Automated Generation of Programming Language Quizzes

Shuktika Jain
Indraprastha Institute of Information Technology Delhi
New Delhi, India
shuktika12163@iiitd.ac.in

ABSTRACT

Formation of quizzes is a vital problem as they are an important part of learning. To create a quiz on a particular topic, its related terms need to be identified for further use in extraction of questions on the topic. These terms are referred to as *entities* for the topic and the task of distinguishing entities from general purpose terms is termed *entity discovery.* We know that discussion forums and question-answer sites on software 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 entities for programming languages using these patterns. We use these entities to extract questions related to the programming language and form automated quizzes using them.

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, Entity Discovery, 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 or short answer questions like "*What is the output of the program?*" instead of questions like "*When is it better to use a 'while' loop instead of a 'for' loop?*". Suppose Sally, an instructor, wants to have a quiz on basic programming for her class. She has to select topics for the quiz and then decide the questions on these topics. This motivates us to solve her problem in two parts: by finding programming language related terms like *array* with the help of a discussion forum, and using these discovered terms to select questions. Sally can now choose the terms out of discovered ones and specify number of questions for automatic generation of quiz.

Users of discussion forums on computer programming use terms in their posts which are related to the subject. We call these terms as entities for the programming language. For example, a user query "*How to call a function?*" contains two entities *call* and *function* as both terms are programming specific. Sally needs to automatically mine different entities without querying about each term. This problem of identifying entities is termed *entity discovery.* We solve the problem of finding terms related to a programming language by devising an approach for automatic discovery of entities.

In our work, property of same type of entities having similar attributes is exploited to extract patterns related to certain known entities which are input to the system as *seed* entities. For example, *collections* in Java have attributes like declaring and adding elements to the collection. Other unknown entities are then discovered by matching them with the mined patterns. Now, Sally can use the seed "*array*" to find the entities "*list*", "*queue*", "*map*" etc. which then translate to the quiz questions as described in Section 3.

To capture common patterns from sentences with similar structures, we convert the titles of StackOverflow posts into their part-of-speech (PoS) counterparts by tagging the title words with *noun, verb, adjectives* etc. We use a converted PoS title sequence as a pattern containing the entity. 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 entity *array* and individually used the mined patterns to discover unknown entities whose PoS titles matched the mined patterns. Table 1 shows the discovered entities for one such mined pattern with the number of times the entity matched the pattern, indicating 75% precision. For other patterns that occurred with high frequency, the average precision was about 71%.

Table 1: Discovered Entities for the pattern "VBG DT NN(entity) IN NNS IN"[1]

Entity	Frequency
array	3
list	3
number	3
lot	2

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

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
ACM. 978-1-4503-3675-8/15/08
http://dx.doi.org/10.1145/2786805.2807559

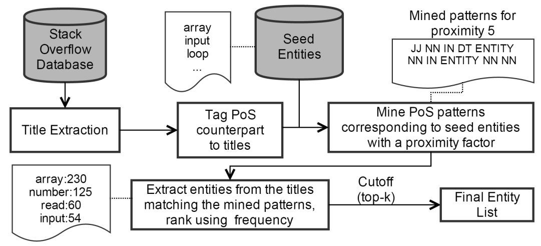
Stack Overflow Database
array
input
loop
...
Seed Entities
Mined patterns for proximity 5
JJ NN IN DT ENTITY
NN IN ENTITY NN NN
Title Extraction
Tag PoS counterpart to titles
Mine PoS patterns corresponding to seed entities with a proximity factor
array:230
number:125
read:60
input:54
Extract entities from the titles matching the mined patterns, rank using frequency
Cutoff (top-k)
Final Entity List

Figure 1: A high level diagram of the approach followed to discover entities for source code

2. BACKGROUND AND RELATED WORK

Approaches for automated quizzes focused on extracting feature words on the topic from blogs for tourism quizzes [1] and creating variations of same question for object-oriented programming quizzes [2]. Programming-by-example has been used for test driven synthesis of code snippets [3]. Other works are on generating problems for geometry [4], natural deduction [5] and algebra [6]. For entity discovery, product reviews from discussion forums and blogs have been used [7]. Links between e-mails and software artifacts have been established by Bacchelli et al. [8]. PoS tags have been used to find lexical relations between software identifiers [9]. Code elements in forum posts have been discovered in [10] whereas we also discover terms used by developers which may not be present in code. We use the high-level approach by Ding et al. and modify the low-level components to adapt to entities for programming languages. To the best of our knowledge, ours is the first attempt on automated quiz generation for programming languages.

3. APPROACH AND UNIQUENESS

Figure 1 gives a high-level idea of the approach used to discover entities. Using a database of seed entities and StackOverflow posts, corresponding patterns are mined to discover more entities. The final entity list is obtained after applying a top-k cut-off. Questions containing top-k entities are then randomly picked from the database to form quizzes. This paper focuses largely on entity discovery part. Adapting entity discovery to source code is a novel problem.

Problem Statement: Given a set of seed entities ($e_1, e_2, ..., e_n$) and a database of sentences, mine patterns related to the seed entities and discover a bigger set of entities for source code being discussed in the sentences. Using this set of entities, mine questions from database and create quizzes.

Part-of-Speech Tagging and Pattern Mining: Titles of StackOverflow posts on Java programming language have been used as the database of sentences. The algorithm first converts all the titles in the database into PoS counterparts[2]. Next, it replaces all PoS tags of seed entity words with the tag *ENTITY* to mark an entity. Using a proximity factor k, patterns of length k are mined by considering k tags around the tag *ENTITY*. Patterns with frequency greater than one are then stored as mined patterns for the seed entities. 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 ?/.*"[3]. Now, if *list* is given to the system as a seed entity and proximity factor is 5, the tag *NN* will be replaced by *ENTITY* and the pattern "*WRB TO VB DT ENTITY*" will be mined. Our entity mining approach is unique as it uses proximity based pattern mining.

[2]We have used the PoS tagger by Stanford NLP group.

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

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

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

Pattern Matching and Entity Discovery: 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 *ENTITY*, but the remaining tags need to be exact matches in the same order. The term in a StackOverflow title having the tag in place of *ENTITY* is output as a candidate entity with frequency stating the number of patterns matched for that term. This frequency acts as a measure of confidence with which the word is considered as an entity. In the above example, if we consider another title "*How to change the datatype of an element?*", its tagged title will be "*How/WRB to/TO declare/VB a/DT list/NN ?/.*". The mined pattern "*WRB TO VB DT ENTITY*" will match the PoS pattern "*WRB TO VB DT NN*" of this title and *list* will be output as a discovered entity.

Quiz Creation: Top-k entities as per frequency are given to Sally as suggestions for commonly discussed topics on programming languages. In case, she does not choose topics, entire top-k list is used for quiz generation. The system randomly chooses titles from StackOverflow containing any term from the topic list given and outputs the posts as quiz questions. Automatic generation of quizzes for programming languages with new questions each time is also unique.

4. RESULTS AND CONTRIBUTIONS

We used two sets of seed entities for evaluation, the first containing just one seed entity *array* and the other containing 11 seed entities related to *collections*. For factors of proximity ranging from 3 to 5, precision at top-10, 15, 20, 35 and 50 were computed. The percentage of relevant entities, i.e. precision at top-10 was 80% for proximity factor 3 with the bigger entity set. Table 2 shows the average precision at various levels of top-k with both sets of seed entities. Sally can now use the top-10 discovered entities and mine the questions. Following quiz containing 3 questions generated using our approach can be used by Sally:

- How to detect a loop in a linked list?
- Why is an array not assignable to Iterable?
- How to parse string to array?

Our main contribution is to solve the problem of automatically finding programming language quiz questions. We do this by adapting the entity discovery approach to source code. Instructors and students can use it for quizzes and better learning. In the future, we hope to improve the precision and include more features like mining answers from the posts and categorizing the questions into difficulty levels.

5. ACKNOWLEDGMENTS

I thank my advisor Rahul Purandare (IIITD), Anita Sarma (University of Nebraska-Lincoln) and Venkatesh Vinayakarao (IIITD) for their valuable guidance 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] 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.

[9] 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.

[10] 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.