1 | INTRODUCTION

Developers can post questions about the problems, errors and any bugs they find, and seek solutions to those problems or any work around for them in question and answer sites (Q&A site) [1]. Stack Overflow\(^1\) is such a Q&A site. It has become an important source for developers to solve various software development-related problems [2,3]. In Stack Overflow, the popularity of a question describes how much attention it receives, which could be measured by the amount of total views [1]. Some questions become hot, and receive high views. These hot questions are of widespread concern to developers, and their solutions help more developers than cold questions. Therefore, it is important to predict hot questions soon after their posts, and give priority to recommend these questions to answers, so as to shorten their response time and increase the response rates. Furthermore, Stack Overflow allows companies to post developer-related product advertisements\(^2\) or job advertisements\(^3\) on the website. Product advertisements are charged based on a Cost Per Mille (CPM) or Cost Per Click (CPC) basis. Advertisements on hot questions receive more attention than those on cold questions, which is conducive to achieving the advertising promotion effect. The prediction of hot questions is useful for making advertising plan, planning advertising campaigns and estimating costs in Stack Overflow.

In recent decades, the popularity prediction has attracted great attention [4–6]. There have been several studies on popularity prediction based on temporal process features [7–9] or text-based features [10]. Some of the above studies used the number of online content forwarded, others used the number of followers of the user who posted the online content, etc. However, these attributes are not available in Stack Overflow. Therefore, the methods proposed in the above studies cannot be used to predict the hot questions of

\(^1\)http://stackoverflow.com/
\(^2\)https://stackoverflow.com/advertising
\(^3\)https://stackoverflow.com/talent/en

---

**Abstract**

Stack Overflow is a very popular programming question and answer community. Some questions become hot, and receive high views, which are of widespread concern to developers. Finding hot questions early can give priority to recommend potential hot questions to answers, thereby shortening the response time. Besides, the hot question prediction is also helpful for making advertising plan, planning advertising campaigns and estimating costs. Therefore, it is important to predict hot questions. The authors propose the VSAF method which analyses the View amount changes, Answer amount changes and Score changes soon after questions’ creation based on Fully convolutional neural network. The performance of the VSAF method based on a training set and two different test sets has been evaluated. The training set has 1600 hot questions and 1600 cold questions. The random test set has 381 hot questions and 2819 cold questions, while the balanced test set has 400 hot questions and 400 cold questions. The experimental results show that using the balanced test set, VSAF achieves Accuracy, $F_1_{\text{hot}}$ and $F_1_{\text{cold}}$ of 80%, 77.77% and 81.81%, which outperforms the baseline approach by 25.59%, 21.52% and 29.04%, respectively. Using the random test set for evaluation, VSAF achieves Accuracy, $F_1_{\text{hot}}$ and $F_1_{\text{cold}}$ of 84.91%, 53.96% and 90.97%, which outperforms the baseline approach by 31.83%, 84.16% and 19.35%, respectively. The VSAF method significantly outperforms the state-of-the-art approach on hot question prediction.
Stack Overflow. Stack Overflow has some own attributes, such as questions’ answer amount, scores, etc. It is necessary to explore a more suitable method for hot question prediction in Stack Overflow.

Phukan et al. presented the first look at predicting hot questions based on text-based feature in Stack Overflow [1]. However, they ignored initial_feedback features, such as the score changes, answer amount changes of questions soon after they were posted. We explore the effectiveness of using initial_feedback features to predict hot questions.

According to previous works [1,11], we use the number of views to characterise whether the question is hot or not. For a question, it is either a hot question or a cold question. Therefore, the hot question prediction can be regarded as a binary classification problem. We propose the VSAF method which analyses questions’ initial_feedback features, including the View amount changes, Answer amount changes and Score changes soon after questions’ creation, based on Fully convolutional neural network. According to the initial_feedback features within K hour(s) of question’s posting, we predict whether the question will be hot T days after it is posted. K and T are defined in Section 4. If there is no special instruction, K and T are set to 3 hours and 7 days by default.

We evaluate the performance of our method (VSAF) based on a training set and two different test sets. The training set has 1600 hot questions and 1600 cold questions. Random test set has 381 hot questions and 2819 cold questions, while balanced test set has 400 hot questions and 400 cold questions. The baseline method [1] mainly uses text features, and the proposed VSAF method mainly uses temporal features. The experimental results show that, using the balanced test set for evaluation, VSAF achieves Accuracy, F1_hot and F1_cold of 80%, 77.77% and 81.81%, which outperforms baseline approach [1] by 25.59%, 21.52% and 29.04%, respectively. Using the random test set for evaluation, VSAF achieves Accuracy, F1_hot and F1_cold of 84.91%, 53.96% and 90.97%, which outperforms baseline approach [1] by 31.83%, 84.16% and 19.35%, respectively. The VSAF method significantly outperforms state-of-the-art approach on hot question prediction.

The main contributions of this paper are as follows:

1. Based on Fully convolutional neural network, we propose the VSAF method to predict hot questions of Stack Overflow, which analyses questions’ initial_feedback features, including the view amount changes, answer amount changes and score changes within K hour(s) of question’s posting.
2. We analyse the real questions in Stack Overflow, and experiments show that our method VSAF outperforms baseline approach [1]. And the effect of using initial_feedback features alone is better than that of combining initial_feedback features and text features.

The reminder of the article is organised as follow. Section 2 presents the background and data collection. Section 3 presents our prediction approach VSAF. Section 4 conducts experiments to evaluate the proposed method. Section 5 discusses threats to validity and Section 6 discusses related works. Finally, Section 7 concludes this paper.

2 RELATED WORK

2.1 Research on popularity of web content

Some researchers used social network structures to predict content’s popularity. Zhang et al. predicted the popularity of an event through a diffusion model that took the content of the event and developer’s information into account [12]. Shengxian et al. proposed a message popularity prediction method based on propagation simulation [13]. Zhang et al. [14] used developer-guided hierarchical attention networks to predict social image popularity based on developer’s features and pattern content.

Besides, Piotrkowicz et al. used only the title of articles to predict the popularity of news articles [10]. They analysed features such as the number of entity mentions in headlines of the relevant news outlet. Sanjo et al. used image and short text features in recipes to propose a visual semantic fusion model to predict the popularity of online recipes [15].

The above studies use some features that are not available in Stack Overflow, such as users’ followers, articles’ publication journals, the number of online content forwarded, the number of entity mentions in headlines of the relevant news outlet, etc. Therefore, the methods proposed in the above studies cannot be directly used to predict the hot questions of Stack Overflow.

2.2 Research on hot questions on Stack Overflow

Few people currently research hot questions in Stack Overflow. Phukan et al. predicted questions’ final popularity based on the initial content of questions on the site, used views as an evaluation metric for popularity [1]. They used the term frequency-inverse document frequency model to extract feature vectors from the processed question’s title, body and tags, and assigned different weights to these three feature vectors. Finally, they used a variety of classification algorithms to binary categorise questions to predict whether they would be hot.

Experiments show that our method VSAF is better than the method proposed by previous work [1].

2.3 Other research on Stack Overflow

Zhou et al. found that questions with bounties were answered in most cases [16]. Prohr-Petrus et al. proposed the POSIT method, which used abstract syntax tree tags from the CLANG compiler and part-of-speech tags from the Stanford part-of-speech tagger, to simultaneously tag natural and programming languages in questions [17]. Haoxiang
Zhang et al. investigated how the knowledge in answers became obsolete and identified the characteristics of such obsolete answers [18]. In addition, some scholars studied the obsolescence of API in questions [19–21]. Xiaoxue Ren et al. designed an automatic open information extraction approach for systematically discovering and summarizing the controversies in Stack Overflow [22]. In addition, there have been many studies on the quality of questions and answers in Stack Overflow [18,23–25].

There are many related researches on Stack Overflow, but the research on prediction of hot questions is still lacking.

3 | BACKGROUND AND DATA COLLECTION

In this section, we introduce the background information of Stack Overflow in more detail. In addition, our data collection process is described.

3.1 | Background

The Stack Overflow website has been in use since 2008. It is one of the largest Q&A sites where developers can share knowledge and seek experts' advice on a wide range of topics in computer programming. Stack Overflow developer base is increasing exponentially and many new questions are being posted everyday [26].

In Stack Overflow, developers can ask questions, browse questions, view answers to questions and give answers [27] to questions. Furthermore, they can vote up or down for a question. The question's score is calculated by subtracting the number of positive (up) votes from the number of negative (down) votes. Figure 1 shows an example of a question. This question was viewed for 16 times and receives 2 answers. Since this question receives 20 up votes and 6 down votes, its score is 14.

3.2 | Data collection

We collect data sets from Stack Overflow. First, we aim to collect new questions' IDs and their posting time. Stack Overflow provides a “Newest” interface, which shows newly created questions. According to this “Newest” interface, we continuously crawl the newly posted questions' IDs and their posting time from November 24, 2019 to December 7, 2019. After obtaining new questions' IDs, we collect detailed information on the above questions. For each question, we collect its titles, bodies and tags. Then we collect each question's views, answers and score per hour within 10 hours of the question being posted. Finally, we collect the questions' views 7 days after it was posted.

The site takes some measures to prevent it from being overvisited. Every application is subject to an IP based concurrent request throttle. Therefore, we use an IP proxy pool to break the single IP limit. IP proxy pool gathers the valid proxy IPs acquired. We configure them in our crawler application. When one IP fails, another valid IP is used immediately, thus avoiding the access restrictions of a single IP.

4 | APPROACH

In this section, we describe our research method VSAF. It is based on the fully convolutional neural network and takes into account the initial_feedback features of questions, including view amount changes, score changes and answer amount changes within \( K \) hour(s) of questions' posting. \( K \) represents the observation duration of questions. We explore the influence of the setting of \( K \) on the prediction effect of hot questions in Subsection 5.4.4. Based on the initial_feedback features within \( K \) hour(s) of question's posting, we predict whether the question will be hot \( T \) days after it is posted. Developers can adjust \( K \) and \( T \) according to the actual needs of the project. In this article, if there is no special instruction, \( K \) and \( T \) are set to 3 hours and 7 days by default. The research framework of this approach is shown in Figure 2. The specific structure of the fully convolutional neural network model is defined in Subsection 4.1.4. Our method is divided into two phases: a training phase and a prediction phase. In the training phase, we aim to build the fully convolutional neural network based on historical information. In the prediction phase, we use the fully convolutional neural network to predict hot questions.

In the training phase, the VSAF method separately obtains the view amount changes, score changes and answer amount changes within \( K \) hour(s) of questions' posting, which construct initial_feedback features of the question. Then, we get the views of the question after 7 days of its creation, and decide its hot level. If the view amount is larger than or equal to 50, then the hot level is considered as hot; otherwise, this question is considered as a cold one. According to the initial_feedback features and hot levels of questions, we obtain the prediction model. Fully convolutional neural network is a kind of convolutional neural networks, which is widely used for classification tasks. It has the advantages of not being easy to overfit and being able to adapt to different input sizes [28]. Therefore, we use a fully convolutional neural network to build the prediction model.

In the test phase, the VSAF method obtains the view amount changes, score changes and answer amount changes soon after new questions' creation, and combines them into the initial_feedback features. Then, the features obtained during the training phase are input into the prediction model. Finally, we get the prediction results of whether the questions are hot or not.
**FIGURE 1**  Problem structure in Stack Overflow

**FIGURE 2**  Research framework
4.1 Training phase

4.1.1 View amount changes within K hour(s)

When a question is posted, it can be viewed by developers. Hot questions may receive more views soon after their creation, and the view amount in the early time may be useful for predicting hot questions [29]. Therefore, after the question is posted, we get the number of views \( v_i \) of the question every 1 hour until the question is posted for \( K \) hours. The formula for calculating the changes in the number of question’s views per hour \( Q_{vc} \) is as follows:

\[
Q_{vc} = (v_1, \ldots, v_i)(1 \leq i \leq K)
\]

\( v_i \) represents the number of question’s views between \( i - 1 \) hour and \( i \) hours after being posted, it is defined as

\[
v_i = \begin{cases} 
  v_1 & i = 1 \\
  v_i - v_{i-1} & i > 1
\end{cases}
\]

4.1.2 Score changes within K hour(s)

After a question is posted, developers can vote up or down on the question. The question’s score is the difference between the number of up votes and the number of down votes. Yao et al. [25] found that high-scoring questions tended to get more attention, because high-score questions were clearer and had better question quality. Due to better quality, potential hot questions may have higher scores soon after their creation, and thus the question’s initial score may be useful in hot question prediction. After the question is posted, we get the score \( s_i \) of the question every 1 hour until the question is posted for \( K \) hours. The formula for calculating the changes in question’s score per hour \( Q_{sc} \) is as follows:

\[
Q_{sc} = (s_1, \ldots, s_i)(1 \leq i \leq K)
\]

\( s_i \) represents question’s score changes between \( i - 1 \) hour and \( i \) hours after being posted, it is defined as

\[
s_i = \begin{cases} 
  s_1 & i = 1 \\
  s_i - s_{i-1} & i > 1
\end{cases}
\]

4.1.3 Answer amount changes within K hour(s)

After a new question is posted, developers can answer it. Neshati et al. [30] found that questions with answers are more attractive to the community. Because many developers visit websites to find solutions to programming questions, questions with answers are more helpful for developers and they are more likely to become hot. After the question is posted, we get the number of answers \( a_i \) of the question every 1 hour until the question is posted for \( K \) hours. The formula for calculating the changes in question’s answers per hour \( Q_{ac} \) is as follows:

\[
Q_{ac} = (a_1, \ldots, a_i)(1 \leq i \leq K)
\]

\( a_i \) represents the number of question’s answers between \( i - 1 \) hour and \( i \) hours after being posted, it is defined as

\[
a_i = \begin{cases} 
  a_1 & i = 1 \\
  a_i - a_{i-1} & i > 1
\end{cases}
\]

4.1.4 Get fully convolutional neural network

The next step is to build fully convolutional neural network in training phase. Fully convolutional neural network is a kind of convolutional neural networks, which is widely used for classification tasks. Because convolutional neural networks have the disadvantages of consuming a lot of storage space and low computing efficiency [31], Long et al. [32] proposed Fully Convolutional Neural Network (FCN) algorithm in 2015. We use the fully convolutional neural network to build the prediction model. The specific structure of the fully convolutional neural network model used is shown in Figure 3.

In Subsections 3.1.1, 3.1.2 and 3.1.3, we obtain the hourly change in views \( Q_{vc} \) of \( Q_{vc} = (v_1, \ldots, v_i) \), score \( Q_{sc} = (s_1, \ldots, s_i) \) and answers \( Q_{ac} = (a_1, \ldots, a_i) \). We combine them into a large vector \( Q = (Q_{vc}, Q_{sc}, Q_{ac}) \), then we get the matrix \( Q = (Q_1, Q_2, \ldots, Q_n) \), where \( n \) represents the size of the training set.

We input \( Q \) into the convolutional layer and convert it into the input \( P_b \) of the pooling layer after three times of convolution layer processing. Convolutional layer processing consists of three steps, namely convolution operation, Batch Normalization (BN) and activation function processing. Next step, we describe the convolutional layer processing in detail.

Firstly, we normalise this matrix \( Q \) and perform a convolution operation to convert it to input \( BN \) for BN. The convolution process is shown below. In the formula, \( Q \) represents the input matrix, \( W \) represents the convolution kernel, * represents the convolution operation, \( i \), \( j \) represents the rows and columns of the matrix, respectively.

\[
BN(i,j) = (Q * W)(i,j)
\]

\[
(Q * W)(i,j) = \sum_{m,n} Q(i - m, j - n) w(m, n)
\]

Secondly, \( BN \) is converted into the input \( AC \) of the activation function through the BN process. BN is used to make the training process more stable and ensure that the input of each layer of the neural network maintains the same distribution. \( s \) is the size of the batch data of the neural network. For a batch, we firstly obtain the mean of the batch data, then calculate the variance, normalise each question’s processed features in the batch. Finally, in order to maintain the same distribution of input at each layer of the neural network, the scaling and offset operations are carried out.
The mean calculation formula is defined as

$$\mu_B = \frac{1}{s} \sum_{i=1}^{s} BN_i$$  \hspace{1cm} (9)

The variance calculation formula is defined as

$$\sigma_B^2 = \frac{1}{s} \sum_{i=1}^{s} (BN_i - \mu_B)^2$$  \hspace{1cm} (10)

The normalization method is shown below:

$$AC_i = \frac{BN_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$  \hspace{1cm} (11)

Finally, AC is transformed into the input $P_b$ of the pooling layer through the activation function processing. After the activation function is introduced, the neural network is no longer a linear combination of inputs and can approximate any function. Traditional activation functions include Sigmoid activation function and TanH activation function. These two activation functions have high calculation complexity and need to calculate the index.ReLU calculates activation values through thresholds, which greatly reduces the calculation complexity and greatly accelerates the speed of convergence. Therefore, we use the rectified linear unit (ReLU) activation function. At this point, we complete the transformation of data from the convolutional layer to the pooling layer.

$P_b$ is transformed into the input $P$ of the Softmax layer through global average pooling conversion. We perform global average pooling to prevent overfitting. Then, $P$ enters the Softmax layer to get the probability that the questions belongs to a certain category.

Through the Softmax layer, the data $P$ obtained after pooling is converted into the predicted value $\hat{Y}$. The $\hat{y}_j$ represents the probability that the $i$th question belongs to the $j$th category. In fact, the actual category of questions in the training data set is $Y$. Each row of $Y$ has only one value of 1 and the rest is 0. $\hat{y}_j = 1$ means the $i$th question belongs to the $j$th category. For example, if $\hat{y}_1 = 1$, it means that the $i$th question belongs to the first category and is a cold question. Otherwise, if $\hat{y}_2 = 1$, it means that the $i$th question belongs to the second category and is a hot question. According to the true category $Y$ and the prediction category $\hat{Y}$ of the questions, we obtain the total cost function of the prediction model, which is defined as follows. $cn$ is the number of categories of questions.

$$COST = \sum \left( -\sum_{j=1}^{cn} y_{ij} \ln \hat{y}_{ij} \right)$$  \hspace{1cm} (12)

The goal of training phase is to reduce errors between predicted values and actual values so as to get a good prediction effect. The smaller the value of the cost function, the better the prediction effect. Therefore, we use a gradient descent optimization algorithm. The gradient descent optimization algorithm is generally used to minimise the value of the cost function.

### 4.2 Prediction phase

Given a new question, we collect its number of views, score and number of answers per hour soon after its creation, and
compute view amount changes, score changes and answer amount changes. We combine them into a large vector, input it into the prediction model based on fully convolutional neural network, and predict whether the new question will become hot or not.

5 | EXPERIMENT

To evaluate the effectiveness of our proposed method VSAF, we conduct several experiments based on real data from Stack Overflow.

5.1 | Datasets and setup

Using the data collection method proposed in Section 2.2, we obtain question data. We make the following definitions.

**Hot question:** questions with views greater than or equal to 50 (views > 50) 7 days after they are posted.

**Cold question:** questions with views less than 50 (views < 50) 7 days after they are posted.

In our datasets, 12.36% of questions are hot, while 87.64% of questions are cold. The number of hot questions is fewer than the number of cold questions. In practice, the administrators can set the threshold of views (hot_viewsAmount) and days (T) based on the actual needs. For example, they can define as follows. If a question with views greater than or equal to hot_viewsAmount after it is posted T days, it is a hot question, otherwise it is a cold question. T is defined in Section 4.

We refer to the method of Liao et al. [11], using a balanced training set to build the prediction model, and use two test sets, including a balanced test set and a random test set. The ratio of the number of hot questions and the number of cold questions in the balanced set is 1, while the ratio of the number of hot questions and the number of cold questions in the random set is closer to the real situation. Using the balanced training set can ensure the number of hot questions and prevent overfitting problem. Using the balanced test set can evaluate the effectiveness of the method in detecting hot questions. Using the random test set can better reflect the prediction effect of the prediction model in real applications. The questions in both test sets are posted later than those in the training set. The method of building the training set and test sets is described below.

First, we obtain the training set. In order to ensure the balance of the training data, we extract 1600 hot questions and 1600 cold questions as the training set.

Second, we obtain the balanced test set. The later the question is posted, the larger the question's ID is. The maximum ID of the questions in the training set is maxID. We randomly select 400 cold questions and 400 hot questions whose ID are larger than maxID. It ensures that the questions in the balanced test set are posted later than those in the training set.

Finally, we obtain the random test set. We randomly select 3200 questions with ID > maxID as the random test set. The random test set includes 381 hot questions and 2819 cold questions.

We use the training set to obtain the prediction model, use the balanced test set and the random test set to evaluate the performance of VSAF method. We use the initial_feedback features within K hour(s) of question's posting to predict whether the question is hot or not. K is defined in Section 4. It represents the observation duration of questions. In the following experiments K is set to 3 by default if there is no special instruction. We explore the influence of the setting of K on the prediction effect of hot questions in Subsection 5.4.4.

The data set used in this experiment is shown in Table 1.

5.2 | Evaluation metrics

In order to evaluate our method VSAF, we use metrics Accuracy, Precision, Recall and F1-score. These metrics are commonly used in evaluation of popularity prediction [1,11].

**Accuracy** is the most common evaluation metric, which is the number of correctly classified samples divided by the total number of samples. The calculation formula is as follows:

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

(13)

TP indicates the number of hot questions predicted as hot, TN indicates the number of cold questions predicted as cold, FP indicates the number of hot questions predicted as cold and FN indicates the number of cold questions predicted as hot.

**Precision** is defined as

$$\text{Precision}_{\text{hot}} = \frac{TP}{TP + FP}$$

(14)

$$\text{Precision}_{\text{cold}} = \frac{TN}{TN + FN}$$

(15)

**Precision_{hot}** is the ratio of correctly predicted hot questions over all the questions predicted as hot. **Precision_{cold}** is the ratio of correctly predicted cold questions over all the questions predicted as cold.

**Recall** is a measure of coverage, it is defined as

$$\text{Recall}_{\text{hot}} = \frac{TP}{TP + FN}$$

(16)

$$\text{Recall}_{\text{cold}} = \frac{TN}{TN + FP}$$

(17)

**Recall_{hot}** is the ratio of correctly predicted hot questions over all actually hot questions. **Recall_{cold}** is the ratio of correctly predicted cold questions over all actually cold questions.

The F1-score takes Precision and Recall into account. The formula for calculating the F1-score for positive samples is shown below. The method is more effective when F1 score is higher.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

(18)
In this study, we answer the following research questions.

5.3.1 | RQ1: How effective is VSAF in hot question prediction?

In order to evaluate the efficiency of our approach VSAF, we compare it with state-of-the-art approach [1]. Devaraj Phukan et al. proposed a method to predict the final popularity of a question using only the textual features of the question, including the title, body and tags of the question [1]. First convert all these textual features to lowercase letters, then remove the URL, stop words, etc., and perform stemming and other processing. Then the term frequency-inverse document frequency (TF-IDF) vectorizer is used to convert the processed question's title, body and tags into word vectors. Then Phukan et al. assign different weights to various vectors. Weights of title's word vectors, body's word vectors and tags' word vectors are 0.5, 0.2 and 0.3, respectively. Next, these vectors are combined to form the question's word vectors. Finally, they input these word vectors into the Linear Support Vector Machine (LSVM) classifier.

\[
F_{1m} = \frac{2 \times Precision_m \times Recall_m}{Precision_m + Recall_m}, m = \text{hot|cold} \tag{18}
\]

In order to compare the performance of approaches, we define the \textit{gain} to compare how the approach \( j \) outperforms the approach \( i \), including \textit{Accuracy} gain, \textit{Precision} gain, \textit{Recall} gain and \textit{F1-score} gain. They are defined as below. Among them, \( m \) stands for hot or cold.

\[
gain(\text{Accuracy}) = \frac{\text{Accuracy}(i) - \text{Accuracy}(j)}{\text{Accuracy}(j)} \tag{19}
\]

\[
gain(\text{Precision}_m) = \frac{\text{Precision}_m(i) - \text{Precision}_m(j)}{\text{Precision}_m(j)} \tag{20}
\]

\[
gain(\text{Recall}_m) = \frac{\text{Recall}_m(i) - \text{Recall}_m(j)}{\text{Recall}_m(j)} \tag{21}
\]

\[
gain(\text{F1}_m) = \frac{\text{F1}_m(i) - \text{F1}_m(j)}{\text{F1}_m(j)} \tag{22}
\]

5.3.2 | RQ2: Is it useful to use textual features in predicting hot questions?

Previous work [1] considered textual features in hot question prediction. We wonder whether textual features are useful in hot question prediction. Therefore, we combine both textual features and initial_feedback features together, and explore the prediction performance.

\[
\left| \text{Precision}_m \left( \frac{\text{Precision}_m(i) - \text{Precision}_m(j)}{\text{Precision}_m(j)} \right) \right| \quad \left| \text{Recall}_m \left( \frac{\text{Recall}_m(i) - \text{Recall}_m(j)}{\text{Recall}_m(j)} \right) \right| \quad \left| \text{F1}_m \left( \frac{\text{F1}_m(i) - \text{F1}_m(j)}{\text{F1}_m(j)} \right) \right|
\]

5.3.3 | RQ3: What are benefits of attribute combination in hot question prediction?

Our method VSAF combines view amount changes, score changes and answer amount changes soon after questions’ creation. We wonder whether all these attributes are necessary in hot question prediction. We compare proposed VSAF method with approaches based on parts of attributes.

5.3.4 | RQ4: How does the observation duration \( K \) affect the prediction results?

We use the initial_feedback features within \( K \) hour(s) of question's posting to predict whether the question is hot or not. \( K \) is defined in Section 4. It represents the observation duration of questions. The task of this RQ is to explore the influence of the setting of \( K \) on the prediction effect of hot questions.

5.3.5 | RQ5: How effective is the hot question prediction using fully convolutional neural network?

This article completes prediction task based on initial_feedback features. Recurrent Neural Network (RNN) is often used for classification tasks based on initial_feedback features [33]. Therefore, we compare the prediction effect using fully convolutional neural network with using RNN and Long Short-Term Memory (LSTM), k-Nearest Neighbor (KNN), Decision Tree and Random Forest.

RNN is a neural network that contains recurrence. It takes sequence data as input, and the cyclic units are connected in a chained manner, so it is very suitable for processing serialised data such as speech, text and time series. Because of this chain structure, the hidden layer in front will affect the hidden layer in the back. LSTM is a variant of RNN structure, which can solve the problem of gradient disappearance and gradient
explosion during long sequence training. LSTM can remember the information that needs to be memorised for a long time, and forget the unimportant information. The idea of the KNN algorithm is that if most of the $k$ nearest samples in the feature space of a sample belongs to a certain category, the sample is also divided into this category. The decision tree algorithm uses the structure of the tree to classify data records. The leaf nodes of the tree represent category labels, and the internal nodes represent a certain feature variable. According to the feature variable selection method, the branches of the tree are established according to the different values of the feature variables, and the lower nodes and branches are repeatedly established in each branch subset, and finally a decision tree can be generated. The random forest algorithm uses the resampling method to extract multiple samples from all samples, construct a decision tree for each sample set which is extracted, and finally combine the prediction results of multiple decision trees to obtain the final prediction conclusion by voting.

5.4 | Results

5.4.1 | RQ1: How effective is VSAF in hot question prediction?

We contrast our work with the baseline approach [1]. In order to answer RQ1, the evaluation results of these two methods using balanced test set and random test set are shown in Tables 2 and 3, respectively. In Table 2, VSAF achieves $\text{Accuracy}$, $\text{Precision}_\text{hot}$, $\text{Recall}_\text{hot}$, $F_1_\text{hot}$, $\text{Precision}_\text{cold}$, $\text{Recall}_\text{cold}$ and $F_1_\text{cold}$ of 80%, 87.5%, 70%, 77.77%, 75%, 90% and 81.81%. In Table 3, VSAF achieves $\text{Accuracy}$, $\text{Precision}_\text{hot}$, $\text{Recall}_\text{hot}$, $F_1_\text{hot}$, $\text{Precision}_\text{cold}$, $\text{Recall}_\text{cold}$ and $F_1_\text{cold}$ of 84.91%, 42.37%, 74.28%, 53.96%, 96.13%, 86.34% and 90.97%. We can observe that the VSAF method performs well on both test sets. According to the prediction accuracy of these two methods on different test sets, we make the histogram as shown in Figure 4.

In order to compare VSAF with baseline approach, we compute $\text{Accuracy}$ gains, $\text{Precision}$ gains, $\text{Recall}$ gains and $F_1 - \text{score}$ gains and describe results in Tables 4 and 5. The experimental results show, using balanced test set for evaluation, VSAF outperforms baseline approach by 25.59%, 38.01%, 5.36%, 21.52%, 17.37%, 43.31% and 29.04% in terms of $\text{Accuracy}$, $\text{Precision}_\text{hot}$, $\text{Recall}_\text{hot}$, $F_1_\text{hot}$, $\text{Precision}_\text{cold}$, $\text{Recall}_\text{cold}$ and $F_1_\text{cold}$ respectively. Besides, using random test set for evaluation, VSAF outperforms baseline approach by 31.83%, 120.79%, 19.92%, 84.16%, 3.77%, 33.57% and 19.35% in terms of $\text{Accuracy}$, $\text{Precision}_\text{hot}$, $\text{Recall}_\text{hot}$, $F_1_\text{hot}$, $\text{Precision}_\text{cold}$, $\text{Recall}_\text{cold}$ and $F_1_\text{cold}$ respectively. We can see that the VSAF method performs better than the baseline method on both test sets.

**RQ 1** VSAF achieves higher $\text{Accuracy}$, $\text{Precision}_\text{hot}$, $\text{Recall}_\text{hot}$, $F_1_\text{hot}$, $\text{Precision}_\text{cold}$, $\text{Recall}_\text{cold}$ and $F_1_\text{cold}$ than the baseline approach.

5.4.2 | RQ2: Is it useful to use textual features in predicting hot questions?

According to previous work [1], we extract textual context of questions, including their titles, bodies, and tags. We use Natural Language Toolkit (NLTK) to remove punctuation, stop words and convert textual context to textual features in word vectors. In Section 4, we define initial_feedback features, including view amount changes, score changes and answer amount changes. We study prediction performance based on three method, including the prediction based on textual features, the prediction based on initial_feedback features (namely our method VSAF), and the prediction based on textual and initial_feedback features.

The experimental results are shown in Tables 6 and 7. As can be seen from these tables, we can find that the approach using all features has better performance than the approach using only text features. Besides, the best $\text{Accuracy}$, $F_1_\text{hot}$ and $F_1_\text{cold}$ are achieved on both test sets when only initial_feedback features are analysed. Therefore, we can conclude that the hot question prediction based on initial_feedback features without text features has the best performance on both test sets. According to the prediction accuracy of the methods based on different features on different test sets, we make the histogram as shown in Figure 5.

**RQ 2** The hot question prediction only based on initial_feedback features has the best performance, without textual features.

5.4.3 | RQ3: What are benefits of attribute combination in hot question prediction?

In order to answer RQ3, we use fully convolutional neural network to separately analyse view amount changes, score changes and answer amount changes for hot question prediction. We compare performance based on different attributes, and the experimental results are shown in Tables 8 and 9. From these tables, we find that hot question prediction based on view amount changes achieves better performance than based on score changes or answer amount changes. Besides, as can be seen from the above tables, the approach using all attributes has better performance than the approach using view amount changes. The best $\text{Accuracy}$, $F_1_\text{hot}$ and $F_1_\text{cold}$ are achieved on both test sets when analysing all attributes. Therefore, it is conclusion that the approach based on all attributes performs best on both test sets. According to the prediction accuracy of the methods based on different attributes on different test sets, we make the histogram as shown in Figure 6.

**RQ 3** The combination of view amount changes, score changes and answer amount changes is effective for hot question prediction.
### Table 2: Results comparison with baseline approach (based on balanced test set)

| Method          | Accuracy | Precision<sub>hot</sub> | Recall<sub>hot</sub> | F<sub>1</sub>hot | Precision<sub>cold</sub> | Recall<sub>cold</sub> | F<sub>1</sub>cold |
|-----------------|----------|-------------------------|----------------------|------------------|-------------------------|----------------------|------------------|
| Baseline approach | 63.7%   | 63.4%                  | 64.6%               | 64%             | 63.9%                  | 62.8%               | 63.4%            |
| VSAF            | 80%      | 87.5%                  | 70%                  | 77.77%          | 75%                    | 90%                  | 81.81%           |

### Table 3: Results comparison with baseline approach (based on random test set)

| Method          | Accuracy | Precision<sub>hot</sub> | Recall<sub>hot</sub> | F<sub>1</sub>hot | Precision<sub>cold</sub> | Recall<sub>cold</sub> | F<sub>1</sub>cold |
|-----------------|----------|-------------------------|----------------------|------------------|-------------------------|----------------------|------------------|
| Baseline approach | 64.41%  | 19.19%                  | 61.94%               | 29.3%           | 92.64%                  | 64.74%               | 76.22%           |
| VSAF            | 84.91%   | 42.37%                  | 74.28%               | 53.96%          | 96.13%                  | 86.34%               | 90.97%           |

### Table 4: Gains of baseline approach and VSAF (in comparison with VSAF approach based on balanced test set)

| Gain(Accuracy)   | Gain(Precision<sub>hot</sub>) | Gain(Recall<sub>hot</sub>) | Gain(F<sub>1</sub>hot) | Gain(Precision<sub>cold</sub>) | Gain(Recall<sub>cold</sub>) | Gain(F<sub>1</sub>cold) |
|------------------|-------------------------------|-----------------------------|-------------------------|-------------------------------|-------------------------------|-------------------------|
| 25.59%           | 38.01%                        | 5.36%                       | 21.52%                  | 17.37%                        | 43.31%                       | 29.04%                  |

### Figure 4: Comparison of accuracy with baseline method
5.4.4 | RQ4: How does the observation duration $K$ affect the prediction effect?

In this section, we explore the effect of observation duration on prediction effects. We study how the prediction effect changes as observation duration $K$ ($K = 1, 3, 5, 7, 9$) changes. The results are shown in Tables 10 and 11. From the above tables, we find that as the observation duration increases, the Accuracy, $F_1_{hot}$ and $F_1_{cold}$ of our method VSAF are generally increasing on both test sets.

Based on the above tables, we draw Figure 7. Figure 7 shows the changes of the Accuracy with the increase of observation duration ($K$) on the balanced test set and the random test set. As can be seen from the above figure, on both test sets, as the observation duration increases, the Accuracy generally increases. In addition, the Accuracy increases the most when the observation duration change from 1 to 3 hours. Since then, the increase rate of Accuracy has slowed down. In this article, we set observation duration as 3 hours by default. In practice, administrators can set the observation duration of questions according to their actual needs.

**RQ 4** As the observation duration increases, the Accuracy of our method VSAF generally increases on both test sets.

5.4.5 | RQ5: How effective is the hot question prediction using fully convolutional neural network?

To answer RQ5, we try a variety of classification algorithms, including RNN, LSTM, KNN, decision tree and random forest. We study whether the prediction effect of using fully convolutional neural network is better than using other proposed algorithms. The experimental results are shown in Tables 12 and 13. Precision and Recall sometimes contradict each other, $F_1$-score comprehensively measures the values of Precision and Recall. Besides, Accuracy can intuitively reflect the prediction effect. Therefore, the most important metrics in this article are Accuracy, $F_1_{hot}$ and $F_1_{cold}$.

Note: Results based on initial_feedback features are better than results based on other features, and thus results based on initial_feedback features are in bold.

**RQ 5** The prediction effect using fully convolutional neural network is better than using RNN, LSTM, KNN, Decision Tree and Random Forest.

6 | Threats to Validity

In this section, we discuss threats to external and construct validity.
FIGURE 5  Comparison of accuracy of methods based on different features

TABLE 8  Performance with different attribute (based on balanced test set)

| Attribute                  | Accuracy | Precision<sub>hot</sub> | Recall<sub>hot</sub> | F<sub>1</sub><sub>hot</sub> | Precision<sub>cold</sub> | Recall<sub>cold</sub> | F<sub>cold</sub> |
|----------------------------|----------|------------------------|---------------------|------------------|-----------------------|---------------------|-----------------|
| View amount changes        | 78.63%   | 84.8%                  | 69.75%              | 76.54%           | 74.31%                | 87.5%              | 80.37%          |
| Score changes              | 56.75%   | 66.27%                 | 27.5%               | 38.87%           | 54.26%                | 86%                | 66.54%          |
| Answer amount changes      | 64%      | 73.53%                 | 43.75%              | 54.86%           | 59.96%                | 84.25%             | 70.06%          |
| All                        | 80%      | 87.5%                  | 70%                 | 77.77%           | 75%                   | 90%                | 81.81%          |

Note: Results based on initial_feedback features are better than results based on other features, and thus results based on initial_feedback features are in bold.

TABLE 9  Performance with different attribute (based on random test set)

| Attribute                  | Accuracy | Precision<sub>hot</sub> | Recall<sub>hot</sub> | F<sub>1</sub><sub>hot</sub> | Precision<sub>cold</sub> | Recall<sub>cold</sub> | F<sub>cold</sub> |
|----------------------------|----------|------------------------|---------------------|------------------|-----------------------|---------------------|-----------------|
| View amount changes        | 83.41%   | 39.67%                 | 75.59%              | 52.03%           | 96.24%                | 84.46%             | 89.97%          |
| Score changes              | 70.31%   | 18.84%                 | 45.14%              | 26.58%           | 90.86%                | 73.71%             | 81.39%          |
| Answer amount changes      | 79.44%   | 27.41%                 | 44.09%              | 33.8%            | 91.77%                | 84.21%             | 87.83%          |
| All                        | 84.91%   | 42.37%                 | 74.28%              | 53.96%           | 96.13%                | 86.34%             | 90.97%          |

Note: Results based on initial_feedback features are better than results based on other features, and thus results based on initial_feedback features are in bold.
FIGURE 6  Comparison of accuracy of methods based on different attributes

| K (hour) | Accuracy  | Precision$_{hot}$ | Recall$_{hot}$ | F$_1$$_{hot}$ | Precision$_{cold}$ | Recall$_{cold}$ | F$_1$$_{cold}$ |
|---------|-----------|--------------------|----------------|--------------|--------------------|----------------|--------------|
| 1       | 73.25%    | 81.2%              | 60.5%          | 69.34%       | 80.00%             | 64.00%         | 56.75%       |
| 3       | 80%       | 87.5%              | 70%            | 77.77%       | 86%                | 88.27%         | 81.81%       |
| 5       | 82%       | 90.5%              | 71.5%          | 79.89%       | 90%                | 92.5%          | 83.71%       |
| 7       | 83.25%    | 86.94%             | 78.25%         | 82.36%       | 88.25%             | 88.25%         | 84.04%       |
| 9       | 84%       | 84.51%             | 83.25%         | 83.87%       | 84.75%             | 84.11%         | 84.11%       |

TABLE 10  Impact of observation duration on performance (based on balanced test set)

| K (hour) | Accuracy  | Precision$_{hot}$ | Recall$_{hot}$ | F$_1$$_{hot}$ | Precision$_{cold}$ | Recall$_{cold}$ | F$_1$$_{cold}$ |
|---------|-----------|--------------------|----------------|--------------|--------------------|----------------|--------------|
| 1       | 80.66%    | 33.96%             | 66.14%         | 44.88%       | 94.75%             | 82.62%         | 88.27%       |
| 3       | 84.91%    | 42.37%             | 74.28%         | 53.96%       | 96.13%             | 86.34%         | 90.97%       |
| 5       | 86.13%    | 45.04%             | 74.54%         | 56.13%       | 96.22%             | 87.69%         | 91.76%       |
| 7       | 87.94%    | 49.56%             | 74.54%         | 59.54%       | 96.31%             | 89.75%         | 92.91%       |
| 9       | 91.13%    | 60.21%             | 75.07%         | 66.82%       | 96.51%             | 93.3%          | 94.88%       |

TABLE 11  Impact of observation duration on performance (based on random test set)
FIGURE 7 Changes in Accuracy as observation duration (K) increases

TABLE 12 Comparison with other classification algorithms (based on balanced test set)

| Classification algorithm | Accuracy | Precision<sub>hot</sub> | Recall<sub>hot</sub> | F<sub>1</sub>hot | Precision<sub>cold</sub> | Recall<sub>cold</sub> | F<sub>1</sub>cold |
|--------------------------|----------|------------------------|------------------|--------------|------------------------|------------------|--------------|
| FCN                      | 80%      | 87.5%                  | 70%              | 77.77%       | 75%                    | 90%              | 81.81%       |
| LSTM                     | 76.63%   | 74.26%                 | 81.5%            | 77.71%       | 79.5%                  | 71.75%           | 75.43%       |
| RNN                      | 70.5%    | 65.07%                 | 88.5%            | 75%          | 82.03%                 | 52.5%            | 64.02%       |
| KNN                      | 75.63%   | 79.04%                 | 69.75%           | 74.1%        | 72.93%                 | 81.5%            | 76.98%       |
| Decision tree            | 71%      | 72.95%                 | 66.75%           | 69.71%       | 69.35%                 | 75.25%           | 72.18%       |
| Random forest            | 76.5%    | 78.8%                  | 72.5%            | 75.52%       | 74.54%                 | 80.5%            | 77.4%        |

Note: Results based on initial_feedback features are better than results based on other features, and thus results based on initial_feedback features are in bold.

TABLE 13 Comparison with other classification algorithms (based on random test set)

| Classification Algorithm | Accuracy | Precision<sub>hot</sub> | Recall<sub>hot</sub> | F<sub>1</sub>hot | Precision<sub>cold</sub> | Recall<sub>cold</sub> | F<sub>1</sub>cold |
|--------------------------|----------|------------------------|------------------|--------------|------------------------|------------------|--------------|
| FCN                      | 84.91%   | 42.37%                 | 74.28%           | 53.96%       | 96.13%                 | 86.34%           | 90.97%       |
| LSTM                     | 73.19%   | 28.42%                 | 82.41%           | 42.26%       | 96.8%                  | 71.94%           | 82.54%       |
| RNN                      | 60.13%   | 21.44%                 | 88.19%           | 34.5%        | 97.24%                 | 56.33%           | 71.34%       |
| KNN                      | 80%      | 33.71%                 | 70.34%           | 45.58%       | 95.3%                 | 81.31%           | 87.75%       |
| Decision tree            | 76.5%    | 28.99%                 | 67.19%           | 40.51%       | 94.61%                 | 77.76%           | 85.36%       |
| Random forest            | 79.09%   | 31.73%                 | 65.62%           | 42.77%       | 94.57%                 | 80.92%           | 87.21%       |

Note: Results based on initial_feedback features are better than results based on other features, and thus results based on initial_feedback features are in bold.

6.1 | External validity

Threats to external validity relate to generalization of our study. Our experimental results are based on Stack Overflow and it is not clear whether they are applicable to other question and answer communities. In future, we plan to apply our method to other question and answer communities and discuss its effectiveness on other communities.

6.2 | Construct validity

Threats to construct validity refers to the suitability of the evaluation metrics we use. We use Accuracy, Precision, Recall and F1-score, which are also used by previous works to evaluate effectiveness of popular prediction methods [1,11]. Therefore, we believe there is little threat to construct validity.
7 | FUTURE APPLICATION SCENARIOS

The usage scenarios of our proposed method are as follows:

7.1 | Without method

First, Bob is an experienced expert who can answer many software related questions. Before applying our proposed method to Stack Overflow, Bob answers some newly posted questions. However, these questions are only concerned by a small number of users, and more users’ concerns have not been answered. A large number of users do not get answers to questions on Stack Overflow in time. Second, Jerry is an employee of Stack Overflow website who is responsible for planning advertising activities. Different advertisers are willing to pay different prices for advertisements on Stack Overflow. It is difficult for Jerry to decide how to put advertisements into corresponding questions.

7.2 | With method

Users and the employees of Stack Overflow website can get help after applying our proposed method to Stack Overflow. First, Bob can easily view the newly posted predicted hot questions and answer some of them. Users who concern these hot questions are answered in time, and Bob therefore helps more users. Second, Jerry can also make corresponding plans for advertising according to whether the question is hot or not. More specifically, Jerry puts high-bid advertisements into predicted hot questions so that these advertisements get higher page views, and puts low-bid advertisements into predicted cold questions so that these advertisements get fewer page views.
A method to predict hot questions in Stack Overflow is studied here. According to previous works [1,11], we use the number of views to characterise whether the question is hot or not. We propose the VSAF method which uses a fully convolutional neural network to analyse the view amount changes, answer amount changes and score changes soon after questions’ creation. We evaluate the performance of VSAF method based on a training set and two different test sets. The experimental results show that the hot question prediction only based on initial_feedback features without textual features has the best performance. In addition, when we use the balanced test set, the Accuracy of the VSAF method is 80%, which outperforms the baseline approach by 25.59%. Based on the random test set for evaluation, the Accuracy of the VSAF method is 84.91%, which outperforms the baseline approach by 31.83%. Therefore, we believe that the VSAF method is useful in predicting hot questions in Stack Overflow.

However, there are still deficiencies in our research work. First, we did not analyse all the factors that may affect the popularity of the question. For example, we did not consider the topic characteristics of the question. The number of users following a topic may be more than the number of users following another topic. In future, we plan to extract more potential influencing factors and perform in-depth and comprehensive explanatory research to analyse which factors have a greater impact on the predicted results. For example, we will add the topic analysis research of the question and explore the impact of adding topic analysis on the predicted results. Second, our experimental results are based on Stack Overflow and it is not clear whether they are applicable to other question and answer communities. In future, we plan to apply our method to other question and answer communities and discuss its effectiveness on other communities.

ACKNOWLEDGEMENT

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1004202, in part by the National Natural Science Foundation of China under Grant 61732019, in part by the State Key Laboratory of Software Development Environment under Grant SKLSDE-2019ZX-05, and in part by Fundamental Research Funds for the Central Universities under Grant No. YWF-20-BJ-J-1018.

REFERENCES

1. Phukan, D., Singha, A.K.: Feasibility analysis for popularity prediction of stack exchange posts based on its initial content. 2016 3rd International Conference on Computing for Sustainable Global Development (2016)
2. Wu, Y., et al.: How do developers utilize source code from stack overflow. Empirical Sowfw. Eng. 24(2), 637–673 (2019)
3. Abasanuzzaman, A.M.R.C.K.M., Schneider, K.A.: Classifying stack overflow posts on API issues. In: 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 244–254 (2018)
4. Tatar, A., et al.: Ranking news articles based on popularity prediction. In: 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 106–110 (2012)
5. Lerman, K., Hogg, T.: Using a model of social dynamics to predict popularity of news. In: Proceedings of the 19th international conference on World wide web, pp. 621–630 (2010)
6. Lee, J.G., Moon, S., Salamatian, K.: An approach to model and predict the popularity of online contents with explanatory factors. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 1, pp. 623–630 (2010)
7. Shen, H., et al.: Modeling and predicting popularity dynamics via reinforced Poisson processes. Twenty-eighth AAAI conference on artificial intelligence (2014)
8. Cao, Q., et al.: Deephawkles: Bridging the gap between prediction and understanding of information cascades. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1149–1158 (2017)
9. Gou, C., et al.: Learning sequential features for cascade outbreak prediction. Knowl. Inf. Syst., 57(3), 721–739 (2018)
10. Piotrowicz, A., et al.: Headlines matter: using headlines to predict the popularity of news articles on Twitter and Facebook. In: Eleventh International AAAI Conference on Web and Social Media. pp. 656–659 (2017)
11. Liao, D., et al.: Popularity prediction on online articles with deep fusion of temporal process and content features. Proceedings of the AAAI Conference on Artificial Intelligence (2019)
12. Zhang, X., et al.: Event detection and popularity prediction in microblogging. Neurocomputing, 149, 1469–1480 (2015)
13. Pan, S., et al.: Tweet popularity prediction based on propagation simulation. J. Chin. Inform. 28(3), 68–74 (2014)
14. Zhang, W., et al.: User-guided hierarchical attention network for multimodal social image popularity prediction. In: Proceedings of the 2018 World Wide Web Conference, pp. 1277–1286 (2018)
15. Sanjo, S., Katsurai, M.: Recipe popularity prediction with deep visual-semantic fusion. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 2279–2282 (2017)
16. Zhou, J., et al.: Bounties on technical q&a sites: a case study of stack overflow bounties. Emp. Sowfw. Eng. 25(1), 139–177 (2020)
17. Partachi, P.P., et al.: Posit: simultaneously tagging natural and programming languages. (ICSE) International Conference on Software Engineering (2019)
18. Zhang, H., et al.: An empirical study of obsolete answers on stack overflow. IEEE Trans. Sowfw. Eng. 25, 557–572 (2019)
19. Linarevsazquez, M., et al.: How do api changes trigger stack overflow discussions? A study on the android SDK. In: Proceedings of the 22nd International Conference on Program Comprehension (ICPC), pp. 83–94 (2014)
20. Robbins, R., Lungeu, M., Rothlisberger, D.: How do developers react to api deprecation? The case of a smalltalk ecosystem. International Symposium on the Foundations of Software Engineering (2012)
21. Azad, S., Rigby, P.C., Guertou, L.: Generating api call rules from version history and stack overflow posts. ACM Trans. Sowfw. Eng. Methodol. 25(4), 29 (2017)
22. Ren, X., et al.: Discovering, explaining and summarizing controversial discussions in community Q&A sites. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 151–162 (2019)
23. Shah, C., Pomerantz, J.: Evaluating and predicting answer quality in community QA. International ACM Sigir Conference on Research & Development in Information Retrieval (2010)
24. Arora, P., Ganguly, D., Jones, G.J.E.: The good, the bad and their kins: Identifying questions with negative scores in stack overflow. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015 (ASONAM), pp. 1232–1239 (2015)
25. Yao, Y., et al.: Detecting high-quality posts in community question answering sites. Inf. Sci. 302, 70–82 (2015)
26. Chatterjee, P., Kong, M., Pollock, L.: Finding help with programming errors: an exploratory study of novice software engineers focus in stack overflow posts. J. Syst. Softw. 159, 1–13 (2020)

27. Zhang, H., et al.: Reading answers on stack overflow: not enough! IEEE Trans. Softw. Eng. 115 (2019)

28. Shi, Z., Zhao, Q.: Object detection algorithm based on fully convolutional neural network. Comput. Technol. Dev. 28(5), 55–58 (2018)

29. Ravi, S., Pang, B., Rastogi, V., et al.: Great question! question quality in community q&a. Eighth International AAAI Conference on Weblogs and Social Media (2014)

30. Neshati, M.: On early detection of high voted q&a on stack overflow. Inf. Proc. Manag. 53(4), 780–798 (2017)

31. Bing, Y.: Research and application of image segmentation algorithm based on fully convolutional networks. University of Electronic Science and Technology of China (2018)

32. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440 (2015)

33. Du, N., et al.: Recurrent marked temporal point processes: Embedding event history to vector. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1555–1564 (2016)

How to cite this article: Zhao LX, Zhang L, Jiang J. Hot question prediction in Stack Overflow. IET Softi. 2021;15:90–106. https://doi.org/10.1049/sfw2.12013