Evaluating Simple and Complex Models' Performance When Predicting Accepted Answers on Stack Overflow

Osayande P. Omondiagbe, Sherlock A. Licorish and Stephen G. Macdonell

August 3, 2022

Evaluating Simple and Complex Models' Performance When Predicting Accepted Answers on Stack Overflow

Osayande P. Omondiagbe[*†], Sherlock A. Licorish[†], and Stephen G. Macdonell[†‡]

[*] Department of Informatics
Landcare Research, Lincoln, New Zealand
omondiagbep@landcareresearch.co.nz

[†] Department of Information Science
University of Otago, Dunedin, New Zealand
sherlock.licorish@otago.ac.nz, stephen.macdonell@otago.ac.nz
Software Engineering Research Lab
[‡] Auckland University of Technology, Auckland, New Zealand
stephen.macdonell@aut.ac.nz

Abstract— Stack Overflow is used to solve programming issues during software development. Research efforts have looked to identify relevant content on this platform. In particular, researchers have proposed various modelling techniques to predict acceptable Stack Overflow answers. Less interest, however, has been dedicated to examining the performance and quality of typically used modelling methods with respect to the model and feature complexity. Such insights could be of practical significance to the many practitioners who develop models for Stack Overflow. This study examines the performance and quality of two modelling methods, of varying degree of complexity, used for predicting Java and JavaScript acceptable answers on Stack Overflow. Our dataset comprised 249,588 posts drawn from years 2014–2016. Outcomes reveal significant differences in models' performances and quality given the type of features and complexity of models used. Researchers examining model performance and quality and feature complexity may leverage these findings in selecting suitable modelling approaches for Q&A prediction.

Keywords— Feature Selection; Modelling and Prediction; Neural Network; Random Forest; Stack Overflow

I. Introduction

Stack Overflow is one of the most popular question and answer (Q&A) portals used regularly by software developers [1]. Studies have shown that most questions that are asked on Stack Overflow receive an answer [2]; hence, developers turn to this portal to solve programming-specific issues during software development [3]. The answer that is accepted by the user who posted the question is usually regarded as accepted (or acceptable). This mechanism satisfies the user who created the post and credits the answer provider. The foregoing process makes it appropriate for others searching for similar help to locate solutions that have been accepted by others. Despite Stack Overflow being popular among software engineers, at times accepted answers are delayed [4]. This delay could increase the time it takes software developers to investigate and assess a working solution when using the platform. To reduce such issues researchers have proposed various modelling techniques to predict acceptable answers on Stack Overflow, and on Q&A portals more generally [5, 6]. The outcome of such modelling techniques could be useful for software developers focused on developing plugins that integrate with IDEs that display Stack Overflow Q&A pairs.

The utility of such modelling techniques is influenced by the type of features (textual and non-textual) that are available. For instance, past studies have established that specific non-textual features affect Q&A portals' answer quality [7]. Non-textual features here refer to those that are not textual in nature (e.g., answer score, answer count and response time). Other works have considered only textual features (e.g., text polarity and length of answer) when exploring the quality of content in Q&A platforms [8]. Textual and non-textual features have also been combined in other studies to determine the quality of answers to questions in a Q&A forum. For instance, Blooma et al. [9] combined both textual and non-textual features in predicting answer quality and concluded that both types of features predicted answer quality. However, these authors noted that, in their study, the textual features had a greater influence on the quality of answers than the non-textual features.

In our preliminary work, we found that specific Q&A features can aid in distinguishing answer acceptability (e.g., length of code in answers and reputation of users) [10]. However, this work did not evaluate features' or models' complexity. Considering the conduct and outcomes of this and earlier work, little is known about whether the specific features selected may have been the driver of the model outcomes when predicting answer acceptability, or if the complexity of the model could have been the driver in improving prediction outcomes. Beyond the provision of knowledge around the use of different modelling approaches and the prediction value of different features, such insights could have implications for the quality of software generally, as Stack Overflow is used extensively by developers [3]. It would be undesirable if poor-quality answers are used to inform software development. Therein exists the opportunity to investigate the performance and quality of models' complexity when considering a range of feature complexity.

We investigate the performance and quality of two models and a range of feature types using a Stack Overflow dataset. In this paper, we used the following terms: **(1) Model Quality:** which refers to the fit of the model and not its correctness (i.e., ability to perform false-negative and -positive classification); **(2) Model Performance:** which refers to the number of correctly classified data instances (i.e., accuracy); and **(3) Model Complexity:** refers to the computational complexity and the number of parameters in the predictive model. Beyond these measures, we validated our outcome via a questionnaire completed by a sample of software developers in New Zealand. We believe that our outcomes contribute to the body of knowledge aimed at understanding the performance and quality of models with varying degrees of complexity when used in Q&A settings.

The rest of this paper is arranged as follows: Section II examines related work, and Section III presents our study design. Section IV provides our results and we discuss our outcomes and explore implications in Section V, before considering threats to the work in Section VI. Section VII concludes the paper and considers future research.

II. Related Work

A. *Evaluating Features and Models*

Most studies that have built models to predict acceptable answers in Q&A settings focus on the semantic relevance of features (e.g., [11, 12]). These features are usually grouped as textual or non-textual and are used as input to various models to predict answer acceptability. Here we regard the answer that is selected by the user who posed the question as the most acceptable from the list of answers provided by other users. Beyond consideration of textual features [13] and non-textual features [14], a small body of research has also considered combining textual and non-textual features [14, 15] for building predictive models to select the best answers on Q&A portals. The feature extraction methods used are often grouped into the hand-crafted [16] approach (extracting features manually) or a deep learning approach (without feature engineering or specialist linguistic data).

Typically, modelling techniques are applied individually, rather than providing a comparative evaluation of approaches. A comparative study done by Calefato et al. [17] revealed that the choice of a model and its associated parameters affected the prediction accuracy when determining the best answers. The need to increase the range of features used for training models has led researchers to combine both textual and non-textual features. For instance, the work of Blooma et al. [9] combined textual and non-textual features to predict the best answer in a stack of answers by using a Bayesian model, concluding that best answers were most influenced by textual features. Jizhou et al. [13] combined structural information with textual and non-textual features to extract high-quality pairs in discussion threads from an online discussion forum using a support vector machine. Using the same method, Buse and Weimer [18] also combined textual and non-textual features to predict the best answers in a Yahoo Q&A dataset.

Previous studies have found that by using the correct features the quality of models is improved [19], so it becomes important for a model to be able to determine the correlation between the different types of features found during Q&A data modelling (such as Stack Overflow). However, it is difficult at times for some models to determine the correlation between textual and non-textual features [20]. This has led some researchers to use deep neural networks as a technique to enhance model outcomes. The intent here is to construct new valuable features automatically, eliminating the task of manually selecting the important features to use for training. For instance, Wang et al. [21] applied a deep belief network in modelling the semantic relevance of Q&A pairs, while Lei et al. [15] used a convolutional network to learn a distributional sentence model for Q&A prediction using a bag-of-words approach and bigram-based word representations. Gao et al. [22] proposed a three-stage deep neural network–based approach to identify the most relevant answer among a set of answer candidates. Suggu et al. [14] examined deep features using convolutional networks and then combined these features with hand-crafted features (textual and non-textual) to determine the quality of answers. Their outcomes revealed enhanced performance over other works [23], which may be attributed to fusing deep learning and hand-crafted features.

B. *Features' and Models' Behaviours*

Reviewing the studies above, it is not clear under which modelling circumstances complex model such as deep learning and/or approaches that promote the use of hand-crafted features will outperform a less complex model which uses only hand-crafted features. The choice of the model and its associated parameters affects the model outcomes. On the premise that developers regularly use Stack Overflow to solve problems during software development [1, 2], in our preliminary work we studied the features that distinguish acceptable answers on this portal [10]. While we found specific Q&A features to predict answer acceptability (e.g., length of code in answers and reputation of users), questions remain around *how* the types of Q&A features affect the behaviour of various models that are used to predict acceptable answers in a Q&A setting.

Zou et al. [24] noted that this is an area in need of additional research. These authors used Stack Overflow data in a preliminary study examining the impact of the feature weighting method and feature set on a classifier's performance, where outcomes varied depending on those factors. However, this study only examined a single classifier (Bayesian logistic regression). The performance of a model should not be judged only on the availability of certain features, and evaluating a single classifier does not provide insights around variances in model performance [25]. It is therefore important to understand how varying features influence models' performance, and how varying the complexity of models in relation to features affects prediction outcomes. This insight could lead to both informed model choice and feature selection.

III. Study Design

A. Research Questions

The purpose of this paper is to investigate the performance and quality of models' complexity when considering a range of features. We performed analysis using two different models of varying complexities, guided by the following research questions:

RQ1: *How does the degree of model and feature complexity affect the* ***performance*** *of a Stack Overflow Q&A accepted answer prediction model?*

Motivation: The benefit of accepted answer prediction is important for Q&A platforms as at times this platform lacks the features for marking an answer as accepted when there is no accepted answer. This feature could help to move an accepted answer to the top of the Q&A thread, thus potentially saving developers the time and effort of investigating and assessing a working solution. Accordingly, we investigate how the different levels of feature and model complexities can affect the performance of an accepted answer prediction model. Software developers developing Stack Overflow plugins can leverage the outcome of this RQ for marking an answer as accepted when there is no accepted answer in the Q&A thread retrieved from Stack Overflow. Stack Overflow is typically used by developers during software development [1], and there is a need to have reliable accepted answers. For this reason, we define our second research question as follows:

RQ2: *How does the degree of model and feature complexity affect the* ***quality*** *of a Stack Overflow Q&A accepted answer prediction model?*

Motivation: The Stack Overflow platform has a reputation system and the reputation ranking of members may enhance other users' trust for their contributions on the website. This could be problematic for new software developers seeking solutions to their software problems. The benefit of having a quality accepted answer prediction is important for providing a reliable accepted answer. Thus, the need to minimise the false negative and false positive in the prediction model is important. Therefore, we investigate how the different levels of feature and model complexities can affect the quality of an accepted answer prediction model. The findings from this RQ could help improve the quality of accepted answers predicted in Stack Overflow plugins.

B. Dataset Features

To enable comparative analysis with earlier outcomes [10], we used the Stack Overflow dataset that was added on September 12, 2016. We extracted data for the top two tags (JavaScript and Java). We used the selection criteria from our previous study [10] to select the feature sets (see Sections 3 to 4 in our previous study [10] for details). Our final dataset for experimentation consisted of 249,588 posts from 2014, 2015 and 2016, which have at least two answers, one of which is an acceptable answer. We had an imbalanced dataset, where the number of accepted answers was 88,607, while answers provided but not labelled as accepted numbered 160,981. We randomly selected 70% (174,711) of the data for training, and the remaining 30% (74,877) was used for testing when performing our predictive modelling. To address the imbalanced dataset during training, we used the synthetic minority over-sampling "SMOTE" algorithm [27], and the adaptive synthetic sampling "ADASYN" algorithm [28]. The "SMOTE" and "ADASYN" algorithms were chosen because they are widely used when learning from imbalanced datasets. For the text preprocessing step, we follow the steps in Section 4.3 from our previous study [10]. For replication purposes, our dataset is available here: https://tinyurl.com/y7m3k2mk.

C. Feature Selection

1) *Hand-Crafted:* We follow our previously published work [10] and so use the same set of features in this study. These features were grouped into four categories (code, textual, non-textual and user features) as listed below:

- **Code Features**: Number of code line, Code Length (number of identifiers).
- **Non-Textual Features**: Number of comments, Answer count, Answer score (score), View count, Response Time (Timelag).
- **Textual Features**: Question and Answer Similarity (TFAnswerText [1], TFAnswerCode [2]), Text Polarity [3], Textual Similarity [4], Number of Words, Number of sentences, and Url count.
- **User Features**: Reputation, Sign up date-time lag.

To avoid multi-collinearity, we follow the method in our previous work and discard the "NumberOfWords" and "SignupDateTimeLag" features as they were seen to be correlated with the Number of sentences and Reputation features, respectively. This method compared feature pairs with a Pearson's correlation plot and discard feature pairs where the root mean square is <0.7.

2) *Neural Generated Features:* To derive the complex feature which we called "neural-generated feature"; we used a model called Distributed Memory Model of Paragraph (PV-DM) to generate a neural generated weight. This was chosen given the success of the work done by Suggu et al. [14] in predicting answer quality in a Q&A setting. We used the same approach as they did, by using the question and answer as input to a deep model. The model works by taking a question and answer pair as input to learn a good representation of the pair (i.e., it learns how questions and answers are related). The PV-DM model works by remembering the missing context in a paragraph, or the topic of the paragraph and returns a score for

[1] *Similarity between question and text*
[2] *Similarity between question and code*
[3] *Emotion content of answers*
[4] Count of every word that occurred in the question and answer separately

each question and answer pair. This score indicates if the answer is related to the question.

D. *Modelling*

Calefato et al. [17] presented 26 models of varying complexity commonly used in best-answer prediction, where random forest and recurrent neural networks are established to have greater complexity. Given that random forest is seen to have significantly lower complexity than recurrent neural networks [29] and both methods use similar classifier-specific feature importance methods, we anticipate that these two modelling approaches would usefully identify differences in model complexity in this work. Also, these two approaches were seen to be among the best-performing simple (random forest) and complex (RNN) models in a Q&A setting when compared to other types of models in the work of Calefato et al. [17] and a previous study [30]. We use the four categories of features listed in the previous section and a neural network-generated or derived feature. Our models and features are combined in various ways to predict acceptable answers, as described below.

Random Forest Model with Hand-crafted Features: We started our modelling process by using a random forest model to build a classifier to classify answers as either accepted or not accepted (answer acceptability). As noted above, random forest was chosen as a model with lesser complexity as it takes many input variables without the need for replacement [31]. Also, random forest estimates the importance of each variable in the classifier, while using an out-of-bag estimator to estimate the classification error when sampled with replacement [32]. As noted in Section III.A, due to the imbalanced nature of our dataset, we applied the SMOTE and ADASYN algorithms. We used Bayesian optimization techniques [33] to find the best set of hyperparameters to minimise the log loss objective function. The final hyperparameters chosen were:
n_estimators:240, max_features:16, max_depth:15, Boostrap: True, min_samples_split:5 and min_samples_leaf:4

Random Forest Model with Hand-crafted and Neural-generated Features: We repeated the experiment above by adding the complex feature derived from the PV-DM model. The mathematical details of the PV-DM may be probed further by accessing [36].

Recurrent Neural Network (RNN) Model with Hand-crafted Features: In increasing the complexity of our model, we repeated our experiment with an RNN model to classify answers. This model was chosen because RNN is known to capture the compositional aspects of sentences or paragraphs when compared to models with less complexity [34]. Again, we used Bayesian optimization techniques to select the optimal parameters in our RNN, which had 15 hidden layers. We used a stochastic gradient descent (SGD) as our optimizer to train our RNN. SGD was used because it is fast, given that it requires one data point at a time or a mini-batch of data points, which also makes it less memory intensive. Also, SGD is less prone to bad local minima because it converges faster to the local minimum by taking smaller step sizes to refine the network [35]. We also applied the two sampling algorithms (SMOTE and ADASYN) when modelling our RNN (as is the case for the other modelling tasks below). We seeded the random split function to have the same split for each execution of our algorithms. The final hyperparameters chosen for our RNN were 20, 64 and 0.01 for our epoch, batch size and learning rate, respectively.

Recurrent Neural Network (RNN) Model with Hand-crafted and Neural-generated Features: We repeated the experiment above by including the neural-generated feature generated from our PV-DM model.

We executed the above models 100 times in keeping with established recommendations for investigating models' performance and quality (further considered next) [37].

E. *Performance and Quality Measures*

Performance: We computed the balanced accuracy of our models to evaluate their performance. Balanced accuracy measures model performance by taking into account class imbalances, and they also overcome bias in binary cases [38]. The balanced accuracy is computed as the average of the proportion of correct predictions for each class separately.

Quality: To assess the impact of various features in our models we need models that classify accepted answers as accepted (few false negatives) and that also minimise classifying answers that are not accepted as accepted (few false positives). This ultimately determines model quality. The F1-score and Matthews Correlation Coefficient (MCC) are used in this regard. The F1-score is designed to handle data imbalance, and maximizing the F1-score improves model quality [39]. In addition, the F1-score is the harmonic mean of precision and recall, making it relatively precise. The best possible F1-score is 1 (perfect precision and recall), with the worst being 0. The MCC was used as a second measure to evaluate the models' quality because this measure is considered a good metric for assessing quality when using imbalanced datasets [40]. MCC values range from -1 to $+1$; with $+1$ indicating a perfect model, 0 showing that a model is no better than a random prediction, and -1 signaling total disagreement between the prediction and observation. These two measures provide further augmentation for SMOTE and ADASYN in addressing any threats that may result due to our imbalanced dataset.

F. *Evaluation Via Developers Questionnaire*

As a countermeasure to using contributors' acceptance to indicate answer acceptability, and to evaluate how the models perform in terms of performance and quality, a questionnaire was designed to gather experts' opinions about the suitability of answers in our dataset. The questionnaire was anonymous and targeted Java and JavaScript software developers in New Zealand. See a sample of the questionnaire here: https://figshare.com/s/3d9e5a8d49f03a186afb.
The questionnaire was presented as a simple online form asking developers to rank answers based on how likely an answer was to be suitable for a given question. The experts ranked the answers by assigning a value of 1 to 3 in line with the degree to which they believed an answer was suitable for a given question. Here an answer scoring 3 was assessed as highly acceptable, while one given a 2 was assessed as reasonably

acceptable, and an answer assigned 1 was assessed as weakly acceptable. Aware that if the questionnaire took too long to complete, we would not be able to recruit developers, we selected questions from our test sample which had three answers, and each of the answers' length was <750 characters. In populating the questionnaire, 236 questions were randomly selected (of those that had three answers), comprising 150 Java and 86 JavaScript questions.

The questionnaire was sent to known Java and JavaScript developers with industry experience in New Zealand (accessible via software engineering mailing lists). A total of 10 developers participated in our questionnaire. They answered an average of 11.5 questions each, where the highest number of questions answered by a single participant was 15, and the lowest number of questions answered was 9. Over 95% of the questions answered were Java questions, and the average Java experience level for all participants was 4 (on a scale of 1 to 5, where 1 = novice and 5 = expert). Some Java questions were answered by more than one developer, and so we took the average score for these answers. We ended up with a total of 110 Java responses, and 5 JavaScript responses, in response to 90 unique questions.

IV. Results

A. Model Performance

As noted above, we evaluated model performance through the use of the balanced accuracy measure. Here we consider the outcomes of the models in turn.

Random Forest Model with Hand-crafted Features: Outcomes of our modelling revealed balanced accuracy outcomes of 69.89% when sampling was done with ADASYN and 71.74% when sampling was done with SMOTE. We further examined the features to derive their contributed coefficient (Column A in TABLE I). Measurement of this coefficient is based on the extent to which model accuracy decreases when a variable is excluded. The outcome is only provided for SMOTE in TABLE I, given that we recorded better performance (+1.85% gain) when sampling with this algorithm. In Column A it is observed that the time it takes to post an answer (Timelag) and the reputation of the answerer (Reputation) had the largest coefficients in the random forest (Timelag = 0.179 and Reputation = 0.164). It is also shown that the code length (Codelength) and textual similarity between question and answer pairs (TFAnswerText) were noteworthy features in predicting Stack Overflow acceptable answers, with coefficients of 0.143 and 0.153 respectively.

RNN Model with Hand-crafted Features: Using a more complex model (RNN), a balanced accuracy outcome of 62.87% was observed when sampling was done with ADASYN, and an outcome of 65.89% when sampling was done with SMOTE (+3.02% gain). Similar to the random forest outcomes above, we also computed the coefficient for each feature as shown in TABLE I COLUMN B. Again, it is seen that the time lag and reputation of the answerer are the largest coefficients in predicting nswers' acceptance (Timelag = 0.181 and Reputation = 0.172). Overall, there is a reduction in balanced accuracy (performance) of the RNN model when compared to the random forest model (71.74% versus 65.89%). That said, the coefficients of nine features were of higher magnitude in the RNN model (refer to TABLE I). This outcome is plausible, since an answerer with a good reputation is likely to post a reasonable answer. The answerer's attempt to address the question may also result in a substantial amount of code (Codelength) and an answer with context from the question (TFAnswerText), to properly address the question.

TABLE I
Coefficients and Balanced Accuracy of Random Forest and RNN Models for Hand-crafted Features

Feature	A	B
	Random Forest	**Neural Network**
Timelag	0.179	0.181
URLCount	0.032	0.031
CommentCount	0.012	0.021
Reputation	0.164	0.172
TextPolarity	0.054	0.001
AnswerCount	0.012	0.054
ViewCount	0.089	0.043
Score	0.032	0.076
NumberOfcodeLine	0.065	0.054
NumberOfSentence	0.078	0.044
TextualSimilarity	0.031	0.057
Codelength	0.143	0.167
TFAnswerCode	0.024	0.029
TFAnswerText	0.153	0.157
Balanced Accuracy	***71.74***	***65.89***

Random Forest Model with Hand-crafted and Neural-generated Features: To increase the complexity of the features sets, we combined the hand-crafted features and that generated via the PV-DM to train another random forest model. Outcomes of our modelling reveal balanced accuracy outcomes of 58.49% when sampling was done with ADASYN and 60.30% when sampling was done with SMOTE (+1.81% gain). As above, we focus on SMOTE outcomes given the higher performance observed when sampling with this algorithm. Interestingly, the 60.30% accuracy observed here reflects a decrease in performance when compared to the random forest and RNN models that were trained with only hand-crafted features (i.e., balanced accuracy of 71.74% and 65.89% respectively). Examining the coefficients of the random forest model in TABLE II (Column A), we observed that four features were dominant: the neural-generated feature (Weight = 0.172), and hand-crafted features Timelag = 0.189, Reputation = 0.176, and Codelength = 0.162. Of note is that a similar pattern of outcomes for the prominent hand-crafted features was observed in our models above (refer to TABLE I). We observe in TABLE II that most of the coefficients in Column A were of a higher order of magnitude than those in TABLE I Column A, in divergence with the better overall performance of the model with lesser complexity as noted above (i.e., 71.74% versus 60.30%).

RNN Model with Hand-crafted and Neural-generated Features: To evaluate the effect of increasing features in a more complex model, we re-trained a RNN by using the hand-crafted and neural-generated features. Outcomes of our modelling reveal balanced accuracy outcomes of 81.52% when sampling was done with ADASYN and 82.73% when sampling was done with SMOTE (+1.21% gain). These outcomes are the most accurate of all the modelling done in the study. TABLE II Column B shows that the coefficients for the features were of a higher magnitude in the RNN model, with both hand-crafted and neural-generated features affected. Of note here also is that the same features retained their prominence in this model (Weight, Timelag, Reputation, and Codelength).

TABLE II
COEFFICIENTS OF RANDOM FOREST AND RNN MODELS FOR HAND-CRAFTED AND NEURAL-GENERATED FEATURES

Feature	A	B
	Random Forest	**Neural Network**
Timelag	0.189	0.311
URLCount	0.012	0.029
CommentCount	0.043	0.062
Reputation	0.176	0.298
TextPolarity	0.054	0.014
AnswerCount	0.043	0.075
ViewCount	0.024	0.019
Score	0.087	0.043
NumberOfcodeLine	0.076	0.089
NumberOfSentence	0.054	0.132
TextualSimilarity	0.021	0.043
Codelength	0.162	0.252
TFAnswerCode	0.123	0.134
TFAnswerText	0.032	0.176
Weight	0.172	0.276
Balanced Accuracy	***60.30***	***82.73***

In striving to rigorously evaluate model performances we executed the models 100 times, as noted above. Summary statistics (mean, median (Md), and standard deviation (SD)) for balanced accuracy are provided in Column A of TABLE III. Here it is shown that the mean and median are similar to those recorded in TABLES II and III, with our measurements indicating an absence of outliers. This is validated with the low standard deviation values in TABLE III, confirming that the outcomes of our repeated runs were similar. Formal statistical testing was done on the balanced accuracy outcomes from our 100 executions of the models. The Kruskal-Wall test was used to check for statistical differences in outcomes for the four models, given that our data violated the normality assumption. Outcomes revealed statistically significant differences between the model outcomes, $X^2(3) = 374.07$, $p < 0.01$, with a mean rank score of 250.5 for the random forest model with hand-crafted features, 150.5 for the RNN model with hand-crafted features, 50.5 for the random forest model with hand-crafted and neural-generated features, and 350.5 for the RNN model with hand-crafted and neural-generated features.

We next performed post hoc pair-wise Wilcoxon testing with appropriate Bonferroni adjustments. Our outcomes reveal that there were statistically significant differences in the model outcomes for all comparisons ($p < 0.01$). We observed that the balanced accuracy model outcome was significantly higher for the RNN with hand-crafted and neural-generated (PV-DM) features (model 4 in TABLE IV) when compared to the other models. Also, the random forest with hand-crafted features model (model 1 in TABLE IV) performed significantly better than the RNN with hand-crafted features (model 2 in TABLE IV) and the random forest with hand-crafted and neural-generated (PV-DM) features (model 3 in TABLE IV). Model 2's balanced accuracy was also significantly higher than that of model 3 (refer to TABLE IV for details).

B. *Model Quality*

To understand if there are differences in the quality of outcomes of Q&A prediction models with varying degrees of complexity that are derived using hand-crafted and neural-generated features, we computed a confusion matrix for all four models when sampling with the SMOTE technique. TABLE V shows that all models have acceptable F1-scores, confirming that they are all better than random guesses. The RNN models with both hand-crafted and hand crafted and neural-generated features has the highest F1-scores (0.783 and 0.779 respectively), confirming that the more complex models were of a higher quality. Even so, the random forest model with hand-crafted features has higher quality than the random forest model with hand-crafted and neural-generated (PV-DM) features (0.708 versus 0.573). Results for MCC in TABLE V show that all models have positive MCC values. That said, the random forest model with hand-crafted features and the RNN model with hand-crafted and neural-generated features record the highest MCC values (0.722 and 0.711 respectively). The MCC value for the RNN model with hand-crafted features was only marginally lower than the aforementioned models (0.681). However, as with the F1-score measures, the random forest model with hand-crafted and neural-generated features recorded the poorest quality (MCC = 0.564).

Performing rigorous statistical testing to evaluate the quality of our models, we executed the models 100 times. Summary statistics (mean, median (Md), and standard deviation (SD)) for the F1-score and MCC values are provided in Columns B and C of TABLE III. Here it is shown that the mean and median are similar in each of the two measurements (for F1-Score and MCC), indicating that there were few outliers. This is validated by low standard deviation values in TABLE III. Maintaining a similar pattern to the outcomes in TABLE V, our outcomes in TABLE III show that the F1-Score was highest for the RNN model with hand-crafted features (0.781), followed by the RNN model with the hand-crafted and neural-generated (PV-DM) features (0.779), and the random forest model with hand-crafted features (0.706). The random forest model with hand-crafted and neural-generated (PV-DM) features performed the poorest (0.574). MCC values in TABLE III are slightly variable, where the best performance was noticed for the random forest model with hand-crafted features (0.718). However, again, outcomes were very similar for three of the models, with the random forest model with hand-crafted and neural-generated (PV-DM) features performing the poorest (0.568).

As above, formal statistical testing was conducted on the F1-Score and MCC outcomes from our 100 executions of the models. We first executed the Kruskal-Wallis test for F1-Scores, where outcomes revealed statistically significant differences between the model outcomes, $X^2(3) = 374.06$, $p < 0.01$, with a mean rank score of 150.5 for the random forest model with hand-crafted features, 350.5 for the RNN model with hand-crafted features, 50.5 for the random forest model with hand-crafted and neural-generated features, and 250.5 for the RNN model with hand-crafted and neural-generated features.

TABLE III
MODELS' SUMMARY STATISTIC FOR BALANCED ACCURACY, F1-SCORE AND MCC VALUES

Model	**A**	**B**	**C**
	Balanced accuracy	**F1-Score**	**MCC**
Random forest with hand-crafted features	Mean: 71.185 Md: 71.187 SD: 0.109	Mean: 0.706 Md: 0.706 SD: 0.228	Mean: 0.718 Md: 0.717 SD: 0.004
Neural network with hand-crafted features	Mean: 65.887 Md: 65.942 SD: 0.109	Mean: 0.781 Md: 0.783 SD: 2.075	Mean: 0.686 Md: 0.686 SD: 0.012
Random forest with hand-crafted and neural-generated (PV-DM) features	Mean: 60.670 Md: 60.659 SD: 0.199	Mean: 0.574 Md: 0.574 SD: 0.065	Mean: 0.568 Md: 0.567 SD: 0.001
Neural network with hand-crafted and neural-generated (PV-DM) features	Mean: 82.870 Md: 82.873 SD: 0.067	Mean: 0.779 Md: 0.770 SD: 0.003	Mean: 0.715 Md: 0.716 SD: 0.002

TABLE IV
PAIRWISE COMPARISONS RESULTS FOR BALANCED ACCURACY, F1-SCORE AND MMC VALUES FOR MODELS

Model	**Balanced Accuracy, F1-Score, MCC**		
	1	**2**	**3**
1. Random forest with hand-crafted features			
2. Neural network with hand-crafted features	< 0.01		
3. Random forest with hand-crafted and neural-generated (PV-DM) features	< 0.01	< 0.01	
4. Neural network with hand-crafted and neural-generated (PV-DM) features	< 0.01	< 0.01	< 0.01
Note: Pairwise results were significant at < 0.01 for Balanced accuracy, F1-Score and MCC values (all outcomes)			

MCC outcomes also showed statistically significant differences between model outcomes, $X^2(3) = 342.53$, $p < 0.01$, with a mean rank score of 318.9 for the random forest model with hand-crafted features, 150.5 for the RNN model with hand-crafted features, 50.5 for the random forest model with hand-crafted and neural-generated features, and 282.0 for the RNN model with hand-crafted and neural-generated features.

We next performed post hoc pair-wise Wilcoxon testing with appropriate Bonferroni adjustments. TABLE IV provides our F1-score and MCC outcomes (in addition to those for balanced accuracy, where the pattern of significance was repeated). Here it is revealed that there were statistically significant differences in the model outcomes for all comparisons ($p < 0.01$). We observed that for the F1-Score, the quality of the RNN model with hand-crafted features was significantly better than all the others (p < 0.01). In TABLE IV it is confirmed that the RNN model with hand-crafted and neural-generated (PV-DM) features was of higher quality than both the random forest model with hand-crafted features (p < 0.01) and the random forest model with hand-crafted and neural-generated (PV-DM) features (p < 0.01). Finally, the random forest model with hand-crafted features produced higher quality outcomes than the random forest model with hand-crafted and neural-generated (PV-DM) features ($p < 0.01$). A slightly different pattern of outcomes was observed for the MCC outcomes, albeit the three better performing models above recorded superior scores than the random forest model with hand-crafted and neural-generated (PV-DM) features. This outcome suggests that our predictions are superior to random guesses.

TABLE V F1-SCORE AND MCC VALUES FOR MODELS

Model	**F1-Score**	**MCC**
Random forest with hand-crafted features	0.708	0.722
Neural network with hand-crafted features	0.783	0.681
Random forest with hand-crafted and neural-generated (PV-DM) features	0.573	0.564
Neural network with hand-crafted and neural-generated (PV-DM) features	0.779	0.711

TABLE VI MODEL BALANCE ACCURACY SUMMARY

Model	**Model accuracy for the 90 unique questions taken from the test sample**
Random forest with hand-crafted features	69.37%
Neural network with hand-crafted features	60.0%
Random forest with hand-crafted and neural-generated (PV-DM) features	68.42%
Neural network with hand-crated and neural-generated (PV-DM) features	72.92%

C. Evaluation Via Developers' Questionnaire

A total of 90 unique question and answer pairs were completed and ranked manually by software developers with vary degree of professional experience. These developers ranked answers according to how acceptable they were for a given question. This ranking provides a countermeasure and triangulation for the accepted answer tag that was used as our model outcome measure, and so enables further evaluation of our models. We compare the outcomes of our four approaches with the results gathered from the questionnaire, and the results are shown in TABLE VI.

The accuracy for each model was determined by finding those answers that were predicted to be the accepted answer and also ranked "3" (highly acceptable) by developers. The total number of answers satisfying both conditions was divided by the total number of questions and answers that were evaluated by the questionnaire (90 pairs). Comparing the result of our questionnaire with the four models described in Section III.C, we found that when using the random forest model with hand-crafted features our model was able to accurately predict the acceptable answers as ranked by developers 69.37% of the time. There was, however, less convergence for the RNN model with hand-crafted features (60.00%). Adding the neural-generated (PV-DM) with the hand-crafted features when using random forest modelling recorded a slightly lower accuracy than the random forest model with hand-crafted features (68.42%). However, when we combined the feature generated from the PV-DM model with the hand-crafted features, the RNN model resulted in the highest accuracy (72.92%). While we acknowledge that the number of responses obtained from our questionnaire was small, the pattern of outcomes here largely mirrors that observed above. Further, developers' ranking of highly acceptable answers (i.e., assigning a "3") converged with Stack Overflow users' accepted answer label 84% of the time, suggesting that our respondents had a similar judgement to contributors on Stack Overflow. We further discuss our outcomes in the next section.

V. Discussion

RQ1. *How does the degree of model and feature complexity affect the* ***performance*** *of a Stack Overflow Q&A accepted answer prediction model?* Reflecting on our outcomes, the model with lesser complexity (random forest) tended to outperform our model with more complexity (RNN) when only simple features were used. This finding suggests that the type of features (or sampling techniques) used may affect the performance outcomes of models with varying degrees of complexity when studying prediction models.

In further exploring how the prediction performance varies when changing the model complexity, given the availability of hand-crafted features, we explored the features and their contributed weight in our random forest and RNN models. Our outcomes show that the time it takes to post an answer **(**Timelag), the textual similarity of a question and answer pair and the code length and reputation of the answerer were the most influential predictors of acceptable Stack Overflow answers in both models. This pattern of outcomes was confirmed in preliminary work [10], indicating that the similarity in the text provided in questions and answers enhanced Stack Overflow answers' acceptability. Even so, the reputation of the user was the second most dominating feature that distinguished a chosen Stack Overflow answer. The current finding was also supported by previous work, which linked users' reputation to post quality in other contexts [41].

While code length [11] or features of code readability, such as the number of lines of code and the average number of identifiers per line [18], could be linked to an answer's quality, this might not always be true, as evidence shows that such predictors can be due to chance [42]. The time lag was seen to be the feature with the highest coefficient. Those posting questions reflected on the answers available on the community portal before choosing an acceptable answer. However, on combining the two features sets (hand-crafted features and neural-generated features), the dominating coefficients were the time it takes to post an answer (Timelag), reputation, neural feature (weight) and code length. Of note here is that the neural-generated feature had the third-highest magnitude of all influential features. Also, time lag, reputation and code length were consistent in their prominence among the four top features for both models with varying complexity.

While the (less complex) random forest model outperformed the (more complex) RNN model when hand-crafted features were used, the opposite was observed when neural-generated features were added. In fact, the coefficients of features returned for the RNN model with hand-crafted and neural-generated features were of a much larger magnitude. We also observed that the code length feature (among others) gained prominence when we combined both hand-crafted and neural-generated features for both models. Overall, when considering both types of features, the RNN model had a higher balanced accuracy (i.e., 82.73%) compared to the same model when only hand-crafted features were used (i.e., 65.89%). This outcome aligns with the work of Suggu et al. [14], where the authors proposed that combining features from convolutional networks with hand-crafted features improves model performance. This performance gain may be linked to the ability to learn some features automatically.

These findings are in contrast to those seen for the random forest models, where there was a decline in performance when both hand-crafted and neural-generated features were modelled (performance being 71.74% versus 60.30%). This decline in performance could be attributed to the model not being able to learn the features automatically since previous studies have indicated that it is difficult at times for some models to determine the correlation between textual features and non-textual features [20], which may be responsible for the decline in performance observed. Previous studies have also indicated that using the best features tends to increase the quality of the model, although "best" may be hard to decipher given the difficulty with models in determining the correlation between textual features and non-textual features. This assessment aligns with our outcome where our RNN model outperformed the other models used in this study when we used a combination of complex and hand-crafted features. That said, the challenge of determining which features are "best" remains. A reasonable approach would be to **consider the complexity of both the models and features when choosing a model.**

RQ2. *How does the degree of model and feature complexity affect the* ***quality*** *of a Stack Overflow Q&A accepted answer prediction model?* Our findings show that all of our models were of good quality, and better than random guesses when F1 scores were considered. The RNN models (with hand-crafted features only, and with hand-crafted and neural-generated features) recorded the highest F1 scores (0.783 and 0.779 respectively), confirming that the models with a high degree of complexity were of a higher quality. Despite this, the random forest model with hand-crafted features performed much better than the random forest model with hand-crafted and neural-generated (PV-DM) features (0.708 versus 0.573). Burel et al. [43] reported outcomes with an F1 score of 0.659 when predicting the best answers using the Multi-Class Alternating Decision Tree classifier (MADT) with only hand-crafted features from the Stack Exchange dataset. The MADT is a classifier that is more advanced than the random forest, as it combines decision trees with the predictive accuracy of boosting into a set of interpretable classification rules. Accordingly, this algorithm can be considered more complex than the random forest alternative. When the aforementioned authors increased the feature set by adding other features such as the "relative position of an answer within a post", the F1 score of their MADT increased to 0.769. Their result is similar to those for our RNN classifier, which increases the quality of the model by increasing feature set complexity. Considering that the MCC score is also used to measure the quality of a binary classifier, we examined the MCC score for all our models. Our outcomes show a strong positive MCC score, with the random forest model with hand-crafted features and the RNN model with hand-crafted and neural-generated features recording almost identical MCC values (0.722 and 0.711 respectively). An MCC score of 1 indicates a perfect prediction, and both models had very strong positive MCC values. There was an increase in the RNN MCC score from 0.681 to 0.711 when we increased the complexity of the features by adding the neural-generated feature. In contrast, there was a decrease in quality when observing the random forest MCC score, which reduced from 0.722 to 0.564, as was the case for the F1 score from 0.708 to 0.573, when the complexity of the features increased. This indicates that, **overall, our random forest models decreased in quality when neural-generated features were added**. Here we see a correlation between the quality of models and feature complexity. **When we increased the complexity of our features without increasing the complexity of the model, there was a decline in the quality of the model outcomes**. The opposite pattern is observed when we increase feature complexity and model complexity, where the model quality outcomes increased. These findings converge with those we observed for model performance, where the same pattern of outcomes was recorded, pointing to the need to **match the complexity of the features to that of the modelling techniques.**

Our evaluation done via a questionnaire triangulated our findings, where the same pattern of outcomes was reported, albeit with our models' performance and quality outcomes being slightly lower. Our findings are noteworthy, as a random sample of software developers labelled our Stack Overflow answers independently of the contributors on the platform, where the same pattern of outcomes was observed. Software engineering researchers should thus be cautious in aligning specific modelling approaches with specific feature sets.

VI. Threats to Validity

Construct Validity: We have used two modelling approaches in this work, and thus, we cannot definitively say that our outcomes will hold for other modelling methods (e.g., Bayesian logistic regression or support vector machines (SVM)). It should be noted, however, that our approaches were established to vary widely in complexity based on both their operations and the evidence returned by Calefato et al. [17]. Also, given the limited feature set used (only one complex feature), we cannot guarantee that our outcomes will hold when more complex features are added.

Internal Validity: Also, our conceptualization of acceptable answers is based on the answer that was selected by the user who posted the question. We acknowledge that such answers may not necessarily be the "best" answer in all cases. Nevertheless, our evaluation involved software developers who completed a questionnaire where findings confirmed that 84% of the answers that were accepted by Stack Overflow users were also judged as the most acceptable answer by these practioners.

External Validity: Our dataset consisted of 249,588 records that were posted on Stack Overflow over three years (2014–2016). This dataset does not represent all the types of questions and answers that are provided for posts tagged with the Java or JavaScript label on the Stack Overflow platform. Thus, we do not claim generalisability of our findings, although recent findings suggest that there is consistency in trends across languages for some aspects of Stack Overflow posts [26]. In addition, we acknowledge that the Stack Overflow dataset is not representative of all similar portals. However, those portals that are devoted to addressing technology-related challenges may possess comparable content (e.g., Yahoo!Answers for programming). In this way, the features that distinguish accepted or acceptable answers for such portals are likely to demonstrate similarities.

VII. Conclusion and Future Work

By understanding the performance and quality of different modelling methods with respect to the different features found in a Q&A forum such as Stack Overflow, the prediction of an acceptable answer could easily be achieved when there is none available. Having such answers available early could make it easier for end-users of such a forum to find suitable solutions to their problems. However, from prior work, it was not clear when and/or how features and models with varying degrees of complexity affect prediction outcomes. We have thus bridged this gap. In this paper, we were able to determine that the performance of a model is related to the inputted features' complexity. We observed that models with higher complexity performed better when both hand-crafted and neural-generated features were used. On the other hand, models with lesser complexity work best when only simple hand-crafted features were used. The time it takes to post an answer, the textual similarity of a question and answer pair, code length and reputation of the answerer were the most influential predictors of acceptable Stack Overflow answers, with contributors' reputation standing out as one of the most influential predictors of answer acceptability. We believe this could represent a threat to the Stack Overflow community if contributors employ tactics to game their reputations. Accordingly, our future work will investigate how using different types of complex features will affect the performance and quality of answer prediction models.

REFERENCES

[1] Treude C, Barzila O, Storey M-A, editors. How do programmers ask and answer questions on the web? (nier track). 33rd International Conference on Software Engineering; 2011: ACM.

[2] 2Cordeiro J, Antunes B, Gomes P, editors. Context-based recommendation to support problem solving in software development. 2012 3rd International Workshop on Recommendation Systems for Software Engineering; 2012; Zurich, Switzerland: IEEE Press.

[3] 3Ponzanelli L, Bacchelli A, Lanza M, editors. Leveraging crowd knowledge for software comprehension and development. 2013 17th European Conference on Software Maintenance and Reengineering; 2013; Genova, Italy: IEEE.

[4] 4Nie L, Wei X, Zhang D, Wang X, Gao Z, Yang YJItok et al. Data-driven answer selection in community QA systems. 2017;29(6):1186-98.

[5] 5Sahu TP, Nagwani NK, Verma S. Selecting best answer: An empirical analysis on community question answering sites. IEEE Access. 2016;4:4797-808.

[6] Dong H, Wang J, Lin H, Xu B, Yang Z, editors. Predicting best answerers for new questions: an approach leveraging distributed representations of words in community question answering. 2015 ninth international conference on frontier of computer science and technology (FCST); 2015: IEEE.

[7] Chirag S, Jefferey P, editors. Evaluating and predicting answer quality in community QA. In the 33rd International Conference on Research and development information retrieval on Research and Development in Information Retrieval (SIGIR'10); 2010.

[8] Harper FM, Raban D, Rafaeli S, Konstan JA, editors. Predictors of answer quality in online Q&A sites. SIGCHI Conference on Human Factors in Computing Systems; 2008: ACM New York, NY.

[9] Blooma MJ, Chua AY, Goh DH-L, editors. A predictive framework for retrieving the best answer. Proceedings of the 2008 ACM symposium on Applied computing; 2008: ACM.

[10] Omondiagbe, O. P., Licorish, S. A., & MacDonell, S. G. (2019). Features that Predict the Acceptability of Java and JavaScript Answers on Stack Overflow. Evaluation and Assessment on Software Engineering.

[11] Jeon J, Croft WB, Lee JH, Park S, editors. A framework to predict the quality of answers with non textual features. 29th annual international ACM SIGIR conference on Research and development in information retrieval; 2006; Seattle,USA: ACM New York, NY

[12] Larkey LS, editor Automatic essay grading using text categorization techniques. 21st annual international ACM SIGIR conference on Research and development in information retrieval; 1998.

[13] Jizhou H, Ming Z, Dan Y, editors. Extracting chatbot knowledge from online discussion forums. . International Joint Conference on Artificial Intelligence; 2007; Indian.

[14] Suggu SP, Goutham KN, Chinnakotla MK, Shrivastava M. Deep feature fusion network for answer quality prediction in community question answering. arXiv preprint arXiv:160607103. 2016.

[15] Lei Y, Karl MH, Phil B, Stephen P, editors. Deep learning for answer sentence selection. Neural Information Processing Systems (NIPS): Deep Learning and Representation Learning Workshop; 2014; Montreal, Quebec, Canada.

[16] Wang X-J, Tu X, Feng D, Zhang L, editors. Ranking community answers by modeling question-answer relationships via analogical reasoning. 32nd international ACM SIGIR conference on Research and development in information retrieval; 2009: ACM.

[17] Calefato F, Lanubile F, Novielli NJESE. An empirical assessment of best-answer prediction models in technical Q&A sites. 2019:1-48.

[18] Buse RPL, Weimer WR. Learning a metric for code readability. IEEE Transactions on Software Engineering. 2010;36(4):546-58.

[19] Ghazy RA, El-Rabaie E-SM, Dessouky MI, El-Fishawy NA, Abd El-Samie FEJWPC. Feature selection ranking and subset-based techniques with different classifiers for intrusion detection. 2020;111(1):375-93.

[20] Haifeng H, Bingquan L, Baoxun W, Ming L, Xiaolong W, editors. Multimodal DBN for Predicting High-Quality Answers in cQA portals. Association of Computational Linguistics; 2013.

[21] Wang B, Liu B, Wang X, Sun C, Zhang D. Deep learning approaches to semantic relevance modeling for chinese question-answer pairs. ACM Transactions on Asian Language Information Processing (TALIP). 2011;10(4):21.

[22] Gao Z, Xia X, Lo D, Grundy JJAToSE, Methodology. Technical Q8A Site Answer Recommendation via Question Boosting. 2020;30(1):1-34.

[23] Xiaoqiang Z, Baotian H, Jiaxin L, xiang Y, Xiaolong W. A Deep Learning based Comment Sequence Labeling System for Answer Selection Challenge. ICRC-HIT:. 2015.

[24] Zou Y, Ye T, Lu Y, Mylopoulos J, Zhang L, editors. Learning to rank for question-oriented software text retrieval (t). 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE); 2015: IEEE.

[25] Schaffer C. A conservation law for generalization performance. Machine Learning Proceedings 1994: Elsevier; 1994. p. 259-65.

[26] Lotter A, Licorish SA, Savarimuthu BTR, Meldrum S, editors. Code Reuse in Stack Overflow and Popular Open Source Java Projects. 2018 25th Australasian Software Engineering Conference (ASWEC); 2018: IEEE.

[27] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research. 2002;16:321-57.

[28] He H, Bai Y, Garcia EA, Li S, editors. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. IEEE International Joint Conference on Neural Networks; 2008: IEEE.

[29] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: data mining, inference, and prediction: Springer Science & Business Media; 2009.

[30] Rajbahadur GK, Wang S, Ansaldi G, Kamei Y, Hassan AEJIToSE. The impact of feature importance methods on the interpretation of defect classifiers. 2021.

[31] Svetnik V, Liaw A, Tong C, Culberson JC, Sheridan RP, Feuston BP. Random forest: a classification and regression tool for compound classification and QSAR modeling. Journal of chemical information and computer sciences. 2003;43(6):1947-58.

[32] Cutler A, Cutler R, Stevens JR. Random forests. Ensemble machine learning: Springer; 2012. p. 157-75.

[33] Dewancker I, McCourt M, Clark S. Bayesian Optimization for Machine Learning: A Practical Guidebook. arXiv preprint arXiv:161204858. 2016.

[34] Iyyer M, Boyd-Graber JL, Claudino LMB, Socher R, Daumé III H, editors. A Neural Network for Factoid Question Answering over Paragraphs. EMNLP; 2014.

[35] Léon B. Stochastic gradient learning in neural networks. Neuro-Nımes. 1991;91(8).

[36] Le QV, Mikolov T, editors. Distributed Representations of Sentences and Documents. ICML; 2014.

[37] Kim D, Nam J, Song J, Kim S, editors. Automatic patch generation learned from human-written patches. 2013 International Conference on Software Engineering; 2013: IEEE Press.

[38] Brodersen KH, Ong CS, Stephan KE, Buhmann JM, editors. The balanced accuracy and its posterior distribution. 2010 20th International Conference on Pattern Recognition; 2010: IEEE.

[39] Lipton ZC, Elkan C, Naryanaswamy B, editors. Optimal thresholding of classifiers to maximize F1 measure. Joint European Conference on Machine Learning and Knowledge Discovery in Databases; 2014: Springer.

[40] Ding Z. Diversified ensemble classifiers for highly imbalanced data learning and their application in bioinformatics. 2011.

[41] Lin Z, Li D, Janamanchi B, Huang WJDSS. Reputation distribution and consumer-to-consumer online auction market structure: an exploratory study. 2006;41(2):435-48.

[42] Chen W, Zeng Q, Wenyin L, Hao TJC, Practice C, Experience. A user reputation model for a user-interactive question answering system. 2007;19(15):2091-103.

[43] Burel G, He Y, Alani H, editors. Automatic Identification of Best Answers in Online Enquiry Communities. 9th Semantic Web:Research and Applications: 9th Extended SemanticWeb Conference; 2012: Springer