ESEC/FSE: U: Automated Generation of Programming Language Quizzes

Shuktika Jain
{Advisors: Rahul Purandare (IIITD) and Anita Sarma (Oregon State University)}
Indraprastha Institute of Information Technology Delhi
shuktika12163@iiitd.ac.in

ABSTRACT

Students and instructors often wish to take and give quizzes as part of their courses. Since quizzes are an important part of learning, formation of quizzes becomes a vital problem. This work aims to help users generate quizzes for programming languages in two steps. We identify the terms related to a topic and then use them for extraction of questions on the topic. We know that programming discussion forums and question-answer sites contain questions using programming terms in their posts. In this work, we mine patterns in user queries from such a forum and then automatically discover terms related to programming languages in those queries using the mined patterns. We use these terms to extract questions related to the programming language and form automated quizzes with a precision of **87%**.

Categories and Subject Descriptors

D.3.3 [**Software**]: Language Constructs and Features; I.2.7 [**Natural Language Processing**]: Text analysis; K.3.2 [**Computer and Information Science Education**]: Computer science education

Keywords

Quiz Creation, Pattern Mining, Programming Languages

1. PROBLEM AND MOTIVATION

Online tests mostly have either a fixed database of questions or manually composed quizzes. Moreover, majority of existing quizzes on programming languages focus on multiple choice questions or short answer questions like "*What is the output of the program?*" instead of questions like "*How to sort an array in place?*". Suppose Sally, an instructor, wants to form a quiz on basic programming for her class. For this, she has to first select topics like *array*, and then decide the questions on these topics.

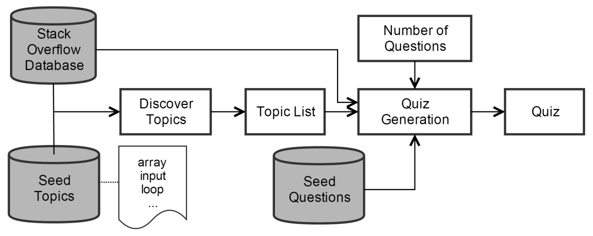

Figure 1: An overview of the approach followed to generate quizzes.

To create a quiz, Sally has to spend time and effort in selecting relevant topics and appropriate questions on these topics. For example, after careful deliberation, she may decide to form a quiz on *loops*. Next, she will spend time in finding good questions like "*When is it better to use a 'while' loop instead of a 'for' loop?*" to test the concept. A good quiz question for a programming language should be about a programming language topic, must be in the form of a question and also be a common query for users. This motivates us to reduce her effort spent on making a quiz by solving her problem using an automated procedure.

Our idea is to create automated quizzes in two parts: by finding programming language related terms like *array* and *loop* using a discussion forum, and using these discovered terms to select questions for quizzes. Users of discussion forums on computer programming use terms in their posts which are related to programming languages. We call these terms as topics for the language. For example, a user query "*How to call a function?*" contains two topics *call* and *function* as both terms are programming specific. We solve Sally's problem of finding terms related to a programming language by devising an approach for automatic discovery of topics. Once the terms are discovered, Sally can choose the topics out of this list and use our quiz generation algorithm to form a quiz automatically and save time.

Discovering topics requires certain patterns related to the topics to be extracted. Without these patterns, we might get terms which are not related to programming languages. Similarly, questions have patterns which separate them from sentences. A term-frequency analysis of the words in discussion forum's titles shows that a lot of top terms are simple English terms used for title formulation and are irrelevant to programming. Hence, finding well-formed questions on relevant topic is a challenging task.

Table 1: Discovered topics for the pattern "VBG DT NN(topic) IN NNS IN"[3]

Topic	Frequency
array	3
list	3
number	3
lot	2

Figure 1 shows an overview of the approach used to generate quizzes. In our work, property of same type of topics having similar attributes is exploited to extract patterns related to certain known topics which are input to the system as *seed* topics. For example, *collections* in Java have attributes like declaring and adding elements to the collection. The set of seed topics and discussion forum is thus input to the system to discover other unknown topics by matching the patterns. Now, Sally can use the seed "*array*" to find the topics "*list*", "*queue*", "*map*" etc.

As shown in Figure 1, the discovered topic list, seed questions from top websites containing frequent programming language questions[1] and discussion forum are input to the system to capture the factors of a good quiz question. The seed questions are converted into their part-of-speech (PoS) counterpart by tagging their words with *noun, verb, adjectives* etc. to extract common patterns followed by questions. The up-vote count of posts in discussion forum is used to incorporate the effect of frequently arising queries for users. The final list of questions is output to the user as a quiz.

StackOverflow[2] is a question-answer site on computer programming. We convert its titles into their PoS counterparts to mine patterns for the topics. To get a sense of the utility of mining PoS patterns from titles, we conducted a few experiments. We mined patterns of various lengths for seed topic *array* and individually used the mined patterns to discover unknown topics whose PoS titles matched the mined patterns. Table 1 shows the discovered topics for one such mined pattern with the number of times the topic matched the pattern, indicating 75% precision since the term "lot" is not related to programming languages as opposed to the other three terms. For other patterns that occurred with high frequency, the average precision was about 71%.

2. BACKGROUND AND RELATED WORK

2.1 Quiz Generation

Generation of automated quizzes or problems has been widely discussed in various works. Previous works have focused on extracting feature words on the topic from blogs for tourism quizzes [1] and creating multiple variations of same question for online object-oriented programming quizzes [2]. Programming-by-example has been used for test driven synthesis of code snippets [3]. Other problem generation works are for geometry [4], natural deduction [5] and algebra [6]. Though the importance of generating automated quizzes for programming languages has been stated, to the best of our knowledge, ours is the first attempt to solve this problem.

2.2 Discovering Terms

Product reviews from discussion forums and blogs have been used to discover product related terms [7]. We use the high-level approach by Ding et al. and modify the low-level components to adapt to discovering topics for programming languages. Code elements in forum posts have been discovered in [8] whereas, we additionally discover terms used by developers which may not be present in code. Links between e-mails and software artifacts have been established by Bacchelli et al. [9].

2.3 Part-of-speech Tagging

PoS tagging has been used in multiple works to mine patterns. Lexical relations have been found between software identifiers by tokenizing the words and using PoS tags as lexical categories in [10]. PoS tagger for program identifiers to improve software tools has been created in [11]. Question retrieval model for multi-sentence questions using PoS tagging has been proposed in [12].

3. APPROACH AND UNIQUENESS

As described in the previous section, quizzes have been formed for a variety of areas and PoS tagging has been used extensively. In this work, we apply part-of-speech tagging and pattern mining approach to generate quizzes in a different area, programming languages. We achieve this in two parts: discovering topics for the quizzes, and finding suitable questions by using the topics.

3.1 Discovering Topics

An overview of the approach used to discover topics is shown in Figure 2. We use a database of seed topics and StackOverflow posts to mine corresponding patterns using PoS tagging to discover more topics. To obtain the final topic list, a top-k cut-off is applied. Mining terms related to programming languages is a novel problem.

3.1.1 Part-of-Speech Tagging and Pattern Mining

Titles of StackOverflow posts on Java programming language are being used as the database of sentences. The algorithm first converts all the titles in the database into PoS counterparts[4]. Next, it replaces all PoS tags of seed topic words with the tag *TOPIC* to mark a topic. Using a proximity factor k, we mine patterns of length k by considering k tags around the tag *TOPIC*. We then store most frequent patterns as mined patterns for the seed topics. For example, if we have the title question "*How to declare a list?*", the PoS tagged title will be "*How/WRB to/TO declare/VB a/DT list/NN ?/.*"[5]. Now, if *list* is given to the system as a

[1]http://www.instanceofjava.com/2015/06/java-basic-interview-questions-freshers.html and http://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html

[2]http://stackoverflow.com/

[3]*VBG, DT, NN, IN, NNS* imply *verb, determiner, singular noun, preposition, plural noun* respectively. *NN(topic)* implies seed topic had *NN* tag.

[4]We use the PoS tagger by Stanford NLP group.

[5]The PoS tags *WRB, TO, VB, DT, NN* correspond to *adverb, the word "to", verb, determiner* and *noun* respectively.

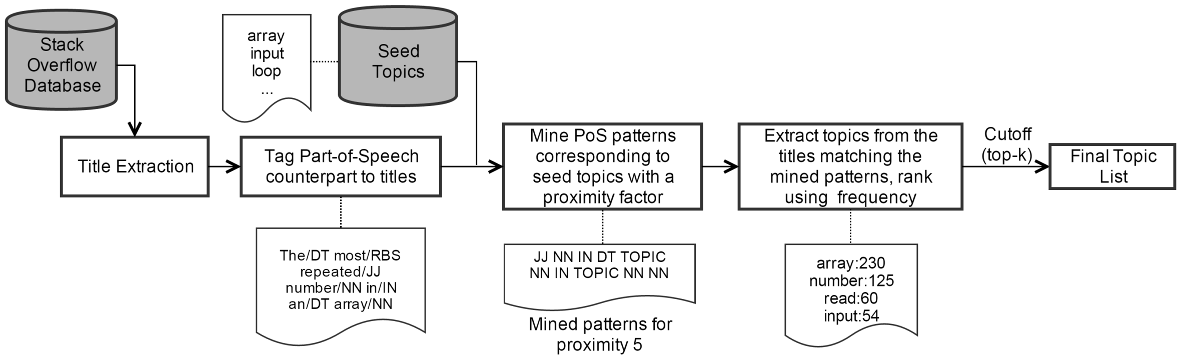
Stack Overflow Database
array input loop ...
Seed Topics
Title Extraction
Tag Part-of-Speech counterpart to titles
Mine PoS patterns corresponding to seed topics with a proximity factor
Extract topics from the titles matching the mined patterns, rank using frequency
Cutoff (top-k)
Final Topic List
The/DT most/RBS repeated/JJ number/NN in/IN an/DT array/NN
JJ NN IN DT TOPIC NN IN TOPIC NN NN
Mined patterns for proximity 5
array:230 number:125 read:60 input:54

Figure 2: A high level diagram of the approach followed to discover topics related to programming languages.

seed topic and proximity factor is 5, the tag *NN* is replaced by *topic* and the pattern "*WRB TO VB DT TOPIC*" is mined. This approach is unique as it uses proximity based pattern mining as opposed to sequential pattern mining.

3.1.2 Pattern Matching and Mining Topics

In the second pass over the database, the PoS pattern of each title is matched against the mined patterns. Any tag can match in place of the tag *topic*, but the remaining tags need to be exact matches in the same order. In our running example, if we consider another title "*How to change the datatype of an element?*", its tagged title will be "*How/WRB to/TO change/VB the/DT datatype/NN of/IN an/DT element/NN ?/.*". The mined pattern "*WRB TO VB DT TOPIC*" will match the PoS pattern "*WRB TO VB DT NN*" of this title. The term in a StackOverflow title having corresponding tag for *topic* is output as a candidate topic. For example, *datatype* will be output as a discovered topic in the present case.

For each pattern matched with PoS tagged title, frequency of the discovered topic increases. This frequency acts as a measure of confidence with which the word is considered a topic. We then list the discovered topics in descending order of their frequency. Out of these, the top-k topics are used to generate quizzes.

3.2 Quiz Generation

Given the number of questions to be generated for a quiz by Sally, our system outputs a quiz using three factors: discovered topics, patterns for questions, and popularity of the question.

3.2.1 Discovered Topics

From the list of candidate topics, the top-k terms as per decreasing order of frequency are taken. Since more number of questions would require larger variety, k is chosen depending on the number of questions required. The equation $k = f_1 * number\_of\_questions$ is used to decide the value of k, where f_1 has been empirically set to 5. The frequencies of top-k topics are then normalized to a scale of 0 to 100 using feature scaling by applying equation 1. Next, all StackOverflow titles containing at least one topic are extracted with frequencies based on the normalized frequencies of topics and weight assigned to discovered topics as a factor.

$$X' = 100 * \frac{(X - X_{min})}{(X_{max} - X_{min})} \tag{1}$$

3.2.2 Question Patterns

Once a list of narrowed down titles is created, question-like titles are figured out. A seed set of Java questions is used to mine question patterns. Each question has a component which makes the sentence question-like. For example, sentences containing *'What is...'*, *'Describe...'* and *'How to do...'* are questions. The seed set of questions are converted to their PoS counterparts to extract such patterns. From the PoS seed questions, all possible n-word patterns are mined with their frequencies. These frequencies are normalized to the same scale of 0 to 100. The question patterns are then matched against the titles and frequencies of the titles are incremented depending on the frequency of question pattern and weight for question patterns.

3.2.3 Question Popularity

StackOverflow allows users to up-vote questions. The last factor, popularity of a title, is captured using the number of up-votes for the title. All the up-votes for narrowed down titles are taken from the database and again normalized to a scale of 0 to 100. For each title, frequency is increased by the number of normalized up-votes weighted by the factor for popularity of question.

3.2.4 Selecting Quiz Questions

Combining the three factors i.e. discovered topics, question patterns and question popularity, the equation below computes the frequencies of titles. Top-k questions, depending on number of questions required, are taken as candidate questions in terms of decreasing frequency. To generate unique quizzes every time, the system randomly picks the required number of questions from the list of candidate questions and outputs them as a quiz.

Table 2: Average precision at different levels of top-k with two sets of seed topics.

No. of seeds	top-10	top-15	top-20	top-35	top-50
1	0.70	0.66	0.60	0.52	0.51
11	0.80	0.69	0.56	0.54	0.54

$$Title_{freq} = w_t * f_t + w_p * \sum_{p_i \in P_m} f_{p_i} + w_v * f_v$$

where,

w_t, w_p, w_v = weights for topics, question patterns and votes respectively

f_t, f_v = Normalized frequency of topics, votes

P_m = Matched patterns for the title

f_{p_i} = Normalized frequency of pattern p_i

4. RESULTS AND CONTRIBUTIONS

4.1 Results

4.1.1 Discovering Topics

For discovering topics related to programming languages, we used two sets of seed topics for evaluation, the first containing just one seed topic *array* and the other containing 11 seed topics related to *collections*. Table 2 shows the precision i.e. the percentage of relevant topics at top-10, 15, 20, 35 and 50 for patterns with proximity ranging from 3 to 5. Precision at top-10 was **80%** for a proximity factor of 3 with the bigger topic set. Sally can now use the top-k discovered topics and mine the questions for quiz.

4.1.2 Quiz Generation

We performed a user study for quiz generation on 5 participants. All 5 participants have been teaching assistants and have set quizzes for some courses. We generated 3 Java quizzes by varying the weights of three main factors stated in the approach and performed a within-group evaluation using Latin square design to mitigate the learning effect. Two metrics - precision and usefulness - were used as a measure of quiz performance. Precision is the number of questions that are related to the programming language Java out of all questions output in the quiz. For usefulness, we asked the users to mark how useful do they believe the question is in testing a student's concept for a basic Java course. Usefulness was measured on a Likert scale of 1 to 5 where 1 represents least useful and 5 represents most useful.

From the results of discovering terms related to programming languages, we set the number of words n for question patterns to 3 and k to 5. Number of questions were fixed to 10 per quiz. Three quiz variants with more weight assigned to either topics, question patterns or votes in each one were generated and used for evaluation.

Table 3 shows the precision for each participant for every quiz. Average precision across all quizzes is **87%**. For quiz variants A, B and C, the average precision is **90%**, **86%** and **84%** respectively.

Table 3: Weights of the factors and precision for all participants, for each quiz.

Quiz	w_t	w_p	w_v	P_1	P_2	P_3	P_4	P_5
A	0.50	0.25	0.25	0.90	0.90	0.90	0.90	0.90
B	0.25	0.50	0.25	0.90	0.80	0.70	1.00	0.90
C	0.25	0.25	0.50	0.60	0.90	0.90	1.00	0.80

Figure 3 shows the number of quiz questions at each level of Likert scale for each quiz for the questions marked as related to Java. High frequency of questions for 4 and 5 shows that users believe most of the questions to be useful for a basic programming course. In coherence with precision, quiz variant A has the highest number of useful questions. These results suggest that users find a lot of automatically generated questions as useful and relevant to Java language. It was also observed that quizzes generated by assigning more weight to topics have a better precision.

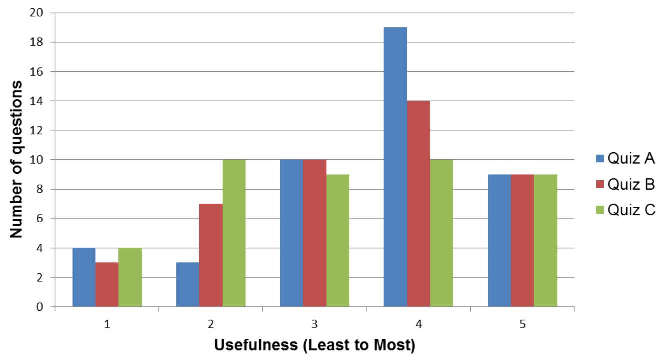

Figure 3: Usefulness of quiz questions

Following quiz containing 5 questions was generated using our approach for use by Sally for her class:

- How would I read only the first word of each line of a text file in java?
- How do I read in an input file with a scanner or buffered reader and count all the occurrences of a specific character in the input file?
- Why is an array not assignable to Iterable?
- How can I print a new line in a toString() method in java?
- Can I Pass a return value of a getter method to a setter method of another object in a different class?

4.2 Contributions

Our main contribution is to solve the problem of automatically finding programming language quiz questions. We do this by discovering terms related to programming languages. Instructors and students can use our system to generate quizzes and enhance learning. In the future, we hope to include more features like mining answers from the posts and categorizing the questions into difficulty levels.

5. ACKNOWLEDGMENTS

I would like to thank my advisors Dr. Rahul Purandare (IIITD) and Dr. Anita Sarma (Oregon State University) for their valuable guidance. I would also like to thank my colleagues Venkatesh Vinayakarao (IIITD) and Saumya Jain (IIITD) for their worthy suggestions and support.

6. REFERENCES

[1] J. Zeng, T. Sakai, C. Yin, T. Suzuki, and S. Hirokawa, "Automatic generation of tourism quiz using blogs," *Artificial Life and Robotics*, vol. 17, pp. 412–416, February 2013.

[2] I.-H. Hsiao, P. Brusilovsky, and S. Sosnovsky, "Web-based parameterized questions for object-oriented programming," in *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pp. 3728–3735, 2008.

[3] D. Perelman, S. Gulwani, D. Grossman, and P. Provost, "Test-driven synthesis," in *Proceedings of the 35th ACM SIGPLAN conference on Programming language design and implementation*, pp. 408–418, 2014.

[4] C. Alvin, S. Gulwani, R. Majumdar, and S. Mukhopadhyay, "Synthesis of geometry proof problems," in *AAAI Conference on Artificial Intelligence*, 2014.

[5] U. Z. Ahmed, S. Gulwani, and A. Karkare, "Automatically generating problems and solutions for natural deduction," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pp. 1968–1975, 2013.

[6] R. Singh, S. Gulwani, and S. Rajamani, "Automatically generating algebra problems," in *AAAI Conference on Artificial Intelligence*, 2012.

[7] X. Ding, B. Liu, and L. Zhang, "Entity discovery and assignment for opinion mining applications," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (New York, NY, USA), pp. 1125–1134, 2009.

[8] P. C. Rigby and M. P. Robillard, "Discovering essential code elements in informal documentation," in *Proceedings of the 2013 International Conference on Software Engineering*, pp. 832–841, 2013.

[9] A. Bacchelli, M. Lanza, and R. Robbes, "Linking e-mails and source code artifacts," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, pp. 375–384, 2010.

[10] J.-R. Falleri, M. Huchard, M. Lafourcade, V. P. C. Nebut, and M. Dao, "Automatic extraction of a wordnet-like identifier network from software," in *Proceedings of the 18th IEEE International Conference on Program Comprehension*, (Braga, Minho), pp. 4–13, 2010.

[11] S. Gupta, S. Malik, L. Pollock, and K. Vijay-Shanker, "Part-of-speech tagging of program identifiers for improved text-based software engineering tools," in *Proceedings of the 21st Annual International Conference on Program Comprehension*, (San Francisco, CA, USA), pp. 3–12, May 2013.

[12] K. Wang, Z.-Y. Ming, X. Hu, and T.-S. Chua, "Segmentation of multi-sentence questions: towards effective question retrieval in cqa services," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 387–394, ACM, 2010.