Ask It Right! Identifying Low-Quality questions on Community Question Answering Services

Udit Arora†
IIIT-Delhi
New Delhi, India
udit18417@iiitd.ac.in

Nidhi Goyal†
IIIT-Delhi
New Delhi, India
nidhig@iiitd.ac.in

Anmol Goel
IIIT-Hyderabad
Hyderabad, India
anmol.goel@research.iiit.ac.in

Niharika Sachdeva
InfoEdge India Limited
Noida, India
niharika.sachdeva@infoedge.com

Ponnurangam Kumaraguru
IIIT-Hyderabad
Hyderabad, India
pk.guru@iiit.ac.in

***Abstract*—Stack Overflow is a Community Question Answering service that attracts millions of users to seek answers to their questions. Maintaining high-quality content is necessary for relevant question retrieval, question recommendation, and enhancing the user experience. Manually removing low-quality content from the platform is time-consuming and challenging for site moderators. Thus, it is imperative to assess the content quality by automatically detecting and 'closing' the low-quality questions. Previous works have explored lexical, community-based, vote-based, and style-based features to detect low-quality questions. These approaches are limited to writing styles, textual, and handcrafted features. However, these features fall short in understanding semantic features and capturing the implicit relationships between tags and questions. In contrast, we propose LQuaD (Low-Quality Question Detection), a multi-tier hybrid framework that, a) incorporates semantic information of questions associated with each post using transformers, b) includes the question and tag information that enables learning via a graph convolutional network. LQuaD outperforms the state-of-the-art methods by a 21% higher F1-score on the dataset of 2.8 million questions. Furthermore, we apply survival analysis which acts as a proactive intervention to reduce the number of questions closed by informing users to take appropriate action. We find that the timeframe between the stages from the question's creation till it gets 'closed' varies significantly for tags and different 'closing' reasons for these questions.**

***Index Terms*—Stack Overflow, Community Question Answering, Low-quality questions**

I. Introduction

Stack Overflow (SO) is a popular Community Question Answering (CQA) service with 22.1 million questions and 16.6 million users as of 2022.[1] Since SO is an open-access website used by novice and experts, it is essential to maintain the quality of questions posted over the platform [1]. To this end, SO issues guidelines and employs a reward-based voting mechanism to incentivize users to ask good quality questions [2]. Questions not following the guidelines are 'closed' via a community-based voting system. 'Closed' questions can not receive answers, but the user can improve them for reopening. Fig. 1 shows 'closed' questions from SO. The primary reasons

Title	Which strategy .... project?	Title	Convert a list to dict .... key value
Body	You work on an important project that contains 7 independent .... choose?	Body	What I have tried so far is: self.dict_total_words1 = {i.split(': ')[0]: int(i.split(': ')[1]) for i ....
Tags	time-management	Tags	python regex java
Decision	**Closed:** off-topic	Decision	**Closed:** unclear what you're asking
Title	Why do people hate java?	Title	How do I code a forum in PHP?
Body	In the world of python programming	Body	Hi, I'm a beginner. I've been tasked with coding a web forum in PHP ....
Tags	java python	Tags	php sql
Decision	**Closed:** primarily opinion-based	Decision	**Closed:** too broad

Fig. 1. A sample of 'closed' questions from Stack Overflow. These are 'closed' due to different reasons such as '*off-topic*', '*unclear what you're asking*', '*too broad*' and '*primarily opinion-based*'.

for 'closing' questions from quality perspective are *'off-topic'*, *'unclear what you're asking'*, *'primarily opinion-based'*, and *'too broad'*. There are approximately 5900+ new questions posted daily over the SO platform.[2] As a result of high traffic, it is inefficient and difficult to review every question manually. These questions also increase the workload of moderators and experienced users with distinguished privileges who spend time 'closing' questions. Therefore, it is imperative to identify and close low-quality questions automatically in order to minimize workload. Previous works focus on predicting the quality of questions on CQA websites [1]–[4]. They rely on textual styles like *title length*, *body length*, handcrafted features (*number of URLs*), and supervised learning (e.g., Random forest, SGBT) for binary classification tasks. Literature also explores the question's quality using Convolutional Neural Network [5], Hierarchical Attention Network [6], and Gated Recurrent Units [7], [8].

However, these approaches fall short in: a) effectively capturing the semantic and structural information of the content, b)

[1]https://stackoverflow.com/questions

[2]https://stackexchange.com/sites?view=list#traffic

detecting low-quality questions at the time of creation (*cold-start problem*) when features such as votes, question scores, answers, and comments are unavailable, c) estimating the survival probability of a question till it gets 'closed.'

To address these limitations, we utilize the semantic information present in the question's content using a transformer-based model, i.e., BERT [9] to improve the modeling of questions. Additionally, we utilize tag-specific information, which plays a crucial role in determining the question's visibility and answering the questions. Moreover, including relevant tags is one of the guidelines for asking a good quality question. Recent advances in graph-based deep learning [10] have led to the rise of graph neural networks (GNNs) that can model the structural relationships well. Therefore, we introduce a novel way to exploit the tag-related information to capture implicit relationships between a question and tag using Graph Convolutional Networks (GCNs) [11]. Towards this end, we propose a multi-tier hybrid framework, Low-Quality Question Detection (LQuaD), that incorporates the content-related information associated with each question using BERT. Using GCNs over a graph of 2.9M nodes and 8.4M edges, LQuaD can collaboratively identify patterns between questions and tags in a transductive manner. Furthermore, we conduct survival analysis [12] to provide various insights on questions to inform users about a potential timeframe for the question's closure. We use Kaplan-Meier Estimator [13] for survival analysis to evaluate the temporal event of low-quality questions being 'closed'. Our framework addresses the cold-start problem using only pre-submission information of questions posted over the platform. It evaluates the closure of questions and timeframe for closing based on exact reasons and tags. Our major contributions are:

- We propose a novel framework, LQuaD, which establishes the utility of a question-tag graph and transformers to detect low-quality questions that are likely to get 'closed' at the time of posting. Our framework acts as an early-assessment tool to assist users in composing a question, which would remain open and receive responses.
- We also examine the impact of non-content related characteristics of the question using survival analysis to estimate the time duration of closure of the question.
- We evaluate LQuaD on 17.7M questions and make the code publicly available.[3]

The organization of the rest of the paper is as follows: Section II contains related work and Section III elaborates our methodology in detail. Section IV describes the experiments; Section V discusses results and analysis; Section VI reports ablation study , Section VII shows error analysis followed by conclusion and future work in Section VIII.

II. Related Work

In this section, we provide an overview of related literature on detection of low-quality questions, graph-based approaches, and survival analysis.

[3]https://anonymous.4open.science/r/submission2022-41E0

a) Low-quality Questions Detection: Previous works [2], [14] studied the handcrafted features of posts, community, and users for detection of low-quality questions. Baltadzhieva et al. [1] studied and analyzed the linguistic terms contained in the post's content to predict the question's quality. However, it defines question's quality in abstract terms making it prone to subjectivity and may result in somewhat arbitrary assessments. Ahmed et al. [15] studied user's behavior in guiding the feature engineering process to determine the question's quality. Jan et al. [16] on identifying unclear questions as they have low-quality. Deepak et al. [17] estimate the relative difficulty of questions. Previous works [18], [19] demonstrate that duplicate questions constitute 60% of the 'closed' questions. Liang et al. [20] found that user reputation is a good indicator for question quality. Roy et al. [5] contributes with a multi-class classification of 'closed' questions along with the reasons why questions get 'closed'. Most of these methods [1], [2], predict question quality by utilizing handcrafted features like, community-based features, part-of-speech tagging, user profile features, textual features (*body length*, *no. of URLs, count of words*) and writing styles. These features capture limited information related to the context and semantics of the question's content which is a crucial step to exploit the maximum efficiency of the natural language processing [21]. Hence, we combine contextual and semantic properties into our proposed framework to improve the identification of low-quality questions, i.e., 'closed' questions.

b) Graph Neural Networks: Graph Neural Networks (GNNs) have found their applications in recommender systems [22], machine translation [11], role labeling [23], image classification [24] and many more real-world applications. Recent advancements in GNNs [10], [25], [26] allow using node neighborhoods to learn discriminative features collaboratively. Yao et al. [26] utilize joint document-words graph for text classification via node classification task. However, GNNs are unexplored in capturing the information using a heterogeneous question-tag graph. GNNs are well-equipped to capture information flow between the questions and their associated tags for node classification tasks.

c) Survival Analysis: Survival analysis provides statistical methods to study the time-to-event data [27]. It assists in determining the longevity of responses on social media platforms [28] and in assessing the wait time of the callers [29]. The technique incorporates valuable information about unresolved questions in the prior studies [30]. A previous work [31] uses survival analysis to estimate the time till first response and time till accepted answer of questions on Stack Overflow. In this work, we extend the study and carry out the temporal analysis using the Kaplan-Meier estimate [13] to provide a time estimate in which is question is 'closed'.
Our research is uniquely inclined towards using graph convolutional networks and transformers in the cold-start scenario for low-quality question detection. We also provide various insights on the timeframe to 'close' a low-quality question across different tags and 'closing' reasons.

III. Methodology

In this section, we discuss the problem definition and framework, which consists of two parts a) Low Quality Questions Detection b) Temporal Event Analysis.

A. Preliminary & Problem Definition

Consider the set of questions $\mathcal{Q} = \{q_1, q_2, \cdots\cdots, q_i\}$, a question $q_i \in \mathcal{Q}$ is a tuple $(c_i, \mathcal{T}_i, y_i)$ where c_i, $\mathcal{T}_i$, y_i represents the content, tag set, and label of the i^{th} question, respectively. The content is represented by the concatenation of *title* and *body* of the question. The tag set is represented as, $\mathcal{T}_i = \{t_1^i, t_2^i, t_3^i ..... t_k^i\}$ $\forall$ $1 \leq k \leq 6$ and t_k^i represents the k^{th} tag of the i^{th} question. Also, $y_i \in \{0, 1\}$, where y_i = 0 corresponds to the 'closed' class i.e., the question is unfit for the SO platform, and y_i= 1 corresponds to the non-'closed' class. Then, we construct an undirected question-tag graph $\mathbb{G}$ = ($\mathbb{V}$, $\mathbb{E}$), where $\mathbb{V}$ and $\mathbb{E}$ are the set of nodes and edges respectively. Here $\mathbb{V}$ consists of questions belonging to $\mathcal{Q}$ and tagset $\mathbb{T} = \{\mathcal{T}_1 \cup \mathcal{T}_2 \cup ... \cup \mathcal{T}_{|Q|}\}$. We construct an edge $e \in \mathbb{E}$ between q_i and t_k^i where $\mathbb{E} \subset \mathcal{Q} \times \mathbb{T}$. Additionally, for each question q_i, we represent its content c_i and each tag t_k^i with d-dimensional vectors. Let $\mathcal{C} \in \mathbb{R}^{n \times d}$ be the feature matrix with each row containing vector representation of question's content c_i or tag t_k^i where $n = |\mathbb{V}|$.

B. Low Quality Question Detection (LQuaD)

a) Module I: In this module, our goal is to capture semantic features from content c_i of the question. First, we pre-process the question's content to remove code snippets, HTML tags, non-ASCII characters, and hypertext references. Then, we fine-tune BERTOverflow [32] model, trained on 152 million sentences from the Stack Overflow's ten-year archive, for our classification task. The BERT model consists of 12 layers, with each layer consisting of an attention mechanism [21] to capture the hidden styles, context, and complexities of low-quality questions effectively. We obtain m-dimensional vector g_i after applying the dropout layer to the pooled output from BERT. Furthermore, the final output is returned by classification layer as shown in Eq. 1.

$$p_i^{bert} = \sigma\ (W_g g_i + b_g) \tag{1}$$

where σ is softmax; p_i^{bert} are the final predictions; weight matrix W_g and bias b_g are trainable parameters.

b) Module II: This module captures the structural relationships between the questions and their corresponding tags. Specifically, we construct a heterogeneous graph consisting of question and tag nodes. We initialize the feature matrix $\mathcal{C}$ by fine-tuning the fastText skip-gram model [33] on the SO data in an unsupervised manner as shown in Fig. 2. Each node $v \in \mathbb{V}$ is associated with a row of matrix $\mathcal{C}$ that gets updated during training. Let $\mathbb{A}$ be the adjacency matrix which represents the connection between the nodes of $\mathbb{G}$. We add self-connections with the equation $\widetilde{\mathbb{A}} = \mathbb{A} + \beta I$ where β is a trainable trade-off parameter. The degree matrix is given as $\mathbb{D}_{ii} = \Sigma_{j=0}^{n} \widetilde{\mathbb{A}}_{ij}$. The normalized adjacency matrix is given by $\hat{\mathbb{A}} = \mathbb{D}^{-\frac{1}{2}} \widetilde{A} \mathbb{D}^{-\frac{1}{2}}$. Eq. 2 shows the GCN layer that updates the node representation using a weighted sum of neighboring node features.

$$h_i^{(l+1)} = ReLU\left(\left(\frac{1}{d_i} h_i^{(l)} + \sum_{j \in \mathcal{N}_i, j \neq i} \frac{1}{\sqrt{d_i d_j}} h_j^{(l)}\right) W^{(l)}\right) \tag{2}$$

where $\mathcal{N}_i$ be the set of neighboring nodes of a i^{th} node; $d_i = |\mathcal{N}_i|$, $h_i^{(l)}$ be the representation of the i^{th} node after layer l, and $W^{(l)}$ be the weight matrix of layer l. Eq. 3 represents the output feature matrix after l layers of GCN.

$$H^{(l+1)} = ReLU\left(\hat{\mathbb{A}} H^{(l)} W^{(l)}\right) \tag{3}$$

where $H^0 = \mathcal{C}$ i.e. the input feature matrix when $l = 0$ depicts initialization of network. We utilize ReLU as non-linearity and use dropout layers between successive convolutions. The first message passing layer facilitates the information flow between questions and their respective tags. Here, the questions capture the aggregate information from their neighborhood nodes (tags). The second layer captures information between questions connected via common tags. Module II employs two layers to execute transductive learning via GCN and employs a classification layer for predictions as shown in Eq. 4.

$$p_i^{gcn} = \sigma(W_h h_i^{(l+1)} + b_h) \tag{4}$$

where σ is softmax; p_i^{gcn} are the final predictions; weight matrix W_h and bias b_h are trainable parameters.

c) Module III: In this module, we combine the predictions (p_i^{bert} and p_i^{gcn}) from Module I and Module II using late fusion techniques [34] in two different settings, i.e., maximum and weighted mean. Eq. 5 and Eq. 6 fuses the predictions from both modules for maximum and weighted mean.

$$\begin{aligned} p_i^{pred} = argmax(max(p_i^{bertc},\ p_i^{gcnc}),\\ max(p_i^{bertnc},\ p_i^{gcnnc})) \end{aligned} \tag{5}$$

$$\begin{aligned} p_i^{pred} = argmax(\lambda * p_i^{bertc} + (1-\lambda) * p_i^{gcnc},\\ \lambda * p_i^{bertnc} + (1-\lambda) * p_i^{gcnnc}) \end{aligned} \tag{6}$$

where λ is a trainable parameter. For Module I, p_i^{bertc} and p_i^{bertnc} are the probabilities obtained for 'closed' and non-'closed' class respectively. Similarly, for Module II, p_i^{gcnc} and p_i^{gcnnc} are the probabilities obtained for 'closed' and non-'closed' respectively.

C. Temporal Event Analysis

We perform survival analysis, a statistical tool to examine the duration until a particular Event Of Interest (EOI) occurs. We define EOI as the time elapsed from the question's creation to when it gets 'closed'. We explore the temporal aspects of the questions before they get 'closed' using non-parametric Kaplan-Meier estimator [13] as shown in Eq. 7. Our dataset is represented more accurately by median than mean using Kaplan-Meier estimator due to its non-parametric nature.

$$\widehat{S}(t) = \prod_{i:t_i \leq t} (1 - \frac{d_i}{n_i}) \tag{7}$$

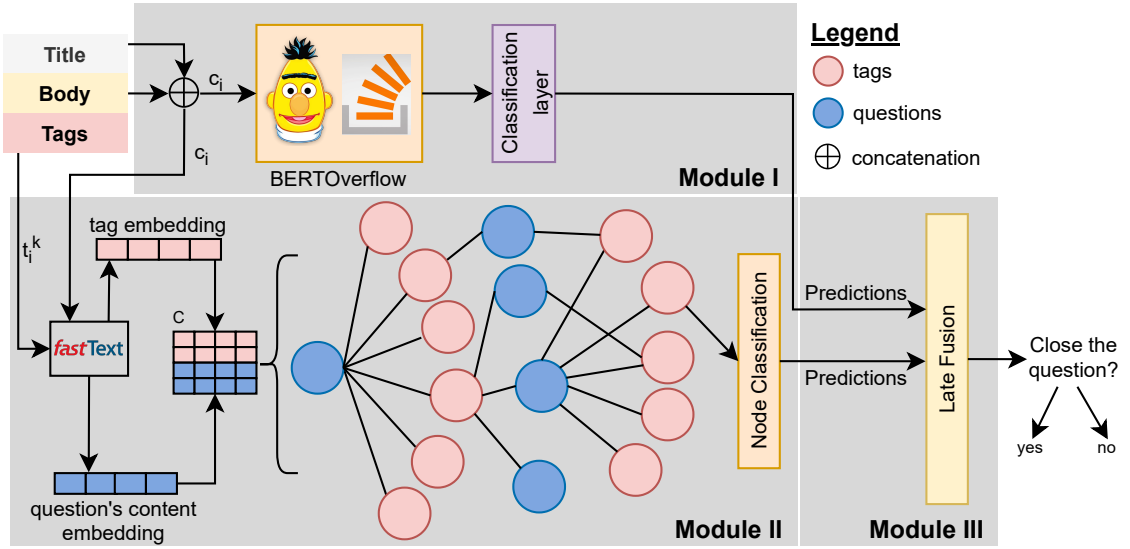

Fig. 2. **LQuaD** consists of three components: Module I fine-tunes the BERTOverflow model, which inputs the title and body for the question's classification task, Module II consists of a graph convolutional network initialized with the question's content and tag embedding for the node classification task, Module III consists of a late fusion strategy that combines predictions from both modules.

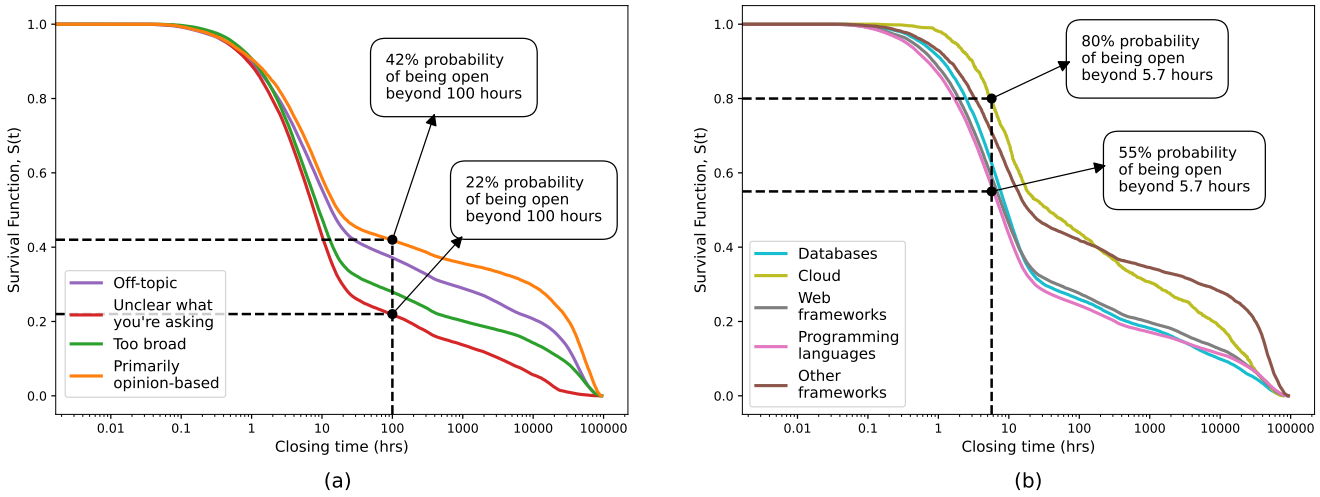

Fig. 3. Kaplan-Meier estimator of survival function for time period for 'closed' questions based on different reasons of 'closing'. Plot (a) specifies the tag category whereas Plot (b) specifies the reasons due to which the question get 'closed'.

where t_i is time at which at question is 'closed', d_i denotes no. of posts 'closed' at time t_i, and n_i denotes no. of posts which haven't been 'closed' upto time t_i but will get 'closed' at some point after t_i. We perform survival analysis by categorizing questions based on the reason for 'closing' and tag categories (see Section IV-A). Table I reports mean time to close the question, and median statistics of the survival function for 'closing reasons' and tag category. We observe that *'primarily opinion-based'* has the highest mean time till EOI and median $\widehat{S}(t)$ while *'unclear what you're asking'* has the least mean till EOI and median $\widehat{S}(t)$. This suggests that *primarily opinion-based* questions remain open for longer time duration than *'unclear what you're asking'* questions. We also observe that *cloud* and *other frameworks* have higher mean time till EOI and median $\widehat{S}(t)$ than other categories. This suggests that question with tags in these categories are not 'closed' for relatively longer time periods. Fig. 3 (a) shows *'primarily opinion-based'* questions have a higher probability of survival than *'unclear what you are asking'* questions for most of the time period after a question's creation. These trends show that the SO community promotes and encourages users to discuss and share opinions on the highly subjective, *'primarily opinion-based'* questions by not 'closing' them during their early stages. On the other hand, the community doesn't support unclear questions by 'closing' them relatively early that inhibits users from spending time in answering these

ambiguous questions.
Fig. 3 (b) shows that the questions with tags related to *cloud* and *other frameworks* consistently have higher survivability (i.e., less chance of getting 'closed' in the future) than question with tags related to *database*, *web frameworks*, and *programming languages.*

IV. Experiments

This section provides dataset description, baselines, and experimental setup.

A. Dataset Description

We describe our dataset creation methodology for LQuaD (Module I and II) and temporal event analysis.

a) LQuaD (Module I) dataset: We create a dataset of 'closed' and non-'closed' questions using a data dump of 17.7M questions from SO platform. First, we filter 847,721 'closed' questions from the data dump of SO posts from August 2008 until June 2019. Table II reports our dataset statistics from SO platform. When SO went live, questions were closed due to seven different reasons. However, SO changed the reasons to close the questions after June 2013. The new reasons to close the question are *'duplicate'*, *'off-topic'*, *'unclear what you're asking'*, *'too broad'*, and *'primarily opinion-based'*. We filter out *'duplicate'* questions while making the 'closed' dataset because these are not necessarily low-quality questions. SO closes duplicate questions to eliminate redundancy over the platform. We consider the pre-submission information of questions present when the question gets 'closed.' It ensures the retention of the original content before the owner makes any edits to improve its quality. We consider only those questions that are 'closed' only once since their creation and not reopened after that. The number of questions in the'closed' class is significantly less than that of the non-'closed' class. Therefore, we randomly sample non-'closed' questions from the data dump such that its size is ten times the size of the 'closed' dataset, which leads to the formation of an imbalanced dataset. We utilize weighted loss [35] to handle the class imbalance problem for our dataset.

b) LQuaD (Module II) dataset: For Module II, we extract the tag-related information corresponding to the questions dataset created in the previous module. A question can have a maximum of five tags as per SO guidelines.[4] However, it turns out that a question has a maximum of six tags in the data dump.[5] Therefore, we find 344,491 questions with only 1-tag, 740,264 questions with 2-tags, 825,405 with 3-tags, 567,209 questions with 4-tags, 374,295 questions with 5-tags, and 173 questions with 6-tags, respectively. Finally, we construct a bipartite graph $\mathbb{G}$ by connecting questions to their corresponding tags. Table II reports the number of nodes, edges, and tags for Module II.

c) Temporal event analysis dataset: We use the 'closed' questions dataset (Module I) and define our EOI for survival analysis as the 'time until a question gets closed.' We collect the temporal data of the question's creation and the 'closing' date for these questions. We filter out the questions from closed dataset with unknown timestamps and 'closing' reasons. To compute the time elapsed for each question, we find the difference between the timestamp of a question's 'closed' date and its creation date. We extract the reasons for closing these questions and found 118,048 *off-topic*, 49,391 *unclear what you're asking*, 52,023 *too broad*, and 22,132 *primarily opinion-based* questions. We also categorize[6] the corresponding tags into 17,774 *database*, 1,438 *cloud*, 20,912 *web frameworks*, 132,875 *programming languages*, and 6,684 *other frameworks*.

B. Baselines

We compare LQuaD with the following state-of-the-art baseline models.

Denzil et al. [14] carry out the characterization study and detection of 'closed' questions using Stochasitc Gradient Boosting Trees (SGBT) classifier to predict whether the question will get 'closed' or not.

Toth et al. [8] introduce a gated-recurrent-unit-based (GRU) classifier that uses textual features of the post to accomplish binary classification task.

CountVec + LR [36] We utilize count vectorizer as a feature extractor module for post's textual data and pass it through the logistic regression model.

CountVec + XGBoost [37] We utilize count vectorizer to extract features from textual data and pass it through the XGBoost classifier model.

FastText + LR [33] We extract the pre-trained embeddings using fastText and pass it through the Logistic regression model.

Distilled-BERT + LR [38] We utilize the pre-trained embeddings using DistilBERT and pass it through the Logistic Regression classifier model.

GCN (fastText) [39] We initialize the node embedddings using pre-trained embeddings from fastText [33] model and pass it through the Module II of LQuaD.

C. Experimental setup

Table III reports the optimal values after tuning hyperparameters of Module I and II of LQuaD. In Module I, we use the coarse grid-search algorithm to search the hyperparameter space. The BERTOverflow model converges before one epoch during the fine-tuning task. In Module II, we initialize nodes with d-dimensional vector representation using fine-tuned fastText model where d=300 and run it for 500 epochs. We find that the trainable trade-off parameter, β, has optimal value two. We use the train-validation-test (60:20:20) split and perform 10-fold calculations for classification run of Module I and II. For Module III, the optimal value of λ for weighted

[4]https://stackoverflow.com/help/tagging

[5]https://stackoverflow.com/questions/14071930/creating-a-playlist-out-of-songs-from-selected-artists

[6]https://insights.stackoverflow.com/survey/2021

TABLE I
WE REPORT THE MEAN TIME (DAYS) TILL EOI AND MEDIAN SURVIVAL TIME (HRS) CORRESPONDING TO REASONS AND TAG CATEGORIES. THE HIGHEST VALUES IN RESPECTIVE ROWS ARE SHOWN IN BOLD AND THE LOWEST VALUES ARE UNDERLINED.

	Reasons				**Tag Categories**				
Category	*Off-topic*	*Unclear what you're asking*	*Too broad*	*Primarily opinion-based*	*Database*	*Cloud*	*Web frameworks*	*Programming languages*	*Other frameworks*
Mean Time Till EOI (days)	330.76	65.91	230.87	**531.81**	145.54	228.13	185.03	177.89	**482.89**
Median Survival Time (hrs)	12.89	7.51	8.57	**15.07**	9.04	**27.33**	8.17	7.27	16.51

TABLE II
DATASET STATISTICS FROM THE STACK OVERFLOW PLATFORM.

	Module I			**Module II**	
Dataset	*No. of train samples*	*No. of val. (test) samples*	*Total*	*Graph*	*Total*
'Closed'	155,554	51,852	259,258	Nodes (questions)	2,851,838
Non-'Closed'	1,555,548	518,516	2,592,580	Nodes (unique tags)	48,374
Total questions	1,711,102	570,368	2,851,838	Edges	8,442,584

TABLE III
HYPERPARAMETER SETTINGS FOR MODULE I AND MODULE II OF LQUAD.

Module I		**Module II**	
Hyperparameter	Value	Hyperparameter	Value
Batch size	8	No. of layers	2
Learning rate	1e−5	Learning rate	1e−2
Weight Decay	0.01	No. of output channels after each GCN layer	64
Dropout	0.1	Dropout	0.5

mean technique is obtained by tuning its value in the uniform distribution $\mathcal{U}\{0, 1\}$. Further, we apply 5-Fold cross-validation during late fusion to obtain $\lambda = 0.55$. We use cross-entropy loss and the Adam optimizer [40] to train LQuaD.

TABLE IV
AVERAGE PERFORMANCE METRICS (WEIGHTED) ON THE STACK OVERFLOW DATASET.

Model	**Precision**	**Recall**	**F1**
Denzil et al. [14]	70.25	70.25	70.24
Toth et al. [8]	73.78	73.33	73.66
Count Vectorizer + LR [36]	89.94	76.90	81.38
Count Vectorizer + XGBoost [37]	90.17	77.48	81.82
FastText + LR [33]	90.52	79.71	83.44
DistilBERT + LR [38]	92.19	85.26	87.59
GCN (fastText) [39]	90.69	83.98	86.42
LQuaD (LF [mean])	**94.85**	**95.20**	**94.86**

V. RESULTS AND ANALYSIS

We compare LQuaD with different state-of-the-art baselines (see Table IV). Denzil et al. [14] also predicted closed questions based on various predictive features using post-submission information with F1-score of 70.24%. Toth et al. [8] similar to our work, predicted question closing based on pre-submission information using a gated recurrent unit and obtained an F1-score of 73.66%. LQuaD (LF[max]) and LQuaD (LF[mean]) outperforms by 19% and 21% as compared to the best baseline [8] for the binary classification task. Table IV shows the average over the results obtained from 10-fold calculations. We perform the student's t-test [41] test to compare the results obtained from the modules. The t-test results of Module I and II follow the distribution and differ significantly from each other at the applied significance level $p < 0.01$.

Analysis: We analyze the attention layers of fine-tuned model in Module I. Fig. 4 (a) shows that the model gives higher attention to words like *'tutorial'* and *'beginner'*. The model is able to capture the newbie behavior over the platform, and the question is correctly predicted as 'closed'. Similarly, Fig. 4 (d) shows that the model give higher attention to words like {'error', 'array', 'code', 'compile'}. Therefore, the model can capture semantic and contextual information about the *'debugging a coding error'* which represents the relevant question posted over the platform. LQuaD is able to capture tag information such as '*agile*' and *'open-source'* as majority of questions containing these tags are 'closed'. The questions containing these tags are correctly predicted as 'closed'. The SO website also clarifies that questions using *'open-source'* and *'agile'* tags would be considered *off-topic*.[7] Therefore, LQuaD is able to model the structural relationships between the questions and tags effectively. Fig. 5 depicts the node representations before and after training of Module II using PCA [42]. We observe that Module II learns tags patterns associated with questions in a transductive manner.

VI. ABLATION STUDY

Table V shows the ablation study of Module I and II separately. Module I achieves Weighted performance metrics, i.e., Precision, Recall, and F1-score of 94.81, 95.08, and 94.53, respectively. Similarly, Module II, i.e., GCN model initialized with fine-tuned fastText embeddings achieves Precision, Recall and F1-score of 91.86, 84.38, and 86.92 respectively. These results show that GCNs achieved comparable precision and relatively lower recall than transformer-based models.

VII. ERROR ANALYSIS

We demonstrate some of the errors encountered by LQuaD. For example, words like *'java programming'* and *'MYSQL database'* are weighted more by the underlying attention mechanism. However, due to these words, the model incorrectly predicted the 'closed' as non-'closed'. Additionally,

[7]https://stackoverflow.com/questions/tagged

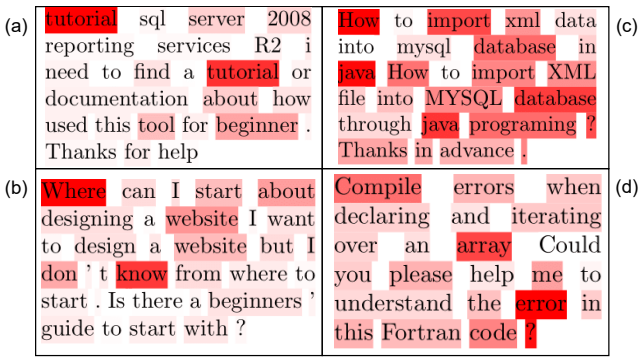

Fig. 4. Some examples of predictions using Module I. For each token in the input, we show the visualization of self-attention averaged over all 12- attention heads of fine-tuned BERT Overflow. Higher attention weights corresponds to darker color (red). Plot (a) is an example of 'closed' predicted correctly, Plot (b) is an example of non-'closed' predicted as 'closed', Plot (c) is an example of 'closed' predicted as non-'closed' and Plot (d) is an example of non-'closed' predicted correctly.

TABLE V
EFFECTIVENESS OF LQUAD (AND VARIANCES) AS COMPARED TO MODULE I AND MODULE II.

Model	Precision	Recall	F1
Module I	94.81 (0.03)	95.08 (0.04)	94.53 (0.01)
Module II	91.86 (0.07)	84.38 (0.07)	86.92 (0.08)
LQuaD (LF [max])	93.62 (0.02)	92.85 (0.04)	93.16 (0.07)
LQuaD (LF [mean])	**94.85 (0.03)**	**95.20 (0.05)**	**94.86 (0.02)**

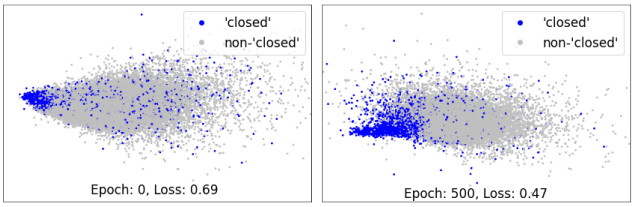

Fig. 5. PCA visualizations of node representations for 'closed' and 'non'-closed questions before (left) and after (right) training.

'Where can .....with' shows that the actual class is non-'closed' whereas the model predicted it 'closed' which is an example of false positive. Our transformer-based model is able to look at some words like 'array' and 'error' to make the decision for non-'closed' class. *'Compile .....code'* shows that the actual class is non-'closed' and our model predicted it correctly as well.

VIII. CONCLUSION AND FUTURE WORK

We propose LQuaD that incorporates semantic information of questions associated with each post using transformers and learns question and tag graphs in a transductive manner using GCNs. Our graph consists of 2.9M nodes and 8.4M edges. LQuaD detects low-quality questions that are likely to get 'closed' at the time of posting. Our framework acts as an early-assessment tool to assist users in composing a question, which would remain open and receive responses. Further, we use survival analysis that reduces the number of questions closed by informing users to take appropriate action. We find that the timeframe between the stages from the question's creation till it gets 'closed' varies significantly for tags and different 'closing' reasons for these questions. LQuaD outperforms the state-of-the-art methods by a 21% in F1-score on the dataset of $2.8M$ questions. We have made the code and dataset available for reproducibility. In the future, we plan to study the 'closed' questions that gets re-open after edits.

REFERENCES

[1] A. Baltadzhieva and G. Chrupała, "Predicting the quality of questions on stackoverflow," in *Proceedings of the international conference recent advances in natural language processing*, 2015, pp. 32–40. 1, 2
[2] D. Correa and A. Sureka, "Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow," in *Proceedings of the 23rd international conference on World wide web*, 2014, pp. 631–642. 1, 2
[3] S. Ravi, B. Pang, V. Rastogi, and R. Kumar, "Great question! question quality in community q&a," 2014. 1
[4] M. Neshati, "On early detection of high voted q&a on stack overflow," *Information Processing & Management*, vol. 53, no. 4, pp. 780–798, 2017. 1
[5] P. K. Roy and J. P. Singh, "Predicting closed questions on community question answering sites using convolutional neural network," *Neural Computing and Applications*, pp. 1–18, 2019. 1, 2
[6] M. K. Ho, S. Tatinati, and A. W. Khong, "Distilling essence of a question: A hierarchical architecture for question quality in community question answering sites," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7. 1
[7] L. Tóth, B. Nagy, D. Janthó, L. Vidács, and T. Gyimóthy, "Towards an accurate prediction of the question quality on stack overflow using a deep-learning-based nlp approach." in *ICSOFT*, 2019, pp. 631–639. 1
[8] L. Tóth, B. Nagy, T. Gyimóthy, and L. Vidács, "Why will my question be closed? nlp-based pre-submission predictions of question closing reasons on stack overflow," in *2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2020, pp. 45–48. 1, 5, 6
[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019. 2
[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017. 2
[11] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017. 2
[12] F. Ortega, G. Convertino, M. Zancanaro, and T. Piccardi, "Assessing the performance of question-and-answer communities using survival analysis," *arXiv preprint arXiv:1407.5903*, 2014. 2
[13] E. L. Kaplan and P. Meier, "Nonparametric estimation from incomplete observations," *Journal of the American statistical association*, vol. 53, no. 282, pp. 457–481, 1958. 2, 3
[14] D. Correa and A. Sureka, "Fit or unfit: analysis and prediction of 'closed questions' on stack overflow," in *Proceedings of the first ACM conference on Online social networks*, 2013, pp. 201–212. 5, 6
[15] T. Ahmed and A. Srivastava, "Understanding and evaluating the behavior of technical users. a study of developer interaction at stackoverflow," *Human-centric Computing and Information Sciences*, vol. 7, no. 1, p. 8, 2017. 2
[16] J. Trienes and K. Balog, "Identifying unclear questions in community question answering websites," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11437 LNCS, pp. 276–289, 2019. 2
[17] D. Thukral, A. Pandey, R. Gupta, V. Goyal, and T. Chakraborty, "Diffque: Estimating relative difficulty of questions in community question answering services," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 4, pp. 1–27, 2019. 2

[18] D. Hoogeveen, A. Bennett, Y. Li, K. M. Verspoor, and T. Baldwin, "Detecting misflagged duplicate questions in community question-answering archives," in *Twelfth international AAAI conference on web and social media*, 2018. 2

[19] R. F. Silva, K. Paixão, and M. de Almeida Maia, "Duplicate question detection in stack overflow: A reproducibility study," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2018, pp. 572–581. 2

[20] D. Liang, F. Zhang, W. Zhang, Q. Zhang, J. Fu, M. Peng, T. Gui, and X. Huang, "Adaptive multi-attention network incorporating answer information for duplicate question detection," *SIGIR 2019 - Proceedings of the 42nd International ACM SIGIR Conference on RD in IR*, pp. 95–104, 2019. 2

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. 2, 3

[22] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 3700–3710. 2

[23] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035. 2

[24] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 7, pp. 5966–5978, 2021. 2

[25] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020. 2

[26] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7370–7377. 2

[27] D. W. Hosmer Jr, S. Lemeshow, and S. May, *Applied survival analysis: regression modeling of time-to-event data*. John Wiley & Sons, 2011, vol. 618. 2

[28] N. Sachdeva and P. Kumaraguru, "Call for service: Characterizing and modeling police response to serviceable requests on facebook," in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2017, pp. 336–352. 2

[29] S. Asthana, P. Singh, and P. Gupta, "Survival analysis: Objective assessment of wait time in hci," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 367–376. 2

[30] F. Ortega, G. Convertino, M. Zancanaro, and T. Piccardi, "Assessing the performance of question-and-answer communities using survival analysis," *CoRR*, vol. abs/1407.5903, 2014. 2

[31] L. Lord, J. Sell, F. Bagirov, and M. Newman, "Survival analysis within stack overflow: Python and r," in *2018 4th International Conference on Big Data Innovations and Applications (Innovate-Data)*. IEEE Computer Society, 2018, pp. 51–59. 2

[32] J. Tabassum, M. Maddela, W. Xu, and A. Ritter, "Code and named entity recognition in stackoverflow," in *The Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020. 3

[33] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. 3, 5, 6

[34] G. Baldewijns, G. Debard, G. Mertes, T. Croonenborghs, and B. Vanrumste, "Improving the accuracy of existing camera based fall detection algorithms through late fusion," in *2017 39th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*. IEEE, 2017, pp. 2667–2671. 3

[35] S. Susan, D. Sethi, and K. Arora, "Cw-cae: pulmonary nodule detection from imbalanced dataset using class-weighted convolutional autoencoder," in *International Conference on innovative computing and communications*. Springer, 2021, pp. 825–833. 5

[36] W. Arshad, M. Ali, M. Mumtaz Ali, A. Javed, and S. Hussain, "Multi-class text classification: Model comparison and selection," in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2021, pp. 1–5. 5, 6

[37] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015. 5, 6

[38] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019. 5, 6

[39] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 5, 6

[40] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–2. 6

[41] Student, "The probable error of a mean," *Biometrika*, pp. 1–25, 1908. 6

[42] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. 6