computers

MDPI

Article

ERF: An Empirical Recommender Framework for Ascertaining Appropriate Learning Materials from Stack Overflow Discussions †

Ashesh Iqbal [1], Sumi Khatun [2], Mohammad Shamsul Arefin [1,*] and M. Ali Akber Dewan [3]

1 Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong 4349, Bangladesh; iqbalashesh@gmail.com
2 Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka 1216, Bangladesh; sumicuet11@gmail.com
3 School of Computing and Information Systems, Faculty of Science and Technology, Athabasca University, Edmonton, AB T5J 3S8, Canada; adewan@athabascau.ca
* Correspondence: sarefin@cuet.ac.bd
† This paper is an extended version of our paper published in the International Conference on Intelligent Tutoring Systems.

Received: 28 May 2020; Accepted: 17 July 2020; Published: 20 July 2020

Abstract: Computer programmers require various instructive information during coding and development. Such information is dispersed in different sources like language documentation, wikis, and forums. As an information exchange platform, programmers broadly utilize Stack Overflow, a Web-based Question Answering site. In this paper, we propose a recommender system which uses a supervised machine learning approach to investigate Stack Overflow posts to present instructive information for the programmers. This might be helpful for the programmers to solve programming problems that they confront with in their daily life. We analyzed posts related to two most popular programming languages—Python and PHP. We performed a few trials and found that the supervised approach could effectively manifold valuable information from our corpus. We validated the performance of our system from human perception which showed an accuracy of 71%. We also presented an interactive interface for the users that satisfied the users' query with the matching sentences with most instructive information.

Keywords: text classification; supervised learning; crowd knowledge; recommender system

1. Introduction

In the field of software engineering, learning is a life-time pursuit for professionals seeking to maintain competency. It has been well-established in educational and training circles that fostering communities of learners can facilitate the achievement of learning goals. Online social networks clearly have a strong role to play in this aspect. At present, many websites are available to share information and exchange knowledge with peers. Stack Overflow is one such popular Web repository that helps programmers to share and obtain useful information about programming. Launched in the year of 2008 by Jeff Atwood and Joel Spolsky, StackOverflow.com has become a widely used forum for people to interact on topics related to computer programming. Over a period of time, it has gained the reputation of being a reliable forum where users get quick responses to their questions, and that too with high level of accuracy. Stack Overflow data have been a pool for research from many perspectives. Such researches have helped to understand the power of programming-specific question and answering (Q&A) forums and how far these forums are serving as a learning platform.

In computer programming, learning is an actual interest for the programmers who thrive off increasing their proficiency. In the rapid pace of technological changes, the official documentation of programming languages are often lagging behind [1]. Bacchelli et al. [2] stated that the project documentations are commonly inadequate, manuals tend to be outdated, and books may be hard to retrieve or link to the actual task. Online social platforms are found to be promising to fill this gap. The reason is that the programming tasks often demand knowledge that are diversified among numerous people with different specializations and skills [1].

Due to this fact, Stack Overflow has picked up the notoriety of being a solid gathering, where users get nimble responses to their inquiries identified with computer programming and with an elevated level of precision [3,4]. Stack Overflow data has been a pool for research from many perspectives. Such researches [1,5–11] have assisted with understanding the intensity of programming-explicit Q&A discussions and how widely these forums are filling in as learning stages. It is even inferred that the appropriate responses on Stack Overflow often become a substitute for authentic item documentation where the official documentation remains to be inadequate [12]. Parnin et al. [5] asserts that Stack Overflow has developed into an enormous storehouse of user-engendered content that could supplement conventional specialized documentations. Nonetheless, jam documentation is made in a casual way with no extensive association or express connects to API components [5,13]. Even though we can surmise the estimation of group documentation by a number of contributors and commitments present, for understanding the extent of contributions or assessing the worth and nature of those legacies, we have not done much yet. Additionally, common studies have not considered the intricacy of cases to make documentation versatile to various degrees of developer experience.

Although there are some research [14,15] to recommend tags for Stack Overflow discussions, to the best of our knowledge there is no efficient technique for recommending learning materials to the users from Stack Overflow discussion. Considering this fact, we develop a technique to recommend appropriate learning materials for the two popular programming languages—Python and PHP. The main goal of this research is to build a recommendation system using a supervised machine-learning technique to suggest richer discussion posts to the programmer on demand. The contribution of this research is the development of a system that can automatically identify richer discussion posts and recommend them to the users from Stack Overflow repositories. We have achieved satisfactory results by validating with the machine extracted information from human perception. Our framework provides users with the appropriate guidance complementary to the original software documentation. This work is an extended version of our previous paper [16]. The main contributions of this work can be summarized as follows:

- We exploit the knowledge repository available from the Stack Overflow discussions to generate learning materials related to PHP and Python.
- We use supervised learning techniques to satisfy users' queries with a matching Q&A discussions those are the most insightful.
- We develop an user friendly interface to help users in easy interaction with the system.
- We perform subjective and objective evaluation of the system to check the efficiency of our system.

The rest of the paper is organized as follows: Section 2 provides a short review of the related literature. Section 3 presents the design consideration and techniques used for the proposed framework. Section 4 details the implementation and experimental results. Finally, we conclude the paper in Section 5.

2. Literature Review

In computer programming, continuous learning is important using online resources. Despite the short history of Stack Overflow, many studies have put a great effort to investigate the contents, available metadata, and user-behavior of this online forum. Existing works identified in our review are categorized into empirical study of stack overflow data, harnessing stack overflow data,

and improving programming language usage documentation utilizing social media channels. We have also reviewed works on stipulating automatic standard appraisal in other advanced archives of text-based information.

2.1. Empirical Study of Stack Overflow Data

With the particular objective of revealing fundamental discussion points, elementary outbuildings and courses with time, Barua et al. [17] represents an approach for analyzing Stack Overflow data. This is a semi-automatic method and it's analysis gives an estimate of the requirements and needs of the concurrent implementer. Similar research was done by Wang et al. [4] who performed both text and graph-based analyses to derive subjects from questions and distribute an inquiry to a few points with certain probabilities. They also investigate the distribution of participants who ask questions and respond to the questions in Stack Overflow and observed their behavior. The main limitation of this study is that they analyzed only 385 questions manually which is not sufficient to analyze such a huge system like Stack Overflow.

Barzilay et al. [1] reviewed Stack Overflow from the perspective of its design and social communication features. This study also explored its usage in terms of model-driven programming worldview and the role it plays in the documentation view of software development. Rosen et al. [18] performed a study on the issues confronted by the mobile application developers. This study examined 13,232,821 Stack Overflow posts related to mobile application development. The findings of this study help in highlighting the challenges met by mobile developers that require more consideration from the research of software engineering and development groups for going forwards. Another similar work has been done by Linares-Vásquez et al. [19], where they performed an explanatory analysis of mobile application development issues utilizing Stack Overflow. However, this study has broken down only 450 Android related posts manually to decide on the issues. Bajaj et al. [20] analyzed fine-grained parts of Stack Overflow with current Web application improvement. They did a subjective investigation of more than 5,000,000 Stack Overflow addresses which were identified with the Web improvement, and removed the noteworthy bits of intuitions from the information for the developers and research groups.

2.2. Harnessing Stack Overflow Data

An Eclipse integrated development environment (IDE) plug-in is proposed by Bacchelli et al. [2] for integrating Stack Overflow contents automatically which can establish questions from the context of IDE. They show the exchange view of results with the ranked index. The disconnection of the tool from the IDEs and the absence of the team efforts have confined their maximum capacity for program design which is a limitation of the system. Ponzanelli et al. [21] proposed another tool that gives the IDE related information retrieval from Stack Overflow. They graded their work's significance by notifying the developers about existing help if a particular level of threshold is crossed. Their system shows high efficiency when it deems itself to have enough recommendations that does not force developers to invoke any additional suggestions.

2.3. Improving Programming Language Usage Documentation

Campbell et al. [22] questioned whether the developer-written documentation provides enough guidance for programmers based on the observation of hundreds of thousands of programming-related questions posted on Stack Overflow. They managed to find topics related to PHP and Python languages in need of tutorial documentation using the Latent Dirichlet Allocation (LDA) technique. However, the project's documentation cannot address sufficiently the manual analysis of the topics guided by their method.

An automated comment generator has been presented by Wong et al. [23] for concentrating code sections from Stack Overflow, and introduces this information to consequently create explanatory comments for comparable code sections in open-source ventures. The authors applied their method to

deal with Java and Android projects and automatically figured out 102 comments for 23 projects. In a survey, most members found the produced comments precise, sufficient, brief, and valuable in helping them comprehend the code. This study worked on small-sized code-depiction mapping databases by including Stack Overflow answers that do not have the most noteworthy vote check. It cannot supplant the code clone identification apparatus with one that can identify expansion and reordering of lines to expand the number of code matches.

Treude et al. [24] analyzed Java API related posts in Stack Overflow to extract useful information which are unavailable in the documentation. Seemingly, this work is the most similar to our work since it likewise harnesses the content of natural language accessible on Stack Overflow. As far as utilizing Artificial Intelligence (AI) to find content on Stack Overflow, there are some basic subjects among our own and crafted by de Souza et al. [8] who built up an improved Web search tool utilizing the machine learning techniques to find content on Stack Overflow. This study has made a qualitative manual analysis of their system with the consideration of only 35 programming problems on three different areas (Swing, Boost and LINQ).

Kim et al. [11] presented a framework which recommended the users the API documentations. This framework uses a Web mining technique to embed summaries of coding related posts to the API. However, the database which is related to programming for addressing their cases may not be representable for utilizing eXoDus. A hands-on, analytical approach for connecting program code examples to the documentation of related API is proposed by Petrosyan et al. [25] and Subramanian et al. [10]. Petrosyan et al. [25] used classification based method for discovering helpful tutorials for a provided API. They used supervised classification of text yielding linguistic features. Subramanian et al. [10] implemented an approach called Baker which retrieves JavaScript and Java API documentation with a high precision of about 97%.

2.4. Other Related Work

Le et al. [26] have mined the posts from "Brainly", a community question-answering site, to determine what constitutes answer quality. They built their classification model by integrating different groups of features: personal, community-based, textual, and contextual. Their findings indicate that personal and community-based features have more prediction power in assessing answer quality. Their approach also achieved high values on other key metrics such as F1-score and Area under Receiver Operating Characteristic (ROC) curve. Numerous answers were erased from their site because of low-quality and were not clear. However, it was not clarified by the authors well how the removal of the questions could influence their system's performance.

Shah et al. [27] conducted a large scale analysis of knowledge sharing within "Yahoo! Answers". They considered the notion of users satisfaction as the indication of answer quality and employed logistic regression to predict which of the answers to a given question the user would pick as the best. They demonstrated the robustness of their custom model by validating the results from human perception. As this study selects answer arbitrarily based on prior information, the classifier may fail to address the best quality answer if it has not been trained with a sufficient amount of good quality training data.

Maity et al. [28] studied "Quora" to build an efficient mechanism to characterize the answerability of the questions asked. They investigated various linguistic activities to discriminate between the open and answered questions. Their research discovered the correspondence of language usage patterns to quality factors in the forum. They have made a comparison among different models related to their mechanism and showed that their system outperforms with a accuracy of 76.26%.

We have reviewed several research studies related to our work and come up with a consideration of a methodology by crawling data from Stack Overflow followed by pre-processing the data and performing a sort of recommendation using classification. This methodology is implemented as a framework for finding helpful learning materials from Stack Overflow for popular programming languages- Python and PHP.

3. Methodology

Figure 1 depicts the architecture of the proposed system. The system consists of an interface module, a search module, a processor module, and an input-output module. In the offline phase, data from Stack Overflow undergoes pre-processing and feature building followed by training set annotation and incorporation of machine learning to predict desired information from never-before-seen-data. Meanwhile, tags available on Stack Overflow are categorized into three distinct sets: novice, intermediate, and expert. All these data are stored in the database. In the online phase, a user interacts with the system and vets to measure the user's expertise on a particular field and the predefined list of tags is leveraged to retrieve appropriate data from the database.

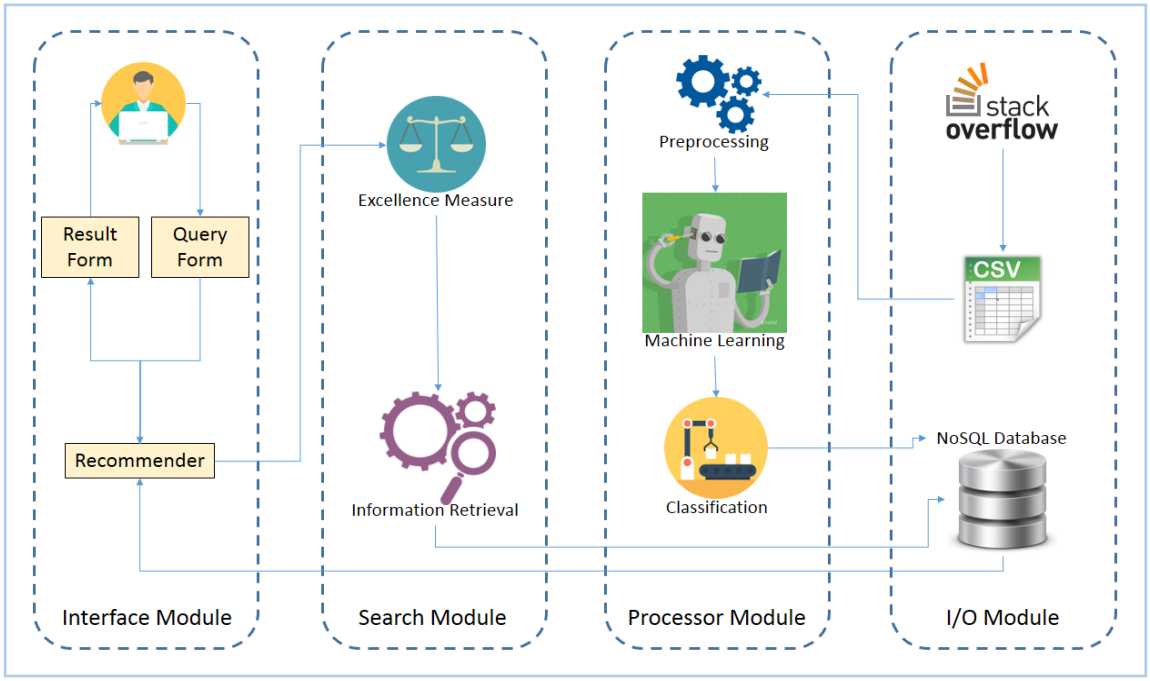

Figure 1. System architecture of the proposed framework.

3.1. Dataset Description

Stack Overflow makes its data publicly available online (https://archive.org/details/stackexchange/) in Extensible Markup Language (XML) format under the Creative Commons (CC) license. We downloaded a data dump containing a total of 33,456,633 posts, spanning from July 2008 to September 2016, and imported the XML files into a MySQL relational database. We used simple SQL queries to identify PHP and Python-related questions and find the matching answers. This query resulted in a dataset containing 930,429 "answer-type" posts. We observed that the dataset contained incomplete records and since such data might mislead the machine learning techniques which we intend to apply, we got rid of those incomplete data and the size of our initial dataset was reduced to 323,801 posts. Next, we extracted the corresponding questions to the answers in our dataset as the "question-type" posts contained extra information concerning the question-answer pair. Then we got rid of records with incomplete information and the size of the dataset is further reduced to 190,215 posts. Following this reduction, we further excluded corresponding posts from the "answer-type" corpus. We included the reputation score of each asker and answerer which was calculated based on their activity on the site. Finally, a total of 190,215 posts were left which were exported from the database as Comma Separated Value (CSV) format to undergo pre-processing.

3.2. Data Preprocessing

In the official data dump of Stack Overflow, standalone code snippets are surrounded with HTML tags `<pre>` and `</pre>`. We removed these code snippets because many code elements in such code snippets are defined by the askers or answerers of questions for illustration purposes and these code elements do not refer to software-specific entities that the other developers are concerned with. However, we kept small code elements embedded in the post texts that are surrounded by `<code>` and `</code>` tags. These small code elements often refer to APIs, programming operators, and simple user-defined code elements for explanation purposes. Removing them from the texts will impair the completeness and meaning of sentences. Finally, we stripped all other HTML tags from the post texts. All these were done using "pandas" (a data analysis library that provides high performance data structures and analysis tools) [29] and "lxml" (a feature-rich library for processing XML and HTML) [30]. Next, the posts were divided into individual sentences. Accordingly, we dived into one of the grand challenges in Natural Language Processing (NLP) called Sentence Boundary Detection (SBD). Surprisingly, the state-of-the-art tools could not perfectly account for the unique characteristics in case of informal programming related discussion not found in other texts. We examined several NLP libraries, such as "NLTK" [31], "segtok" (http://fnl.es/segtok-a-segmentation-and-tokenization-library.html) and "spaCy" [32], and finally selected the "spaCy" as it was found to be performing better on our corpus. Our dataset was divided into 719,614 sentences on which our research was conducted. Since we divided the threads into individual sentences, we assigned a unique identifier to each sentence. This identifier was later used to append the parent posts' attributes to the child sentences. Algorithm 1 summarizes the data pre-processing steps.

Algorithm 1 Preprocessing Algorithm.

```
1:  procedure PRE–PROCESSING
2:      for each post in document do
3:          if pattern match = "< pre > (.+?) < /pre >" then
4:              replace with "∼ HEREWASSOMECODE ∼"
5:          else
6:              if pattern match = "< ahref = .+? > ((.+?) < /a >)*" then
7:                  replace with "∼ HEREWASAURL ∼"
8:          remove html − tags
9:          if pattern match = "\r*\n+" then
10:             remove
11:         if pattern match = "\s+" then
12:             remove
13:         split at sentence
14:         sentence ← "ID" and "PARENT-POST-ID"
15: end procedure
```

3.3. Training and Test Set Generation

Supervised learning requires labeled data. We selected a batch of 1000 sentences from our constructed subset and manually annotated them with a yes/no rating to indicate whether it was informative. We defined "informative sentences" as the ones that are "meaningful on its own and conveys specific and useful information". Such a sentence is supposed to bring a thorough understanding or a fresh standpoint to the topic at hand. However, it should be noted that everyone can make his or her own subjective judgements on the quality of content depending on various

measurements. In Table 1, we list a few sentences that were considered informative. During manual labeling, we followed a set of rules proposed by Treude et al. [24]. The handcrafted set of 1000 sentences described previously was appended with corresponding features to build our training and test set. These sets were excluded from the dataset obtained by data pre-processing (Subsection 3.2) and the remaining 718,614 sentences constituted the validation set. In the next step, we generated the Attribute-Relation File Format (ARFF) files from the training, test, and validation sets. This is the native format for the machine learning tool we used.

Table 1. Example of informative sentences.

Sentence	Post ID (stackoverflow.com)
When an upload occurs , the target script is executed when the upload is complete, so PHP needs to know the maximum sizes beforehand.	949415
After a DeadlineExceededError, you are allowed a short amount of grace time to handle the exception, e.g., defer the remainder of the computation.	2248811
If an element is containing floating element, the wrapping element then needs a overflow:auto or a clear:both; to get the height it needs to add background to it.	12663826
The standard escape character for .htaccess regular expressions is the slash (“\”).	34464235

The types of sentences that were not considered as informative are: the sentence resulted from a parsing error, the sentence contains an explicit reference to another sentence, the sentence contains explicit reference to a piece of context that’s missing, the sentence requires another sentence to be complete, the sentence contains a comparison that is incomplete (i.e., one part of the comparison is missing), the sentence starts with “but”, “and”, “or” etc., the sentence contains references to “it”, “this”, “that” etc. which is not resolved within the sentence, the sentence that is grammatically incomplete, the sentence which is a question, the sentence that contains only a link, the sentence itself is a code snippet or is prefacing one (often indicated by a colon), the sentence references code elements that come from user examples, the sentence references specific community users, the sentence contains communication between Stack Overflow users, the sentence contains a reference to something that is not an obvious part of the Python or PHP language, and the sentence is a generic statement that is unrelated to the Python or PHP language. A few example sentences are listed in Table 2.

Table 2. Example of non-informative sentences.

W hat you’re trying to do here is rather unusual, to say the least.
See the next step if you need a mutable list.
You might be surprised to find out that in your sample code, s1 == s2 returns true!

3.4. Feature Extraction

We defined 18 attributes to characterize our data. The construction of our feature set was based on careful inspection of our corpus. We also took advantage of some previous studies [6,24,33,34] that worked with Stack Overflow data. It is safe to say that our feature set captures the structural, syntactic, and metadata information of the dataset. Out of these 18 attributes, three were related to the number of occurrences of keyword terms in the “question body” and “answer body” of a post, four were related to the presence or absence of source code in the “question body” and “answer body”, and the remaining attributes were either directly obtained or calculated from the post metadata. Our procedure of building the feature set is depicted in Algorithm 2 and the resulting attributes were outlined in Table 3. The Algorithm 2 parsed the dataset to find information like the size and age of an answer, the time taken by the question to get answered, if the answer contains any code, the position

of each sentence in the answer post, if the sentence started with lower-case letter and if the sentence was only a code-snippet.

Algorithm 2 Feature Extraction Algorithm.

procedure FEATURE–EXTRACTION
  for each *post* in *post_document* **do**
    if "$\sim$ *HEREWASSOMECODE* $\sim$' is in "POST-BODY" **then**
      "has_code" $\leftarrow$ "1"
    else"has_code" $\leftarrow$ "0"
    select "age"
    "age" $\leftarrow$ *today* ***minus*** "A-CREATION-DATE"
    convert "age" into *seconds*
    select "timedifftoq"
    "timedifftoq" $\leftarrow$ "A-CREATION-DATE" ***minus*** "Q-CREATION-DATE"
    convert "timedifftoq" into *seconds*
    select "size"
    "size" $\leftarrow$ ***length*** of "POST-BODY"
  for each *sentence* in *sent_document* **do**
    select "s_token"
    "s_token" $\leftarrow$ ***length*** of "SENT-BODY"
    if "SENT-BODY" = "$\sim$ *HEREWASSOMECODE* $\sim$' **then**
      "s_is_code" $\leftarrow$ "1"
    else"s_is_code" $\leftarrow$ "0"
    if "SENT-BODY[0]" = "$[a-z]$' **then**
      "s_starts_lc" $\leftarrow$ "1"
    else"s_starts_lc" $\leftarrow$ "0"
  for each *sentence* in *sent_document* **do**
    if *sent_document*."ID" == *post_document*."ID" **then**
      append
      select "s_pos_inpost"
      "s_pos_inpost" $\leftarrow$ position of *sentence* in *post*
end procedure

Table 3. Feature set.

	Feature Name	Description
1	s_pos_inpost	position of the sentence in the original answer post
2	s_token	number of tokens in the sentence
3	s_is_code	if the sentence is a code snippet
4	s_start_lc	if the sentence starts lowercase
5	a_score	score obtained by the original answer post
6	timedifftoq	time interval between asking of the question and accepting an answer
7	a_size	length of the original answer post
8	a_age	age of the original answer post
9	a_has_code	if the original answer contains any code snippet
10	anserer_rep	reputation earned by the user providing the answer
11	q_score	score obtained by the question post
12	q_favcount	number of favourable votes acquired by the question post
13	q_viewcount	number of times the question has been viewed
14	q_anscount	number of answers submitted against the question
15	q_size	length of the question post
16	q_age	age of the question post
17	q_has_code	if the question post contains any code snippet
18	asker_rep	reputation earned by the user asking the question

3.5. Classification Experiment

To conduct supervised learning from our training dataset, we used the WEKA workbench which is recognized as a landmark system in data mining and machine learning [35]. Initially, WEKA identified that our data was imbalanced towards "not informative" instances. To fix this issue, we applied the Synthetic Minority Oversampling Technique (SMOTE) [36] and increased the number of "informative" instances by oversampling. After SMOTE filtering, cross-validation of the dataset led to choosing $k = 5$ in the case of Support Vector Machine classifier and $k = 3$ in the rest of the cases. We tested five different machine learning algorithms including ensemble approaches on our training set. These algorithms were recommended by previous researchers [8,37] for text classification in the software engineering domain: J48 (Decision Tree) [38], SMO (Support Vector Machine) [39], PART (Decision List) [40], IBk (k-Nearest Neighbour) [38], and Random Forest [41]. For model training and testing, we have used 10-fold cross-validation [42,43] to reduce the risk of over-fitting.

3.6. Ranking and Categorization of Result

During the classification operation, WEKA measured a level of confidence for each prediction made on the "never-before-seen" instances. We exploited this information to rank the "informative" sentences extracted from our test set. To devise a categorization rule in our framework, we followed the approaches used in a few researches [13,20] on topic-modeling. We grouped relevant tags by leveraging the query option of the "Stack Exchange Data Explorer" site (https://data.stackexchange.com/stackoverflow/). We inspected this list of tags, and partitioned the most popular tags into three categories: novice, intermediate, and expert. These categories were maintained during the presentation of data through our user interface.

3.7. Persistent Data Store

We chose a NoSQL database to work at the back-end of our interface. MySQL is adequate for storing the raw data obtained from Stack Overflow. However, in case of integrating a corpus built from our information retrieval approach, it does not provide the flexibility that a document-based database such as MongoDB can provide. MongoDB implements the B-Tree data structure which allows the query optimizing component to quickly sort through and order the documents. Han et al. [44] have documented the access speed in MongoDB to be 10 times than MySQL.

3.8. User Interface

The user interface was designed for simplicity. It could supply users of the system with appropriate documents on particular topics. Figure 2 provides a sample output.

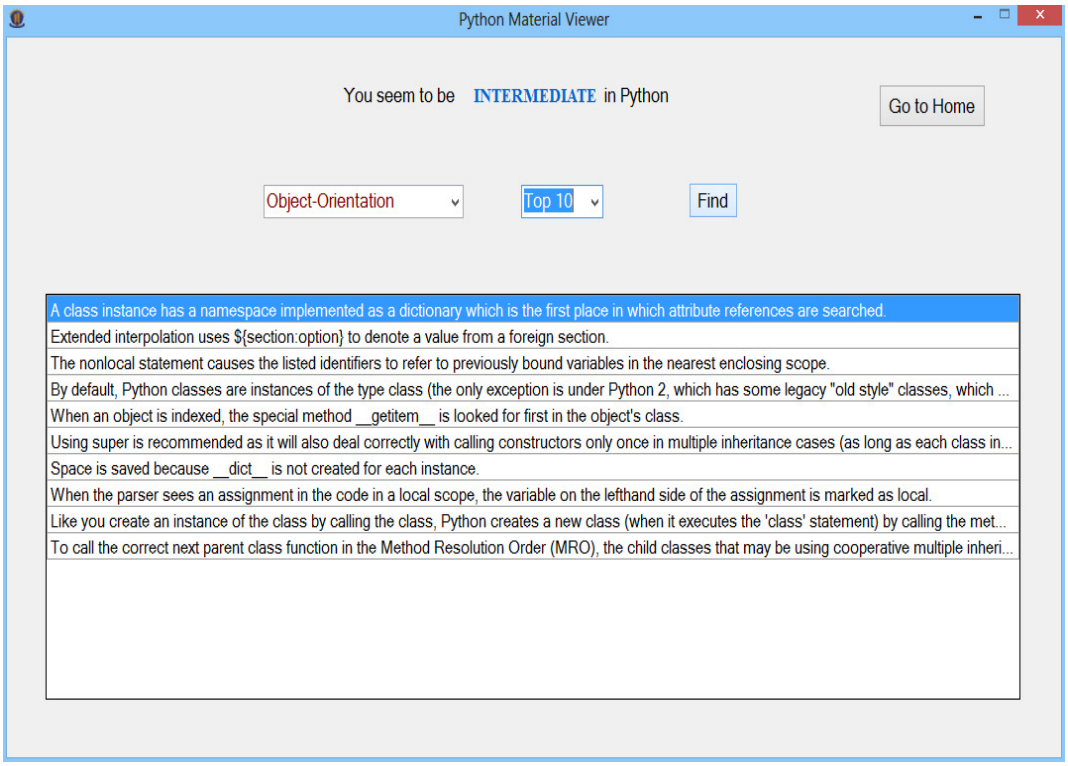

Figure 2. User interface.

4. Experimental Settings and Results

This section notes our experimental setup, quantifies the performance of our approach and provides a qualitative evaluation.

4.1. Experimental Setup

The system has been developed in a machine running on Intel Core i5 2.5 GHz CPU and 4 GB RAM. We have used Python and MySQL for data processing and WEKA for analytics. The front-end was developed using Visual C# and MongoDB NoSQL database is used at the back-end.

4.2. Performance Evaluation

To evaluate the performance of the classifiers in our experiment, we took both accuracy and f-measure as performance metrics. The logic behind taking f-measure as a performance metric is that it allows us to remove any choice biases as most of the instances were annotated as “not informative”. We found that the Random Forest classifier showed the most promising performance. Table 4 shows

the accuracy, precision, recall, and f-measure for the classifiers. From the results of Table 4, we can see that in case of the three classifiers the precision and recall values are identical. This is because the number of false positives and the number of false negatives generated by these there classifiers are found to be almost same.

Table 4. Performance of different classifiers.

Classifier	Accuracy	Precision	Recall	F–Measure
Decision List	95.3093%	0.953	0.953	0.953
Decision Tree	95.8763%	0.959	0.959	0.959
k-NN	89.7938%	0.901	0.898	0.898
Random Forest	98.3505%	0.984	0.984	0.984
SVM	72.2165%	0.723	0.722	0.722

The confusion matrix produced from 10-fold cross-validation of the Random Forest classifier on our training data is shown in Table 5. It can be seen that out of 1940 instances 1048 instances were classified correctly achieving an accuracy of 98.3505%. It is evident from the analysis that our approach is quite successful in both identifying "informative" and "not informative" sentences. The trained model of the Random Forest classifier was re-evaluated on the test set to predict the class labels. It predicted 6886 sentences to be "informative" from the "never-before-seen" 718,614 sentences in our test set.

Table 5. Confusion Matrix.

		Classified as		
		Yes	No	Total
True class	Yes	960	10	970
	No	22	948	970
	Total	982	958	1940

4.3. Human Validation

Like any human activity, our classification is prone to human bias. So, the prediction of the classifier was made subject to human-scrutiny to uncover threats to validity. For this purpose, we randomly selected 100 sentences from the 6,886 sentences extracted by our supervised approach. We asked two personnel with a background in computer science (a master's student and a software developer) to participate in the study. Each of them independently rated each sentence as being one of the following: (a) meaningful and conveys useful information; (b) meaningful but does not provide any useful information; (c) requires more context to understand; (d) makes no sense. The inter-rater agreement matrix illustrated in Table 6 shows that out of the 100 pairs of ratings, 71 were in perfect agreement. The highest number of disagreements was related to sentences that were either type-b (meaningful but does not provide any useful information) or type-c (requires more context to understand). On the whole, it is safe to say that the human raters have found our experiment result to be correct up to 71%.

Table 6. Inter-rater agreement matrix.

	a	**b**	**c**	**d**
a	51	12	2	0
b	–	6	14	0
c	–	–	11	1
d	–	–	–	3

We also perform validation of several chunks containing K-ranked sentences from human perception. The aforementioned participants were called in again to render their services. The results are listed below in Tables 7–9.

Table 7. Inter-rater Agreement Matrix (K = 5).

	a	**b**	**c**	**d**
a	4	1	0	0
b	–	0	0	0
c	–	–	0	0
d	–	–	–	0

Table 8. Inter-rater Agreement Matrix (K = 10).

	a	**b**	**c**	**d**
a	8	1	1	0
b	–	0	0	0
c	–	–	0	0
d	–	–	–	0

Table 9. Inter-rater Agreement Matrix (K = 20).

	a	**b**	**c**	**d**
a	15	3	2	0
b	–	0	0	0
c	–	–	0	0
d	–	–	–	0

We calculate the performance metrics for each case as well. We devise the following heuristics during computation: From the 4x4 matrices in Tables 7–9, we consider cell (1,1) as *True-Positive*; sum of cells (1,2) and (1,3) as *False-Positive*; sum of cells (2,2), (3,3) and (4,4) as *True-Negative*; sum of cells (1,4), (2,3), (2,4) and (3,4) as *False-Negative*.The result of the comparative study is summarized in Table 10.

Table 10. Evaluation for different values of K.

	Top 5	**Top 10**	**Top 20**
ACCURACY	80%	80%	75%
PRECISION	0.8	0.8	0.75
RECALL	1	1	1
F–MEASURE	0.889	0.889	0.857

It is observed that the classification predictions with high confidence match more with human rating. That is why the accuracy and precision/recall of top (more confidently predicted) posts according to our model was higher when matched against human validation results compared to less confidently classified posts. This is an indication that all the evaluation outcomes were consistent.

5. Discussion

In this paper, several models for were constructed for evaluating content quality of Q&A sites based on literature review and found that the Random Forest classifier showed the most promising performance. We also found that for three classifiers the precision and recall values are identical. This happened because the number of false positives and the number of false negatives generated by these there classifiers in our system were found to be almost same. We also conducted some tests to demonstrate the robustness of answer quality via 10-fold cross-validations. The trained models also have been applied to a larger "never-before-seen" dataset. We designed a software-specific tutorial extraction system that presents "informative" sentences from Stack Overflow. We realized that one must first understand the unique characteristics of domain-specific texts that bring unique design challenges. We considered the metadata available on Stack Overflow along with natural language characteristics to construct our information extraction process. Then, based on the understandings of these design challenges, we used state-of-the-art supervised machine learning and NLP techniques to develop our system. We considered several techniques to check the efficiency of our system. The predictions made by our model was further verified by potential users of the system. An interface was built to present the extracted "informative" sentences in a user-friendly manner.

6. Conclusions

Stack Overflow is a popular website among the programmers for finding important information about different programming related issues. Although there are several methods for recommending tags to the users while posting a question at Stack Overflow website, according to our knowledge, there is not any suitable work to obtain quality answers for the users. Considering this fact, in this paper, we provided a method to recommend efficient learning materials for the programmers. In our work, we presented an approach to leverage QA crowd knowledge. By building models and experimenting with several classifiers, the most significant features for predicting answer quality were discovered. As future work, we plan to extend this work beyond Python and PHP and to experiment with more features that capture the grammatical structure of sentences on Stack Overflow discussions.

Author Contributions: Specification of the individual contributions: Conceptualization, A.I. and M.S.A; investigation, A.I., M.S.A., S.K., M.A.A.D.; methodology, A.I. and M.S.A.; software, A.I.; validation, A.I. and M.S.A.; writing—original draft preparation, A.I.; writing—review and editing, M.S.A., S.K. and M.A.A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Barzilay, O.; Treude, C.; Zagalsky, A. Facilitating crowd sourced software engineering via stack overflow. In *Finding Source Code on the Web for Remix and Reuse*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 289–308.
2. Bacchelli, A.; Ponzanelli, L.; Lanza, M. Harnessing Stack Overflow for the IDE. In Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering, RSSE '12, Zurich, Switzerland, 4 June 2012; IEEE Press: New York, NY, USA, 2012; pp. 26–30.
3. Abdalkareem, R.; Shihab, E.; Rilling, J. What Do Developers Use the Crowd For? A Study Using Stack Overflow. *IEEE Softw.* **2017**, *34*, 53–60. [CrossRef]
4. Wang, S.; Lo, D.; Jiang, L. An Empirical Study on Developer Interactions in StackOverflow. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, 18–22 March 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 1019–1024. [CrossRef]
5. Parnin, C.; Treude, C.; Grammel, L.; Storey, M.A. Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow. *Ga. Inst. Technol. Tech. Rep* **2012**, *11*. Available online: http://chrisparnin.me/pdf/crowddoc.pdf (accessed on 19 July 2020).

6. Joorabchi, A.; English, M.; Mahdi, A. Text mining stackoverflow: Towards an Insight into Challenges and Subject-Related Difficulties Faced by Computer Science Learners. *J. Enterp. Inf. Manag.* **2016**, *29*, 255–275. [CrossRef]
7. Ponzanelli, L.; Bacchelli, A.; Lanza, M. Leveraging Crowd Knowledge for Software Comprehension and Development. In Proceedings of the 2013 17th European Conference on Software Maintenance and Reengineering, Genova, Italy, 5–8 March 2013; pp. 57–66. [CrossRef]
8. De Souza, L.B.L.; Campos, E.C.; Maia, M.d.A. Ranking Crowd Knowledge to Assist Software Development. In Proceedings of the 22nd International Conference on Program Comprehension, ICPC 2014, Hyderabad, India, 2–3 June 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 72–82. [CrossRef]
9. Campos, E.; Souza, L.; Maia, M. Searching crowd knowledge to recommend solutions for API usage tasks. *J. Software Evol. Process.* **2016**. [CrossRef]
10. Subramanian, S.; Inozemtseva, L.; Holmes, R. Live API Documentation. In Proceedings of the 36th International Conference on Software Engineering, ICSE 2014, Hyderabad, India, 31 May–7 June 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 643–652. [CrossRef]
11. Kim, J.; Lee, S.; Hwang, S.W.; Kim, S. Enriching Documents with Examples: A Corpus Mining Approach. *ACM Trans. Inf. Syst.* **2013**, *31*. [CrossRef]
12. Treude, C.; Barzilay, O.; Storey, M.A. How Do Programmers Ask and Answer Questions on the Web? (NIER Track). In Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, Honolulu , HI, USA, 21–28 May 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 804–807. [CrossRef]
13. Souza, L.B.L.; Campos, E.C.; Maia, M. On the Extraction of Cookbooks for APIs from the Crowd Knowledge. In Proceedings of the 2014 Brazilian Symposium on Software Engineering, Maceió, Brazil, 28 September–3 October 2014; pp. 21–30.
14. Wang, H.; Wang, B.L.C.X.L.H.J.; Yang, M. SOTagRec: A Combined Tag Recommendation Approach for Stack Overflow. In Proceedings of the ICMAI 2019, Chengdu, China, 12–15 April 2019; pp. 146–152.
15. Wang, S.; Lo, D.V.B.; Serebrenik, A. EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites. In Proceedings of the IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 29 September–3 October 2014; pp. 291–300.
16. Iqbal, A.; Arefin, M.S.; Dewan, M.A. A framework to recommend appropriate learning material from stack overflow discussions. In Proceedings of the International Conference on Intelligent Tutoring Systems, Montreal, QC, Canada, 11–15 June 2018; pp. 446–449.
17. Barua, A.; Thomas, S.W.; Hassan, A.E. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empir. Softw. Eng.* **2012**, *19*, 619–654. [CrossRef]
18. Rosen, C.; Shihab, E. What Are Mobile Developers Asking about? A Large Scale Study Using Stack Overflow. *Empir. Softw. Engg.* **2016**, *21*, 1192–1223. [CrossRef]
19. Linares-Vásquez, M.; Dit, B.; Poshyvanyk, D. An Exploratory Analysis of Mobile Development Issues Using Stack Overflow. In Proceedings of the 2013 10th Working Conference on Mining Software Repositories (MSR), San Francisco, CA, USA, 18–19 May 2013. [CrossRef]
20. Bajaj, K.; Pattabiraman, K.; Mesbah, A. Mining Questions Asked by Web Developers. In Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, Hyderabad, India, 31 May–1 June 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 112–121. [CrossRef]
21. Ponzanelli, L.; Bavota, G.; Di Penta, M.; Oliveto, R.; Lanza, M. Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter. In Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, Hyderabad, India, 31 May–1 June 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 102–111. [CrossRef]
22. Campbell, J.C.; Zhang, C.; Xu, Z.; Hindle, A.; Miller, J. Deficient Documentation Detection: A Methodology to Locate Deficient Project Documentation Using Topic Analysis. In Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, Francisco, CA, USA, 18–19 May 2013; IEEE Press: New York, NY, USA, 2013; pp. 57–60.
23. Wong, E.; Yang, J.; Tan, L. AutoComment: Mining Question and Answer Sites for Automatic Comment Generation. In Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering, ASE'13, San Francisco, CA, USA, 18–19 May 2013; IEEE Press: New York, NY, USA, 2013; pp. 562–567. [CrossRef]

24. Treude, C.; Robillard, M.P. Augmenting API Documentation with Insights from Stack Overflow. In Proceedings of the 38th International Conference on Software Engineering, ICSE '16, Austin, TX, USA, 14–22 May 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 392–403. [CrossRef]
25. Petrosyan, G.; Robillard, M.P.; De Mori, R. Discovering Information Explaining API Types Using Text Classification. In Proceedings of the 37th International Conference on Software Engineering, ICSE '15, Florence, Italy, 24 May 2015; IEEE Press: New York, NY, USA, 2015; Volume 1; pp. 869–879.
26. Le, L.T.; Shah, C.; Choi, E. Evaluating the Quality of Educational Answers in Community Question-Answering. In Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries, JCDL '16, Newark, NJ, USA, 19–23 June 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 129–138. [CrossRef]
27. Shah, C.; Pomerantz, J. Evaluating and Predicting Answer Quality in Community QA. In Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, Geneva, Switzerland, 25–30 July 2010; Association for Computing Machinery: New York, NY, USA, 2010; pp. 411–418. [CrossRef]
28. Maity, S.; Kharb, A.; Mukherjee, A. Language Use Matters: Analysis of the Linguistic Structure of Question Texts Can Characterize Answerability in Quora. In Proceedings of the Eleventh International AAAI Conference on Web and Social Media, Montréal, QC, Canada, 15–18 May 2017.
29. Mckinney, W. Pandas: A Foundational Python Library for Data Analysis and Statistics. *Python High Perform. Sci. Comput.* **2011**, *14*. Available online: https://www.dlr.de/sc/portaldata/15/resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf (accessed on 19 July 2020).
30. Behnel, S.; Faassen, M.; Bicking, I. lxml: XML and HTML with Python. 2005.
31. Bird, S. NLTK: The Natural Language Toolkit. In Proceedings of the COLING/ACL on Interactive Presentation Sessions, COLING-ACL '06, Sydney, Australia, 10 July 2006; Association for Computational Linguistics: Stroudsburg, PA, USA, 2006; pp. 69–72. [CrossRef]
32. Honnibal, M.; Johnson, M. An Improved Non-monotonic Transition System for Dependency Parsing. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September; Association for Computational Linguistics: Lisbon, Portugal, 2015; pp. 1373–1378. [CrossRef]
33. Anderson, A.; Huttenlocher, D.; Kleinberg, J.; Leskovec, J. Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, 12–16 August 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 850–858. [CrossRef]
34. Baltadzhieva, A.; Chrupała, G. Predicting the quality of questions on Stackoverflow. In Proceedings of the International Conference Recent Advances in Natural Language Processing, Hissar, Bulgaria, 5–11 September 2015; INCOMA Ltd. Shoumen, BULGARIA: Hissar, Bulgaria, 2015; pp. 32–40.
35. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.* **2009**, *11*, 10–18. [CrossRef]
36. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority over-Sampling Technique. *J. Artif. Int. Res.* **2002**, *16*, 321–357. [CrossRef]
37. Sebastiani, F. Machine Learning in Automated Text Categorization. *ACM Comput. Surv.* **2002**, *34*, 1–47. [CrossRef]
38. Mitchell, T.M. *Machine Learning*; McGraw-Hill: New York, NY, USA, 1997.
39. Zeng, Z.Q.; Yu, H.B.; Xu, H.R.; Xie, Y.Q.; Gao, J. Fast Training Support Vector Machines Using Parallel Sequential Minimal Optimization. In Proceedings of the 2008 3rd International Conference on Intelligent System and Knowledge Engineering, Xiamen, China , 17–19 November 2008. [CrossRef]
40. Frank, E.; Witten, I.H. Generating Accurate Rule Sets Without Global Optimization. In Proceedings of the Fifteenth International Conference on Machine Learning, ICML '98, Madison, WI, USA, 24–27 July 1998; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1998; pp. 144–151.
41. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32.[CrossRef]
42. Domingos, P. A Few Useful Things to Know about Machine Learning. *Commun. ACM* **2012**, *55*, 78–87. [CrossRef]

43. Kotsiantis, S. Supervised Machine Learning: A Review of Classification Techniques. *Inforatica* **2007**, *31*, 249–268.
44. Han, J.; Haihong, E.; Le, G.; Du, J. Survey on NoSQL database. In Proceedings of the 2011 6th International Conference on Pervasive Computing and Applications, Port Elizabeth, South Africa, 25–29 October 2011; IEEE: New York, NY, USA, 2011; pp. 363–366.