web result rankings [1]. Maintaining quality standards is primarily accomplished by keeping questions as clean and clear as possible (e.g., editing posts to fix grammar, adding factual data, examples, or code samples), closing off-topic/opinion-based/spam content, removing duplicate questions, and protecting high-quality questions from community changes.

Abbreviations: Q&A, Question and answering; NER, Named-Entity Recognition; SoftNER, Software-related named entity recognition.
[*] Corresponding author at: Computer Engineering Department, Sharif University of Technology, P.O. Box 11155-9517, Tehran 14588 89694, Iran.
E-mail addresses: imoradnejad@ce.sharif.edu (I. Annamoradnejad), jhabibi@sharif.edu (J. Habibi), Fazli@sharif.edu (M. Fazli).

https://doi.org/10.1016/j.ins.2022.03.085
0020-0255/© 2022 Elsevier Inc. All rights reserved.

Popular Q&A websites, such as the Stack Exchange[1] family of websites and Quora[2], currently depend on crowdsourcing as a practical way to ensure their quality policies, which in turn is dependent on the engagement of users in moderation tasks. The overall success of these websites has been attributed to various community features such as voting and commenting [2], and a well-designed system of badges and reputation points [3]. The questions are continuously monitored by users and will be flagged, edited, or closed in the event they do not comply with the quality criteria. While this is a practical and feasible approach, depending on community feedback to maintain content quality has serious drawbacks. First, fixing a problem will take time, as it has to be noted by the community, flagged by readers, and voted on by experienced users for a final decision. Valid feedback will cost the time of several users in the process of decision-making, and incorrect feedback (mostly to gain gamification rewards of badges) will result in wasting more time and energy from experienced users. In addition, the subjectivity of decisions by those who flag the questions and those who vote for a final decision can result in disputes and a re-voting process.

With the popularity of text mining and natural language processing methods in recent years, some researchers have utilized the latest machine learning (ML) models to tackle this problem. However, previous research suffers from a few important issues that make them less applicable or helpful in real environments. First, they use community feedback, such as votes, comments, or answers, to predict the quality of a post and suggest moderation actions. Such models cannot be applied to new questions (especially as a recommender system before initial publication), as they depend on community feedback for an accurate recommendation. In addition, the majority of previous works include user-specific features such as previous questions and votes. Similarly, while they would positively affect the accuracy of the recommender system, these features are not applicable to new users who require the highest level of editing and guidance. In fact, such features are suitable for decision support systems that help to pinpoint possible weak posts rather than decide a moderation action based on new content. Furthermore, the high volatility of the question subjects should be considered in creating a fast-adaptable model by reducing the training time [4]. Finally, the proposed approaches should consider the context of the website, instead of solely relying on general textual features and ML models. For example, asking opinion-based questions or questions with examples may be prohibited, allowed, or encouraged based on the user rules of a given website.

In this paper, we aim to build a classifier to separate the given questions based on their quality in the context of the target Q&A platform (case study of Stack Overflow[3]). The classifier only acts upon the textual data on the question itself (title and body) and does not include features from user profiles (such as previous questions) or any community feedback (replies, upvotes, comments, or edit history). Thus, it can be used as an accurate autonomous moderator or a recommender system for new questions and new users, in addition to any other purpose. The proposed multi-view method utilizes BERT sentence embedding, a software-related named-entity-recognition (SoftNER) model, and general text-mining methods in order to extract the question's subjective features, software-related features, and lexical features, respectively. As a final step, the extracted features of sub-modules will be fed to a state-of-the-art gradient boosting classifier to determine the necessity of an action.

The target has three classes in accordance with the required moderation action: 1) high-quality questions that should be kept as-is, 2) low-quality questions that can be fixed by editing, and 3) low-quality questions that should be closed. While the targets could be fine-grained to include the exact reason for editing or deletion, it is more essential to quickly categorize new posts and notify the author or content moderators as the first step. The process is similar to the manual classification of new content that is currently performed in the Triage Review Queue[4] of some Stack Exchange Q&A websites.

We summarize our contributions as follows:

- We propose a novel approach to predict required high-level moderation action for questions in Q&A websites. The approach does not depend on user profiles or community feedback.
- The proposed multi-view approach utilizes state-of-the-art machine learning models to extract three distinct feature groups that examine a question from three different perspectives.
- We introduce an original dataset for the task of question quality classification. The dataset contains 60 k Stack Overflow questions and is equally separated into three classes of moderation actions.
- We evaluate our approach and compare its performance with that of state-of-the-art models using the cross-validation method. We also discuss the impact of each model component on the overall success of the method.

The structure of this article is as follows: Section 2 reviews past work on the task of text quality classification with a focus on Stack Overflow. Section 3 elaborates on methodology and data collection. Section 4 presents our experimental results, and Section 5 is the concluding remarks.

2. Background

Previous research concentrated on automating a single moderation task in a specific context. Recent examples include fixing tags [5,6], finding high-quality [7,8] and duplicate questions [9,10] in community question-answering websites, preventing the spread of false content across online social networks [11,12], and detecting spam/ad in online encyclopedias

[1] StackExchange.com.
[2] Quora.com.
[3] stackoverflow.com.
[4] https://stackoverflow.com/help/review-triage.

[13,14]. The majority of these cases use a supervised text classification model to separate unacceptable content or behavior from the rest.

To determine the impact of various user-related and textual features on the quality of questions, earlier works mostly relied on surveys and correlation analysis [15,16]. According to [16], the quality of a question affects the quality of answers and the overall community attention to that question. In [17], researchers reached a similar conclusion and stated that more users would be engaged in answers in a short period of time if the question was well-written.

Lezina and Kuznetsov [18] used data from the 2012 Kaggle competition to suggest whether a Stack Overflow question should be closed. They applied Random Forest and Support Vector Machine classifiers to the extracted features, which included the asker's user profile features (such as *age, location, reputation, number of up/down votes, number of previous closed questions,* and *number of answers to previous questions*). One of the most important features was *PostScore*, which represented the number of upvotes for the question. Their best combination of features achieved the lowest multiclass log loss in the competition.

Ravi et al. [1] clarified the distinction between the popularity of a question and the quality of its content. By applying topic models to the extracted n-grams of question body and title, they were able to score 72% accuracy in separating "good" questions from "bad" ones. With the same notion of classifying Stack Overflow questions into two classes of "good" and "bad", Arora et al. [19] considered questions with a negative score as "bad" questions, which do not contribute to the system. Their proposed method that included applying the BM25 model on n-grams reached 73.6% accuracy, a 2.8% improvement. Baltadzhieva and Chrupała [20] used the Ridge regression model and a few statistical methods to determine the importance of various textual and user-related variables. They also investigated words or terms that most frequently exist in high-quality and low-quality questions.

With recent advances in NLP, researchers applied and evaluated state-of-the-art methods in tasks related to finding high quality questions and answers [7,8] and modeling computer code [21,22]. Li et al. [23] utilized a combination of Decision Making Trial and Evaluation Laboratory (DEMATEL) and Analytic Network Process (ANP) to evaluate the effectiveness of 23 indicators of answer quality. Based on their results, subjective features (such as *novelty, effectiveness,* and *understandability*) were among the most important indicators. Sha et al. [24] pursued a similar goal of ranking 15 question-quality related features (including social activities of the questioner) according to their impact on page views. Their results, based on the Grey Relational Grade method, ranked features based on user feedback (such as the count of followers, comments, and answers) at the top of the list. Ho et al. [25] proposed a two-stage hierarchical attention network (HAN) to classify questions into three classes based on their content quality. Their macro average of the evaluation results for the three classes report an F1-score of 37.15%.

More recently, Mousavi et al. [26] used a classification algorithm to evaluate the impact of the first answer on the quality of the subsequent answers in the Yahoo! Answers Health section. The results suggested that 17.13 percent of the best answers were not professional answers.

Tabassum et al. [27] investigated the unique challenges of named entity recognition in the computer programming domain and proposed a software-related named entity recognizer (SoftNER). The proposed method, which is based on an annotated corpus of Stack Overflow, extracts 20 software-related named entity types (such as, CLASS, VARIABLE, XML TAG) and utilizes an attention network to combine the local sentence-level context with corpus-level information extracted from the code snippets.

Sen et al. [28] used the BERT language model to assess the quality of questions and answers pairs from the Microsoft Developer Network (MSDN). They trained on a two-class dataset with 100 k instances, and were able to achieve 57% and 69% F1-scores in two separate settings. They confirm that while the goodness of community-based CQAs is not well defined, it is possible to use artificial intelligence to automate expert-validated rules for quality control.

Our case study, Stack Overflow, is a major and well-known Q&A system, specifically designed for programmers. Since its launch in 2008, more than 60 k users contributed to more than 21 million questions[5], including 524 k monthly comments, 24 k monthly edits, and 20 k monthly closed questions [29]. Engagement in activities, such as voting, commenting, and answering questions, earns user reputation, which is an indicator of the user's value to the website [30].

Finally, the proposed method can be considered a multi-view approach because of its use of distinct groups of features. The multi-view approach has been utilized for various goals in conjunction with recent data collection and feature engineering techniques (such as [31–35]). In this work, we extract all feature groups from the same data source (question title and body), contrary to some works that use separate data sources for every view. In this context, Dalip et al. [36] proposed a multi-view framework for quality assessment of collaborative content that uses user profile and edit history alongside textual content to extract features.

3. Data and methods

This section focuses on the proposed methodology using the seven steps of machine learning laid out in [37]. The third step, which describes the model, will be explained thoroughly in several parts. The experimental results and performance discussions will be reported in Section 4.

[5] https://data.stackexchange.com.

3.1. Data collection

The Stack Exchange family of Q&A websites enables researchers to perform SQL queries on the majority of the database tables to view or download requested information[6]. We used the query tool to provide up-to-date general statistics in the previous sections. We also utilized this method with the goal of creating a novel dataset for the task of question quality classification. The collected dataset consists of 60,000 Stack Overflow questions from 2016 to 2020, classified into three categories:

- **HQ**: High quality questions that require no moderation action. They are considered high quality, as they have not been edited at all, even though they received enough attention from the community (views, answers and upvotes).
- **LQ_EDIT**: Low-quality questions that require community edits. The original question is considered a low-quality question that can be fixed through editing since it remained open regarding having syntactic and semantic problems.
- **LQ_CLOSE**: Low-quality questions that were closed by the community without a single edit. These questions are either incomprehensible, spam, or off-topic for the target community.

We did not include closed duplicate questions in this dataset since detecting them is a completely different task that cannot be performed solely based on the content of a given new question and requires the complete questions dataset. Each question in the dataset has a title, body, tags, creation date, and quality category. The question body is in HTML format and contains all parts of the text (such as URLs and code). The dataset is publicly available on our official GitHub repository.[7]

3.2. Data preparation

The question body, as described in the preceding part, is in HTML format and may contain code samples, URLs, and graphics. As the first step in data preparation, we used an HTML parser to extract question content from the given question body. We also corrected a mishmash of line-break characters. We produce three binary columns to represent each class of the target value (quality categories), as we aim to report results and analyze the complexity of prediction for each class separately. Therefore, in technical terms, we have three binary classifiers rather than one multiclass classifier.

To find important links between simple textual features and the targets, we calculated the Pearson correlation coefficient for all selected features in the training part of the dataset. All coefficients are in the [-0.1 to $+0.1$] range, indicating that there is no meaningful association between the selected features. However, it should be noted that we develop various new features in the next steps, some of which have larger coefficient values.

3.3. Model

As shown in Fig. 1, our proposed model takes the question title and body as input, generates various types of indicators, and then makes a decision based on the engineered features. The question body and text are fed into a model to generate subjective indicators of the question (such as readability and clarity) using BERT sentence embedding [38]. The question body is fed into the SoftNER model to generate domain-specific features based on word entity types, and simple methods are used to extract statistical, lexical, and syntactic indicators. Three statistical formulas are used to transform the extracted features into a larger set of features. We removed elements with insignificant relationships with the targets using correlation analysis. Finally, the remaining features were fed into an XGBoost classifier model.

Our proposed model is made up of three internal models that must be trained separately: (1) a BERT-based regression model that generates subjective features, (2) a NER-based model that predicts the entity type of each word, and (3) a final classifier that makes the final decision on the question. The model has six components (Component 1 to 6). In the following subsections, each component of the proposed model will be explained in accordance with the numbers in Fig. 1.

3.3.1. Extracting subjective indicators

The “clearness,” “intent,” and “interestingness” of a question are among the subjective properties of a question. Even though these properties hold an important key in determining the quality of a question, building a model or even a fair dataset for subjective features of questions is not an easy task.

The Crowdsource team at Google Research, a group focused on NLP and other types of ML science via crowdsourcing (https://crowdsource.google.com/), has collected data on a number of quality aspects of questions and answers from Q&A websites in 2019. The data is relatively small, consisting only of 6079 rows, and all aspects have been represented by numerical values in the range of [0,1]. These subjective features and their short descriptions are available in Table 1.

In a recent work [39], we proposed a regression model to predict these subjective aspects of questions. Our proposed method has two main parts: 1) An input embedding layer (1.A) that extracts contextualized embedding from the BERT-base model for each word in the input sentence. 2) A neural network model that predicts the targets using the word repre-

[6] One way is to use https://data.stackexchange.com.

[7] https://github.com/Moradnejad/StackOverflow-Questions-Quality-Dataset/.

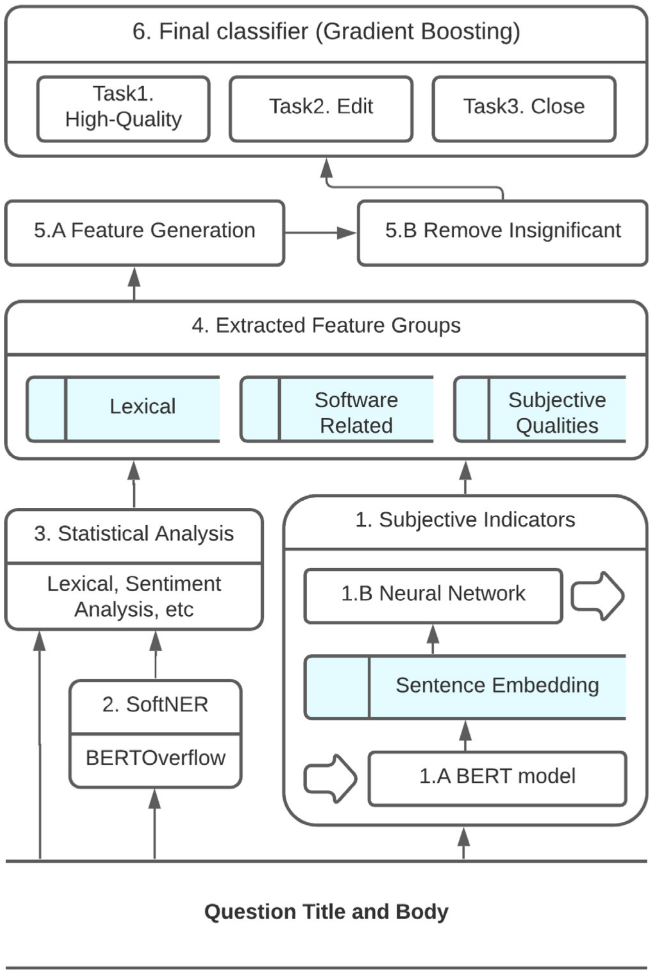

Fig. 1. Architecture of the proposed model.

Table 1
Name and description of subjective features.

	Name	Description
1	Clear intent	Is the questioner's intent clear?
2	Conversational	Is the question conversational?
3	Expects short answer	Does the question expect a short answer?
4	Fact seeking	Is the question asking for facts?
5	Has accepted answer	Does the question have a commonly accepted answer?
6	Interestingness others	Does the question look interesting to others?
7	Interestingness self	Does the question look interesting to the asker?
8	Multi intent	Is the question intended to have multiple meanings (multiple questions within one)?
9	Not a question	Is the text informative, without having a clear question?
10	Opinion seeking	Does the question ask for opinion-based answers?
11	Type-choice	Is the question a multi-choice question?
12	Type-compare	Is the question asking for a comparison between alternatives?
13	Type-consequence	Is the question looking for the consequence of a possible action?
14	Type-definition	Is the question looking to define something?
15	Type-entity	Is the question related to an entity?
16	Type-instructions	Is the question looking for instructions?
17	Type-procedure	Is the question require explaining a procedure?
18	Type-reason explanation	Is the question looking for an explanation of the reason?
19	Type-spelling	Is the question about spelling?
20	Well written	Is the question well written?

sentations from the previous layer. We used this model to predict the values of subjective features for all 60 k questions of this research.

3.3.2. SoftNER model

Named Entity Recognition and Classification (NERC) aims at identifying references to entities in text and is one of the important sub-tasks of Information Extraction (IE). In general domains of text, it looks for names of people, organizations, locations, time, date, currency and other numerical expressions [40].

The extraction of software-related named entities (SoftNER) can contribute to a wide range of downstream applications, such as finding similar questions [41], and imposes significant challenges as it requires resolving a large number of unseen tokens, inherent polysemy, and salient reliance on context [27]. Because the goal of this study is to determine the quality of Stack Overflow questions, which may contain code-samples and programming-related terms, we decided that the level of involvement of software-related entities is critical in determining the quality of a question.

In this part, we utilized a recent SoftNER model [27] to extract features related to programming and software engineering. BERTOverflow has three main elements and uses an attention network to combine the local sentence-level context with corpus-level information extracted from the code snippets. We feed the tokenized body of questions as input, and the model detects tokens that represent software-related entities. If a token is not determined as a reference to an entity, it will be replaced by ‘O’, otherwise the output will show the type of software entity (such as VARIABLE, DATABASE, or ALGORITHM).

3.3.3. Statistical analysis

In this part, we extract some numerical features from the existing data. First, we extract simple lexical indicators from the question title and body, which contain information on the involved letters, words, and sentences. The main features of this component are *letter* and *word count, code percentage of the body, sentiment analysis*, and *the type of title as a sentence (interrogative or not)*. We also attached great importance to a few lexical features in the first and last sentences of the body, as well as the average value for all sentences. *Word count, type of sentence,* and *sentiment analysis* are among the features.

Furthermore, we use the output of the SoftNER model to extract software-related features. These features include a count and percentage of tokens, separated by the types of software entities. Similar to the lexical features, we extract these features regarding the full title and body of the question, the first sentence, the last sentence, and the average values for all sentences.

3.3.4. Extracted feature groups

In this part, we will look at the extracted feature groups in more detail. In Component 1, we predicted 20 subjective features for the entire question’s body. Despite the fact that there was no direct feature extraction in Component 2, we used the output of this part to extract 20 software-related features in Component 3. Using general text-related lexical and statistical methods, we extracted 54 additional attributes in addition to these features in Component 3.

We categorize these features based on their perspective on the input, which includes *subjective question features, software-related features,* and *general statistical-lexical features*. Subjective characteristics are associated with questions in general, and its trained model is applicable to any question-related task on any subject. Software-related features, on the other hand, are domain-specific and can be used in other context-related prediction tasks, whether on questions or normal texts. General statistical-lexical features treat the input as text and can be applied to any text-related task, whether related to questions and software or not.

3.3.5. Feature engineering

In this part, a few minor steps are taken to positively influence the performance of the final classifier.

First, with the overall goal of generating more useful features in mind, we use three numerical functions to transform the extracted ones from the preceding part. The three transform functions are as follows: 1) log, 2) *sqr*, and 3) *sqrt*. As a result, the number of features increased by threefold. Further to that, because some classifiers, such as kNN-based models, are sensitive to input values, we normalized all of the dataset’s features to values ranging from 0 to 1. Finally, we included TF-IDF vectors as a simple additional feature extraction method because they connect a given question to previously existing questions on the platform and may include valuable information that was missing in our extracted data.

We ran a correlation analysis on all of the features (both extracted and generated) with the target values. This step is used to identify and remove features that have a negative or insignificant impact on the final output. It is important to note that this analysis is done on the train portion of the dataset to prevent test data leakage. We removed five features based on the results of this step that had less than a 0.01 impact on the three targets.

3.3.6. Final classifier

The proposed model’s final component is to classify a given question based on the extracted features of the previous parts. We chose the XGBoost classifier [42] for this critical role and tuned its hyper-parameters accordingly. XGBoost is the latest step in the evolution of tree-based algorithms that include decision trees, boosting, random forests, boosting, and gradient boosting [43]. It is an optimized distributed gradient boosting method that provides fast and accurate results and achieves state-of-the-art accuracy especially if we are keen on run time [42,44]. Other optimized gradient boosting implementations, such as Microsoft’s LightGBM, are comparable in performance.

We approach the multiclass task at hand as three distinct binary classifications, which allows us to better answer which type of quality is easier or more difficult to determine. In practice, however, converting the task into a multiclass classification would increase the overall prediction accuracy. The subtasks are as follows:

Task 1. Predicting if the question is of high quality (requires no action).
Task 2. Predicting if the question requires editing.
Task 3. Predicting if the question should be closed.

3.4. Model training

As thoroughly discussed in the preceding subsection, our proposed method includes three supervised models that require training. Component 1 predicts subjective question features for all parts of the new dataset. To train the neural network model, we use the 2019 dataset provided by Google's Crowdsource team, which includes 6 k hand-labeled questions with multiple subjective features. After training for two epochs, the neural network's accuracy did not improve.

Component 2's goal is to predict software entities based on tokens from a given question. We trained on the BERTOverflow NER corpus for this part. Tabassum et al. [27] randomly selected 1237 Stack Overflow question threads and 6501 sentences from GitHub readme files. Four annotators manually tokenized and annotated the text with 20 different types of entities. For the Stack Overflow questions, at most four answers were included, such as the accepted answer and the most upvoted answer. For the final classifier, all selected features were extracted and concatenated for the new dataset's rows.

3.5. Model evaluation

This step delves into the evaluation metrics which will be used to assess the model's performance. Because the final classifier's accuracy is dependent on the accuracy of the two sub-models, we first report on their accuracy.

Component 1 is a regression model that predicts twenty subjective qualities of questions. We chose Mean-Squared-Error (MSE) to evaluate this component. The MSE metric is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \widehat{Y}_i\right)^2 \tag{1}$$

where N is the number of predictions, Y_i is the observed value of target for i_{th} prediction, and $\widehat{Y}_i$ is the predicted value of the model for i_{th} prediction. Previously, we trained this part on a separate dataset and achieved a MSE of 0.046 after 2 epochs of training. The low error rate for this component in predicting subjective features contributes to the good performance of the overall model.

We report the evaluation results of the original paper for the SoftNER component (Component 2) because we relied on the trained model of the original authors. The model reportedly achieved 78.4% *precision* and 79.79% *recall*.

For the overall performance of the proposed model in the final classifier, we only introduce the metrics and techniques here, and the results will be reported in Section 4. We use the cross-validation technique to properly evaluate the overall performance of the proposed model. Cross-validation divides a dataset into multiple equal parts and evaluates it by using each partition as the validation set and the remaining parts for training. The reported results are averages of calculated metrics from multiple runs. As a result, while this procedure takes longer, the outcome is a more accurate estimate of model prediction performance.

As discussed earlier, we converted the multiclass target into three binary classifications. Because the distribution of classes is close, we evaluate and report our results using *Precision, Recall,* and *F1 score*. These three metrics are commonly used in classification tasks and incorporate the number of false predictions into their calculations.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

Where TP, FP, FN represent true positives, false positives and false negatives, respectively.

3.6. Parameter tuning

It is critical to find near-optimal hyper-parameters in order to fine-tune the models and reduce the evaluation log loss. This was a delicate task, as we needed to avoid over-fitting on the training data, which causes low-quality results on the test set. For the BERT-based model that extracts subjective qualities of questions, we set 100 as the maximum sequence length for BERT tokens. In addition, we chose dropout of 0.2 and sigmoid activation function in neural network configurations. For the final XGBoost model, we set *n_estimators = 1000, learning_rate = 0.07, max_depth = 6* and *colsample_tree = 1*.

3.7. Make predictions

Based on three trained components of our model, we predicted the subjective qualities of all questions using Component 1, SoftNER entities of all questions using Component 2, and required moderation actions for the test part of the dataset (15 K questions) using Component 6. Results and evaluation of our predictions will be discussed in Section 4.

4. Results

This section compares our proposed model to a few baseline models that could potentially replace the entire task or the final classifier component. First, we briefly introduce these selected baselines, and then we will report the performance evaluation results. We will discuss the importance of feature groups in the overall success of the model.

4.1. Baselines

The selected baselines for the purposes of evaluation are as follows:

1 XLNet [45] is a generalized pretraining method for language modelling that tries to reduce the problems of previous autoregressive language models and the BERT bidirectional model. For the task of text classification (and some other NLP tasks), XLNet outperforms BERT on several benchmark datasets. Unlike our next baselines, XLNet can take a string input and apply its own embedding to create numerical representations and perform the classification.
2 Convolutional Neural Network (CNN) is a Deep Learning algorithm that uses convolution in place of general matrix multiplication in at least one of its layers. Its architecture includes ten dense layers, two dropout layers, and one flattener with *learning_rate* = 1e-4, *Adam* optimizer, and *relu* for activation function. The input for this baseline, as well as the next ones, are word vectors of the cleaned strings, which perform the best with these models.
3 Random Forest (RF) is a tree-based algorithm that uses a specified number of trees for tasks of classification or regression. We trained a random forest model by using 500 decision trees and a random selection of features at each node creation. The input is converted using the TFIDF vectorizer.
4 K-Nearest Neighbor (KNN) uses Euclidean distance to assign class labels based on the classes of K nearest points of training data in a feature space. For the evaluation purposes, we used $K = 5$.
5 Logistic Regression (LR) is a traditional baseline for the tasks of multiclass categorization. It uses the *sigmoid* function on the output of the Linear Regression model in order to gain a discrete result. The sigmoid function is defined as [46]:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

4.2. Results

Table 2 represents the evaluation results for the proposed model and the baseline models. The table is separated into three parts to better reflect the accuracy of the prediction for each class (thus three binary targets). As discussed earlier, the results are based on the cross-validation approach. Multiple rounds of cross-validation are performed using different partitions (K = 4), and the validation results are averaged over the rounds to give an estimate of the model's predictive performance.

Our proposed model outperforms the state-of-the-art XLNet model in all three tasks in terms of F1-score. It is important to note that the overall performance of any binary classification model is best evaluated by the F1-score as it includes the results of Precision and Recall (thus FN and FP) in its calculations. In general, while CNN and Random Forest algorithms performed admirably according to the Precision criteria, the proposed model surpasses them in Recall and F1 scores. On all three tasks, LR demonstrated a promising stable performance, whereas the KNN algorithm performed very poorly in comparison to other tree-based algorithms. Furthermore, the proposed model achieves 95.6% accuracy if it considers the problem as a multiclass task instead of three binary tasks. This higher accuracy is the result of predicting exactly one class for each case, rather than independent binary predictions.

According to the overall results, predicting the requirements of a question for editing (Task 2) was the easiest task, and predicting closure (Task 3) was the most difficult. In the latter, we can see that Recall scores are relatively low for all selected baselines. In terms of time performance, all experiments were carried out on a computer equipped with 16 GB of RAM and an Intel (R) Xeon (R) 2.00 GHz CPU, using the Python programming language. The Logistic Regression classifier trains and predicts much faster than the alternatives (less than a minute), our proposed model takes less than ten minutes to complete, and the XLNet model, with over 360 million parameters, takes close to an hour for each task.

Table 2
Performance comparison of the proposed model with the baselines.

Task 1: Predicting if the question requires no action (HQ)				
Model	Precision	Recall	F1-Score	Time (s)
XLNet	65.6	81.7	72.7	3040
CNN	79.3	52.1	62.8	1354
RF	87.2	56.0	68.2	951
KNN	60.5	65.4	62.8	384
LR	73.8	71.6	72.6	39
Proposed	**93.1**	**90.4**	**91.7**	**1514**
Task 2: Predicting if the question should be edited				
Model	Precision	Recall	F1-Score	Time (s)
XLNet	81.2	85.0	83.0	3430
CNN	77.6	73.3	75.3	871
RF	93.5	51.6	66.5	295
KNN	80.1	73.2	76.4	331
LR	83.8	74.6	78.9	35
Proposed	**94.7**	**90.7**	**92.6**	**1495**
Task 3: Predicting if the question should be closed				
Model	Precision	Recall	F1-Score	Time (s)
XLNet	62.3	58.9	60.5	3102
CNN	71.3	49.2	58.2	871
RF	72.5	44.3	54.9	248
KNN	58.3	36.7	45.0	317
LR	69.7	58.3	63.4	44
Proposed	**85.8**	**79.2**	**82.3**	**1532**

Table 3
Performance evaluation of the proposed method by excluding subjective features.

Task	Precision	Recall	F1-Score
Task1	84.6	72.2	77.9
Task2	87.9	85.0	84.9
Task3	75.7	59.0	66.3

Table 4
Performance evaluation of the proposed method by excluding software-related features.

	Precision	Recall	F1-Score
Task1	81.7	73.8	77.5
Task2	84.7	85.5	85.2
Task3	78.0	69.9	70.6

4.2.1. Impact of subjective qualities

To signify the impact of features related to the subjective qualities of questions, we performed the evaluations of the proposed model by disabling the Component 1 of our model and excluded these features from the training and prediction. Based on the results of this evaluation (Table 3), the final classifier works poorly by ~7.7–16.0% in *F1-score*, if it does not include subjective features. The gap is wider in predicting high-quality posts and posts that should be closed.

4.2.2. Impact of Software-related features

Similar to the previous subsection, to better understand the overall impact of using software-related features, we performed the evaluations by omitting them from the training and predictions. Table 4 displays the result of this scenario, which indicates an F1-score decrease of 7.4–14.2% for the three tasks. This gap is wider for predicting high-quality posts and posts that should be closed.

5. Conclusion

Popular Q&A websites depend on user feedback for content moderation, which results in slow and subjective handling of violations. Previous research on automating moderation actions mostly relies on community feedback and user history to achieve higher accuracies, which makes their proposed methods less usable for real-world scenarios. Those approaches neglect new questions and new users, who are more in need of moderation actions or recommendations. To respond to these major issues, we borrowed from machine learning and text mining literature to construct a classifier to automatically detect required moderation actions only based on features extracted from the question content. Our proposed method trains a state-of-the-art gradient boosting classifier on the outputs of three components of feature extraction. These three components are proposed to extract text-related, question-related, and context-related features using the latest ML advances. For evaluation purposes, we published a new benchmark dataset consisting of 60 k Stack Overflow questions with quality ratings. Our overall approach achieved 95.6% accuracy on the novel dataset and outperforms all state-of-the-art methods.

Our results clearly show the high influence of context-related features on the accuracy of the classifier, which makes them essential in creating an accurate recommender or moderation system. Removing these features decreased the F1-score by 11.1% on average in our experiments. These features should be selected based on the values, norms, question types, content patterns, user behavior, and rules of the target system. For example, providing examples or using technical abbreviations can be common, encouraged, or prohibited based on the target Q&A system.

In addition, question-related features, such as the type of the question (opinion-based, factual, or multi-purpose), contain valuable information in predicting the overall quality of a question. Some of these subjective features deal with the quality of the question (e.g. being well-written, asking a question with "clear intent", and not having multiple intents in one question) and can be considered as consistent rules for asking questions on all Q&A websites. On the other hand, the impact of some features (e.g. opinion-based answers or using external links) can be related to specific rules of the target system and thus emphasizes our previous note on attention to context.

Future work should consider the high volatility of subjects in Q&A websites and the need for fast adaptability of proposed systems with less need for new training. For example, a moderation system for a health-related Q&A website should be able to adapt quickly and without much training to deal with countless new questions about a new disease. Just as we showcased in this work, Transfer Learning should be considered as a great solution for fast adaptation of trained prediction models on Q&A websites. This approach allows improving or adapting existing trained models to a new situation, with the least amount of training.

While the proposed approach can be taken as-is to develop a recommender or moderation system for questions on Q&A websites, our model structure and findings with regard to fast adaptability and the importance of feature-extraction components would assist similar computer-aided tools for online communities.

CRediT authorship contribution statement

Issa Annamoradnejad: Conceptualization, Data curation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Jafar Habibi:** Validation, Project administration, Supervision, Writing – review & editing. **Mohammadamin Fazli:** Supervision, Formal analysis.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We would like to thank Jeniya Tabassum, the author of BERTOverflow, who helped us with the results of the SoftNER model.

References

[1] S. Ravi, B. Pang, V. Rastogi, R. Kumar, Great question! Question quality in community QandA, in: Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014, 2014, pp. 426–435.
[2] J. Ahn, B.S. Butler, C. Weng, S. Webster, Learning to be a better q'er in social Q&A sites: Social norms and information artifacts, Proc. Am. Soc. Inf. Sci. Technol. 50 (2013) 1–10.
[3] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, B. Hartmann, Design lessons from the fastest q&a site in the west, in: CHI '11 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2011, pp. 2857–2866, https://doi.org/10.1145/1978942.1979366.
[4] I. Annamoradnejad. Requirements for Automating Moderation in Community Question-Answering Websites, in: 2022: pp. 1–4. https://doi.org/10.1145/3511430.3511458.
[5] P. Singh, R. Chopra, O. Sharma, R. Singla, Stackoverflow tag prediction using tag associations and code analysis, J. Discr. Math. Sci. Cryptogr. 23 (1) (2020) 35–43.
[6] N. Khezrian, J. Habibi, I. Annamoradnejad, Tag Recommendation for Online Q&A Communities based on BERT Pre-Training Technique, ArXiv Preprint ArXiv:2010.04971 (2020).
[7] Y. Yao, H. Tong, T. Xie, L. Akoglu, F. Xu, J. Lu, Detecting high-quality posts in community question answering sites, Inf. Sci. 302 (2015) 70–82.
[8] H. Toba, Z.-Y. Ming, M. Adriani, T.-S. Chua, Discovering high quality answers in community question answering archives using a hierarchy of classifiers, Inf. Sci. 261 (2014) 101–115.
[9] L. Wang, L. Zhang, J. Jiang, Duplicate question detection with deep learning in stack overflow, IEEE Access 8 (2020) 25964–25975.
[10] Z. Imtiaz, M. Umer, M. Ahmad, S. Ullah, G.S. Choi, A. Mehmood, Duplicate questions pair detection using siamese malstm, IEEE Access 8 (2020) 21932–21942.

[11] A. Campan, A. Cuzzocrea, T.M. Truta, Fighting fake news spread in online social networks: Actual trends and future research directions, IEEE Int. Conf. Big Data (Big Data) 2017 (2017) 4453–4457.
[12] G. Shrivastava, P. Kumar, R.P. Ojha, P.K. Srivastava, S. Mohan, G. Srivastava, Defensive modeling of fake news through online social networks, IEEE Trans. Comput. Social Syst. 7 (5) (2020) 1159–1167.
[13] T. Green, F. Spezzano, Spam users identification in wikipedia via editing behavior, Proceedings of the International AAAI Conference on Web and Social Media, 2017.
[14] S. Yuan, P. Zheng, X. Wu, Y. Xiang, Wikipedia vandal early detection: from user behavior to user embedding, Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2017) 832–846.
[15] F.M. Harper, D. Raban, S. Rafaeli, J.A. Konstan, Predictors of answer quality in online Q&A sites, Conference on Human Factors in Computing Systems - Proceedings (2008) 865–874, https://doi.org/10.1145/1357054.1357191.
[16] E. Agichtein, C. Castillo, D. Donato, A. Gionis, G. Mishne, Finding high-quality content in social media, in: Proceedings of the 2008 International Conference on Web Search and Data Mining, 2008, pp. 183–194.
[17] B. Li, T. Jin, M.R. Lyu, I. King, B. Mak, Analyzing and predicting question quality in community question answering services, in: Proceedings of the 21st International Conference on World Wide Web, 2012, pp. 775–782.
[18] G.E. Lezina, A.M. Kuznetsov, Predict closed questions on StackOverflow, CEUR Workshop Proceedings 1031 (2013) 10–14.
[19] . Arora, D. Ganguly, G.J.F. Jones, The good, the bad and their kins: Identifying questions with negative scores in StackOverflow, Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015. (2015) 1232–1239. https://doi.org/10.1145/2808797.2809318.
[20] A. Baltadzhieva, G. Chrupala, Predicting the quality of questions on stackoverflow, International Conference Recent Advances in Natural Language Processing, RANLP. 2015-Janua (2015) 32–40.
[21] R. Cai, Z. Liang, B. Xu, Z. Li, Y. Hao, Y. Chen, TAG: Type Auxiliary Guiding for Code Comment Generation, ArXiv Preprint ArXiv:2005.02835. (2020).
[22] M.R. Tavakoli, A. Heydarnoori, M. Ghafari, Improving the quality of code snippets in stack Overflow, Proceedings of the ACM Symposium on Applied Computing. 04-08-Apri (2016) 1492–1497. https://doi.org/10.1145/2851613.2851789.
[23] M. Li, Y. Li, Y. Lu, Y. Zhang, Evaluating indicators of answer quality in social Q&A websites, Proceedings of the 23rd Pacific Asia Conference on Information Systems: Secure ICT Platform for the 4th Industrial Revolution 2019, 2019.
[24] A.S. Sha, Y. Shi, A. Haller, How question quality drives Web performance in community question answering sites, ArXiv Preprint ArXiv:2012.06263 (2020). http://arxiv.org/abs/2012.06263.
[25] M.K. Ho, S. Tatinati, A.W.H. Khong, Distilling essence of a question: a hierarchical architecture for question quality in community question answering sites, in: Proceedings of the International Joint Conference on Neural Networks, 2020, https://doi.org/10.1109/IJCNN48605.2020.9206616.
[26] R. Mousavi, T.S. Raghu, K. Frey, Harnessing artificial intelligence to improve the quality of answers in online question-answering health forums, J. Manage Inf. Syst. 37 (2020) 1073–1098, https://doi.org/10.1080/07421222.2020.1831775.
[27] J. Tabassum, M. Maddela, W. Xu, A. Ritter, Code and Named Entity Recognition in StackOverflow, in: ACM Annual Meeting of the Association for Computational Linguistics (ACL), 2020, pp. 4913–4926.
[28] B. Sen, N. Gopal, X. Xue, Support-BERT: predicting quality of question-answer pairs in MSDN using deep bidirectional transformer, ArXiv Preprint ArXiv:2005.08294 (2020). http://arxiv.org/abs/2005.08294.
[29] Moradnejad, Edits and Comments Count by Month. StackExchange, StackExchange. (2020). https://data.stackexchange.com/stackoverflow/query/1347461/edits-comments-of-past-months (accessed June 21, 2021).
[30] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, C. Faloutsos. Analysis of the reputation system and user contributions on a question answering website: StackOverflow, Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2013. (2013) 886–893. https://doi.org/10.1145/2492517.2500242.
[31] Q. Lin, M. Men, L. Yang, P. Zhong, A supervised multi-view feature selection method based on locally sparse regularization and block computing, Inf. Sci. 582 (2022) 146–166.
[32] Z. Lyu, M. Yang, H. Li, Multi-view group representation learning for location-aware group recommendation, Inf. Sci. 580 (2021) 495–509.
[33] X. Wu, H. Hui, M. Niu, L. Li, L. Wang, B. He, L. Yang, H. Li, J Tian Li, et al, Deep learning-based multi-view fusion model for screening 2019 novel coronavirus pneumonia: A multicentre study, Eur. J. Radiol. 128 (2020).
[34] C. Yan, B. Gong, Y. Wei, Y. Gao, Deep multi-view enhancement hashing for image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 43 (4) (2021) 1445–1451.
[35] M. Guo, Y. Li, Y. Su, T. Lambert, D.D. Nogare, M.W. Moyle, L.H. Duncan, R. Ikegami, A. Santella, I. Rey-Suarez, D. Green, A. Beiriger, J. Chen, H. Vishwasrao, S. Ganesan, V. Prince, J.C. Waters, C.M. Annunziata, M. Hafner, W.A. Mohler, A.B. Chitnis, A. Upadhyaya, T.B. Usdin, Z. Bao, D. Colón-Ramos, P. La Riviere, H. Liu, Y. Wu, H. Shroff, Rapid image deconvolution and multiview fusion for optical microscopy, Nat. Biotechnol. 38 (11) (2020) 1337–1346.
[36] D.H. Dalip, M.A. Gonçalves, M. Cristo, P. Calado, A general multiview framework for assessing the quality of collaboratively created content on web 2.0, J Assn Inf Sci Tec 68 (2) (2017) 286–308.
[37] Y. Gue, The 7 Steps of Machine Learning. Towards Data Science, Towards Data Science. (2017). https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e (accessed October 21, 2021).
[38] J. Devlin, M.W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, 2019.
[39] I. Annamoradnejad, M. Fazli, J. Habibi, Predicting Subjective Features from Questions on QA Websites using BERT, in: 2020. https://doi.org/10.1109/icwr49608.2020.9122318.
[40] D. Nadeau, S. Sekine, A survey of named entity recognition and classification, Lingvisticae Investigationes 30 (2007) 3–26.
[41] A. Shirani, B. Xu, D. Lo, T. Solorio, A. Alipour, Question relatedness on stack overflow: the task, dataset, and corpus-inspired models, ArXiv Preprint ArXiv:1905.01966 (2019).
[42] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[43] V. Mordar, XGBoost Algorithm: Long May She Reign!. Towards Data Science, Towards Data Science. (2019). https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d (accessed December 20, 2020).
[44] S.S. Dhaliwal, A.-A. Nahid, R. Abbas, Effective intrusion detection system using XGBoost, Information 9 (2018) 149.
[45] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R.R. Salakhutdinov, Q.V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, Adv. Neural Inf. Process. Syst. 32 (2019).
[46] M.T. Tommiska, Efficient digital implementation of the sigmoid function for reprogrammable logic, IEE Proceed.-Comput. Digital Tech. 150 (2003) 403–411.