Learnersourcing Subgoal Hierarchies of Code Examples

Hyoungwook Jin
School of Computing, KAIST
Daejeon, South Korea
jinhw@kaist.ac.kr

Juho Kim
School of Computing, KAIST
Daejeon, South Korea
juhokim@kaist.ac.kr

ABSTRACT

A subgoal is a unit that groups a set of steps by their functions in a problem-solving procedure, such as cooking, how-to's and programming. Studies showed that learning hierarchical subgoal structures of worked examples can aid transfer in learning. To support subgoal learning at scale, we need to generate subgoal hierarchies that consist of both the goal *structures* and *labels*. While prior work [3, 8] has focused on using learnersourcing to generate high quality subgoal labels at scale, generation of hierarchical subgoal structures had little attention and has been done manually by domain experts. Generation of hierarchical subgoal structures is especially challenging for both AIs and crowdworkers because it requires comprehensive understanding of the entire problem-solving procedure. In order to enable subgoal hierarchy generation at scale without expert interventions, we propose a novel learnersourcing workflow that combines learners' local understanding of subgoal structures into multi-granular subgoal hierarchies.

CCS CONCEPTS

• **Human-centered computing → Interactive systems and tools**.

KEYWORDS

learnersourcing, subgoal learning, hierarchy generation

ACM Reference Format:
Hyoungwook Jin and Juho Kim. 2022. Learnersourcing Subgoal Hierarchies of Code Examples. In *Proceedings of Learning @ Scale workshop (L@S)*. ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Code examples on online tutorials and QnA websites often lack explanations of code adaptive to learners with diverse levels of expertise. Due to lack of detailed explanations, novice programmers often struggle to recognize solution structures of the examples and tend to memorize the examples as a whole. However, memorizing code without understanding code structures hinder learners from transferring to novel problem contexts [1].

Subgoal learning can help learners understand code structures and transfer to novel problems. A subgoal is a meaningful unit that groups a set of related code snippets by their function. Subgoals in code often form a hierarchical structure, top being high-level goals that explain the function of many codes altogether and bottom being low-level goals that explain codes line by line. We call this hierarchical structure a *subgoal hierarchy* (see Figure 1). Subgoal hierarchies can be used to hint solution structures of code examples [1] or to provide pedagogical feedback in subgoal learning activities [5, 7]. Multi-granular goals throughout subgoal hierarchies can also be used to generate explanations adaptive to learners' needs.

However, generating subgoal hierarchies at scale is challenging. Expert-driven methods are time-consuming and require multiple domain experts [2]. Automatic generation methods also seem infeasible due to low accuracy and lack of huge datasets for training AI models. Human-machine hybrid methods have been investigated to achieve the best of both worlds. Weir et al. [8] showed that machine-coordinated learnersourcing can effectively generate high quality subgoal labels for how-to videos at scale. Furthermore, Choi et al. [3] confirmed that the microtasks for learnersourcing subgoal labels can indeed be pedagogically helpful. Despite these findings, the prior learnersourcing workflows are limited to generate subgoal hierarchies because structure of the hierarchies need to be fed by experts.

In order to enable end-to-end generation of subgoal hierarchies at scale without expert interventions, we propose a novel learnersourcing task design and a computational pipeline to generate subgoal hierarchies. We designed a subgoal learning activity in which learners study code examples by grouping code lines and identifying subgoals on their own. Our computational pipeline then evaluates the code groups and constructs subgoal hierarchies.

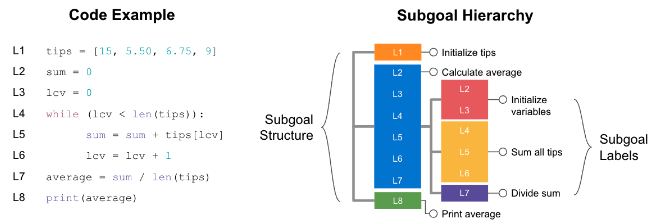

Figure 1: A subgoal hierarchy is a tree whose nodes consist of code groups and a subgoal label. While prior subgoal learnersourcing workflows focused on generating high quality subgoal labels given subgoal structures, our work focuses on generating the structure itself.

2 DESIGN GOALS

G1. Design a workflow that can generate correct and multi-granular subgoal hierarchies

Generation of subgoals is essentially an ill-posed problem because a code example may have multiple correct subgoal hierarchies

L@S, June 01–03, 2022, Roosevelt Island, NYC

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

that have different hierarchical structures and labels. Among many correct subgoal hierarchies, we specifically target to generate hierarchies that have many levels of granularity in goals because multi-granular subgoals are useful for producing adaptive instructional aids [5]. Hence, our workflow should generate subgoal hierarchies that are not only correct but also have multi-granular subgoals.

G2. Design motivational human computation tasks that can benefit crowdworkers

In order to generate subgoal hierarchies at scale, we need a monetarily sustainable approach. Prior work [8] showed that learnersourcing is a sustainable crowdsourcing method that provides intrinsic motivations and encourages voluntary participation of crowdworkers. Learnersourcing systems also have shown that learners can contribute to generating expert-quality data [3, 4, 9]. Hence, we use learnersourcing for generating subgoal hierarchies at scale, and we aim to design tasks that help learners study code examples and subgoals so that voluntary participation is elicited.

3 WORKFLOW

Our workflow is composed of 1) a subgoal generation task and 2) a computational pipeline to construct subgoal structures. In the generation task, we ask learners to group code lines by their goals. We expect that even without guidance, learners will recognize different solution structures in terms of granularity of goals [6]. While novices group code by superficial functions of code, more skilled learners may recognize higher level goals and group code in bigger chunks. After collecting code groups that vary in granularity from the task, our computational pipeline constructs subgoal hierarchies by stacking different code groups from low-level to high-level.

3.1 Subgoal Generation Task

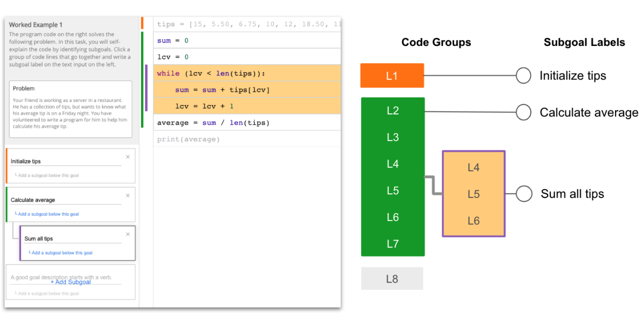

Figure 2: The user interface for subgoal generation task. Learners select the lines that share a common goal, and then explain the goal by writing it on the input box on the left.

We referred to the unguided constructive method of subgoal learning [7] for our task design. Although the constructive method is best-practiced with guidance or correct response feedback, we decided not to utilize data from peer learners during the collection in order to keep each session independent and make learners less susceptible to possibly poor data. Nevertheless, we believe that the subgoal generation task can encourage learners to recognize and self-explain goal structures of code examples.

The user interface is designed to support the generation of hierarchical subgoal structures (see Figure 2). Learners can add subgoals below other subgoals as far as they recognize hierarchical goal structures, and each pair of a subgoal label and a code group is color-coded to show clear mappings between them.

3.2 Subgoal Hierarchy Construction

Our computational pipeline constructs hierarchies by stacking learner-submitted code groups. Two code groups may conflict and cannot coexist in a hierarchy if they partially overlap (see the green and purple code groups in Figure 3). This happens when there are multiple ways to organize subgoal structures, and learners submit code groups that belong to different structures. In this case, we favor the code group that is more likely to be correct, and we use the number of identical submissions to determine the correctness of a code group. We chose a majority agreement scheme because we assume that the majority of learners are capable of identifying correct subgoals. When two conflicting code groups have the same number of submissions, the pipeline choose more inclusive code group in order to include more code groups in resulting hierarchies.

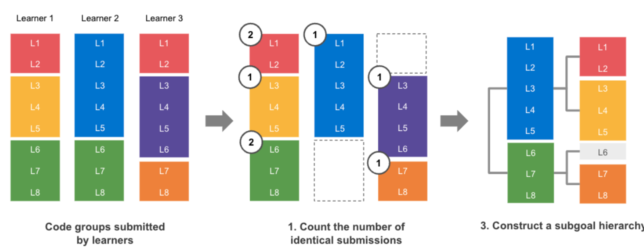

Figure 3: The pipeline constructs subgoal hierarchies by adding code groups, in the decreasing order of the number of submissions (noted in white circles). Code groups that have conflicts with pre-added code groups are left out to keep the integrity of entire hierarchies.

REFERENCES

[1] Richard Catrambone. 1995. Aiding subgoal learning: Effects on transfer. *Journal of educational psychology* 87, 1 (1995), 5.
[2] Richard Catrambone. 2011. Task analysis by problem solving (TAPS): Uncovering expert knowledge to develop high-quality instructional materials and training. In *Learning and Technology Symposium, Columbus, GA*.
[3] Kabdo Choi, Hyungyu Shin, Meng Xia, and Juho Kim. 2022. AlgoSolve: Supporting Subgoal Learning in Algorithmic Problem-Solving with Learnersourced Microtasks. (2022).
[4] Elena L Glassman, Aaron Lin, Carrie J Cai, and Robert C Miller. 2016. Learnersourcing personalized hints. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*. 1626–1636.
[5] Hyoungwook Jin, Minsuk Chang, and Juho Kim. 2019. SolveDeep: A System for Supporting Subgoal Learning in Online Math Problem Solving. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6.
[6] How People Learn. 2000. Brain, mind, experience, and school. *Committee on Developments in the Science of Learning* (2000).
[7] Lauren E Margulieux and Richard Catrambone. 2019. Finding the best types of guidance for constructing self-explanations of subgoals in programming. *Journal of the Learning Sciences* 28, 1 (2019), 108–151.
[8] Sarah Weir, Juho Kim, Krzysztof Z Gajos, and Robert C Miller. 2015. Learnersourcing subgoal labels for how-to videos. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*. 405–416.
[9] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z Gajos, Walter S Lasecki, and Neil Heffernan. 2016. Axis: Generating explanations at scale with learnersourcing and machine learning. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. 379–388.