# On Domain-Agnostic Approaches to Hint Generation using Collaborative Filtering

Daniel M Swoboda

Department of Computer Science, RWTH Aachen University, Aachen, Germany
daniel.maximilian.swoboda@rwth-aachen.de

**Abstract.** Several approaches for hint generation in educational programming have been introduced. Some have focused on the usage of collaborative filtering algorithms to generate hints from the crowd of users. This publication discusses the potential of applying known algorithms and concepts from programming education hint generation to other educational domains. It introduces the basics of collaborative filtering and intelligent tutoring systems and how those are used for automatic hint generation. Furthermore, intelligent tutoring systems that make use of collaborative filtering are introduced and categorized. This is then used to discuss which categories of intelligent tutoring systems could benefit directly from existing algorithms for hint generation. Especially, the requirements for the introduction of these algorithms are discussed.

**Keywords:** Collaborative Filtering · Intelligent Tutoring Systems · Novice Programming Environments · Domain Agnostic Educational Systems.

## 1 Introduction

Online environments for educational purposes are commonly used. They provide a system through which a user can access learning materials, complete electronic tests or communicate with teachers and other students. However, these technologies suffer from a lack of personalized supervision and feedback which are crucial for the success of students. [4] As a result, Intelligent Tutoring Systems (ITS) have emerged in recent years; such systems have the aim to provide adaptive and personalized guidance through course material and are usually based on expert-crafted models specifying what content to teach and how to teach it [7]. The unstructured and unquantifiable nature of educational data sets a high barrier for algorithms to effectively replace these expert-crafted models, with exercises being the most difficult to handle. [12] Over the past two decades, collaborative filtering (CF) has emerged as a recommender system technology able to create predictions on information based on similar characteristics of user-behavior. Such systems rely on the behavior and preferences of other users to create suggestions for users with similar behavior and preferences. Due to their design, approaches based on collaborative filtering only require information on how the data is used—in contrast to other approaches in the field—making it ideal for applications that handle unquantifiable or unprocessable data. [14] There has been recent research into the application of CF as a replacement for expert-crafted models in education for some specific ITS use cases (such as hint generation) [12]. One notable topic of research in this area is programming education for novices with the aim being the provision of instant feedback and guidance whilst a student is solving a programming exercise. [12]

This publication addresses the state of the art of intelligent tutoring systems and the usage of collaborative filtering in education, with a focus on programming education. It further tries to create a classification of ITS and suggests areas in which technology and research from programming education can be transferred to other educational systems.

Firstly, the concepts of collaborative filtering and intelligent tutoring systems are introduced. Then, an overview of ITS which make use collaborative filtering is given, common design decisions and usage scenarios are discussed and a categorization is made. Furthermore, the application of collaborative filtering in programming education is introduced by highlighting the advantages and limitations two specific and recent implementations. Finally, the possibility of conveying the principles and algorithms developed for programming ITS to other domains of teaching are discussed and challenges are highlighted.

## 2 Background

### 2.1 Collaborative Filtering

Recommender systems are software that suggest or recommend items to users. As such one of their purposes is to filter down massive data to a consumable and fitting selection. One possible classification of recommender systems introduces three categories: Non-personalized systems that recommend one or several items to all users; Content-based filtering, filters items for a user based on their content and the content of previously rated items by the user; Collaborative filtering, which uses historic interaction data of all users to predict the preferences of a specific user based on their similarity to other users. [15]

Collaborative filtering can be narrowly defined and categorized into two sub categories: neighborhood based algorithms and matrix factorization algorithms. Whilst the former calculates the similarity between system users or items, the latter uses techniques from linear algebra to analyze the user/item ratings data. In a broader sense, collaborative filtering can be any method of suggestion creation based on the data produced by users of the system. [14]

Figure 1 highlights the basic principle of collaborative filtering in a scenario with three different users of a system for travel suggestions. For three given travel destinations (China, USA, Italy) each user can select "like" or "dislike". The system tries to make a suggestion for user A who has not entered their opinion for the destination "USA". It now tries to find the closest match to user A, which is user B as all their selected preferences match up. Since user B liked the destination "USA", it now gives a positive suggestion to user A for that location.

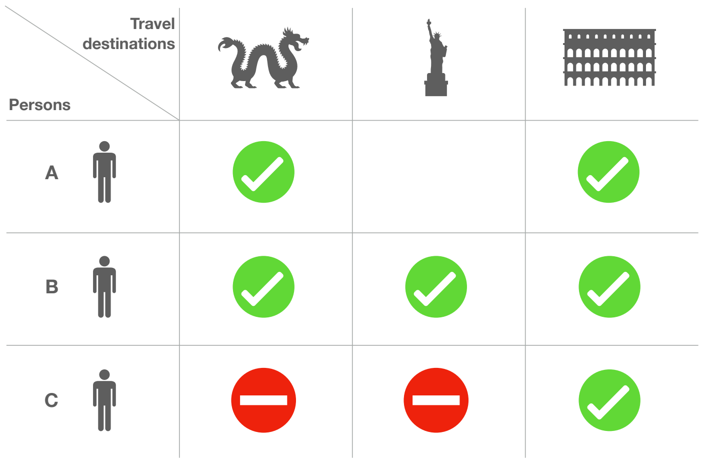

**Fig. 1.** Example of collaborative filtering. Three users and three travel destinations and the selected preferences are displayed. Based on this data the system can suggest the USA as a potential travel destination for person A.

Because of its simple design, collaborative filtering found widespread adoption in social web applications as well as e-commerce settings. [14]

### 2.2 Intelligent Tutoring Systems

Intelligent tutoring systems (ITS) are computer-based instructional systems that make use of models that specify the content that is taught and the strategy that is used to teach them. They firstly were used in classroom settings in the 1990s where they showed effectiveness. While the nature and virtue of ITS differs, they follow the same fundamental principle of creating individualized instructions and tasks to closely approach the benefits of competent human tutors. This is achieved by dynamically adapting a model of the student to their level of mastery. [7] One of the key features of any ITS is the ability to give students individualized and instant feedback during the problem solving process [10].

Figure 2 depicts a typical workflow of an intelligent tutoring system as it is used by a student. The tutee interacts with the instructions and tasks it gets delivered by the ITS, these help the student to better understand a topic. Student performance then is used to update the model representing their level of mastery. An instructional model then uses this information to pick or even generate the right content through the content model. Finally a new set of personalized

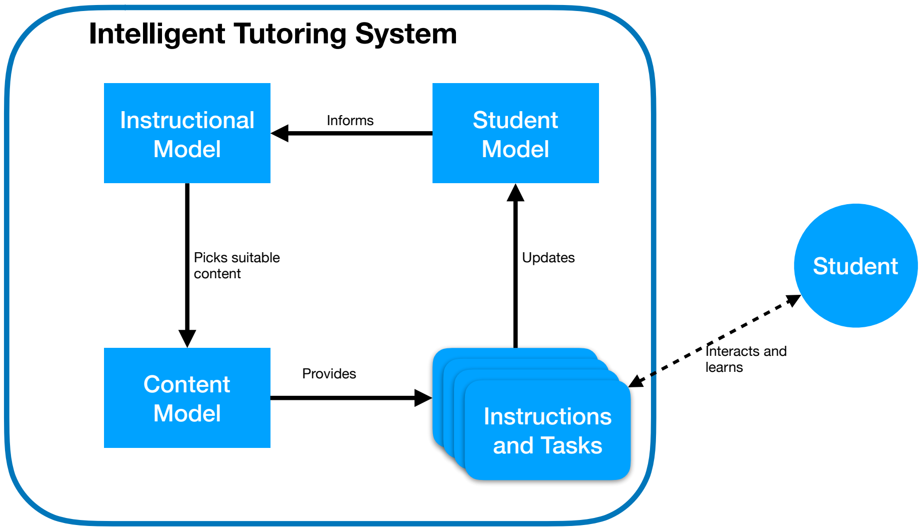

**Fig. 2.** Diagram of the typical workflow within an ITS. The depicted system is made up of a student model, an instructional model and a content model as well as a set of instructions and tasks. Based on [8]

instructions is provided to the student. A function of the system to help the user with feedback and guidance while working on a task could also be a part.

## 3 Application-Domains of Collaborative Filtering Intelligent Tutoring Systems

Because of their broad definition as tools that support learning, ITS have been implemented in a variety of different ways. In the past, attempts have been made to incorporate collaborative filtering into ITS as a way of replacing or enhancing the expert-crafted models. The resulting implementations are specific to the tasks and goals of the platform they were designed for and feature a specific set of limitations. The broad architecture of domain/content model, tutoring/instructional model and student model is transcended by most realizations of ITS. However, it still can be used to broadly categorize functionalities of tutoring systems. [8]

This publication focuses on instruction models, which receive input from the content and student models to make tutoring decisions. Doing so, the tutoring model ultimately is the teaching agent the students interact with, get their tasks and are provided with hints from. [8] Because of the broadness of implementations, several exemplary ITS have been picked.

### 3.1 Example Application-Domains

The given ITS were selected based on their different scopes, their usage—or potential usage—of collaborative filtering and their recentness.

**ITAP** is a computer programming ITS which supports novice programmers by providing hints—which are selected through collaborative filtering—while students write code to solve a task. It is designed to work with Python programs. [13]

**iSnap** is a computer programming ITS similar to ITAP for graphical programming in Snap which is an extension for Scratch programming environment. [10]

**ForMath** is a mathematics ITS which makes use of a hint-based educational model to teach students how to solve equations in multiple steps. It relies on pre-generated set of problems and solutions. [3]

**Crystal Island** is a game-based interactive intelligent tutoring system for literacy education. It features an interactive story with integrated tests, which are selected using collaborative filtering based on a model of the student's current level of virtue and the best existing path already visited by a different user with similar levels of virtue. It makes use of combined single-step tasks. [6]

**Thermo-Tutor** is an ITS for teaching closed-cylce thermodynamics. Each task is comprised of a diagram that needs to be drawn and a set of variables that need to be calculated by the student. The solution is compared to correct ones with a pre-created knowledge database. [1]

**Module Advisor** is a hybrid-recommender system for course-modules. It makes use of past student's elective modules to help and guide students through the process of electing their own modules. It incorporates student's own history and preferences in this process. In this context it is broadly classified as tutoring system but does not meet all criteria. [5]

### 3.2 Cross-Domain Categorization of ITS

In order to categorize ITS based on their tutoring model their type of guidance and the kind of solutions they work with are viewed. Figure 3 depicts the suggested categorization of ITS based on the kind of guidance they utilize and shows the placement of the introduced ITS within this categorization. The categorization itself is based on two factors: task-based guidance vs. curriculum-based guidance and answer-based tasks vs. multi-step-solution tasks. This model is not intended to represent all forms of ITS but rather tries to create a broad classification to differentiate between the focus of tasks and support different ITS provide. For the intersection of the categories "curriculum-based guidance" and "multi-step-solution tasks" none of the introduced systems qualified.

| Guidance / Tasks | Task-based guidance | Curriculum-based guidance |
|---|---|---|
| Multi-step solutions | ITAP<br>iSnap<br>ForMath | - |
| Answer-based solutions | Thermo Tutor | Module Advisor<br>Crystal Island |

**Fig. 3.** The categorization scheme based on the guidance and task categories. The previously introduced ITS are categorized according to this scheme. Highlighted in blue are the programming ITS which are analyzed in section 4.

**Task-based guidance vs. Curriculum-based guidance.** With task-based guidance the focus of the ITS is to guide the student through a specific task and find the correct solution. Curriculum-based guidance systems on the other hand suggest certain successions of tasks in order to achieve a greater curricular goal.

**Answer-based tasks vs. Multi-step-solution tasks.** Answer-based tasks are tasks that require simple answer formats like ”yes or no”, ”choosing the right box”, or ”inserting the right word/number” into a text field. Multi-step-solution tasks are those where the solution itself is comprised of multiple steps and whose order is relevant to the correctness of the solution. These are often linear successions of solution steps that transform the input to the correct answer.

## 4 Collaborative Filtering in Programming Education

Within the introduced categorization, ITS in the categories ”task-based guidance” and ”multi-step solution” are arguably the most complex ones to build. However, systems like ITAP and iSnap show that collaborative filtering can be successfully used to automate the guidance process by generating hints for tasks in the domain of programming education. The concept of intelligent tutoring systems was first applied to this area as a result of students seeking immediate feedback. Systems that support novice programmers (sometimes referred to

Novice Programming Environments, NPE) have been in use since 1989 and experimentally demonstrated to have positive effects on the learners. However, creating expert-crafted models for programming ITS comes at cost ratios of up to 100:1 (e.g. 100 work hours per one hour of content). [13] Additional complexity is added by infinite solutions spaces which are possible in open ended programming tasks. Therefore, instead of using expert crafted models collaborative filtering is incorporated into these systems to automatically create hints from existing solutions. [10][13] The principle of using existing solutions of tasks to create hints and guidance was first introduced by Barnes and Stamper in 2008 with their system Hint Factory [2].

## 4.1 Technological Overview

Recent implementations of collaborative filtering based hint generation systems still show similarities to the original Hint Factory design. Therefore two recent and tested implementations are introduced instead of a general introduction. Firstly ITAP, an intelligent tutoring system for Python is introduced. Secondly the iSnap extension for the Snap! graphical programming language is discussed. These implementations were selected as both have been extensively evaluated [11].

**ITAP** is an algorithm for generating hints for programs in the Python programming language. ITAP uses the concept of a solution space graph, which models intermediate states of a student's solution as nodes on a graph, where the state is represented by the code itself (in the form of an abstract syntax tree). Edits are modeled as edges between state-nodes. [13] This was first introduced by the Hint Factory algorithm [2]. The design extends the Hint Factory approach by automatically adding new states into the solution space based on existing ones. New states are generated by comparing existing ones and recreating a set of edits that leads from one to the other. ITAP therefore can cover novel states which are not in the historic data. Before it can generate hints it needs at least one sample solution to build up the graph. ITAP furthermore requires a test function to score the correctness of solutions. [13]

**iSnap** is an extension to the Snap! programming environment which itself is based on Scratch. Snap! provides a graphical programming interface and is specially designed for novel programmers. In a first step iSnap extends Snap! by introducing extensive logging of all student actions, including user interface and coding area interactions as well as complete snapshots of the code. It then provides contextual hints based on previous solutions. [10]

Similarly to ITAP, iSnap also makes use of abstract syntax trees (AST) to represent the code states. The algorithm builds contextual interaction networks which models how students edit a subsection of the program as a root path within the AST. By comparing valid root paths in the database to the current one, iSnap can create hints. This increases matching probability as edits rather than whole code states are matched. Therefore the scope is only a small subpart instead of the whole program. The hints itself are generated by the Hint Factory algorithm. [9] [10] Compared to ITAP, iSnap does not generate novel states itself but relies on a more fine edit model to increase the effectiveness of its predictions. [10]

### 4.2 Advantages

Both algorithms have demonstrated in evaluations that they achieve their goal of generating meaningful hints that can be used by students to improve their code [11]. The provided feedback guides student to better solutions and increase the effectiveness of the teaching platform. It was also shown in studies that instant and on-demand feedback increases the benefit of teaching systems. [4] Furthermore it was shown that ITAP and iSnap on average generate better hints than a single human tutor in the same scenarios. This indicates that hints are not generic but rather contextually meaningful. Therefore both algorithms provide cost-ratio decreasing value to educational programming settings. [11]

### 4.3 Limitations

Performance evaluations of ITAP and iSnap conducted by Price et. al. suggest that whilst hint generation can outperform human tutors, the amount of solutions in the solution space have an influence on the quality of the generated hints. Hint quality stopped improving after 15-20 data sets. Furthermore there was also a measurable decrease in the hint quality with more data in the system, for reasons not yet known. Additionally multiple human tutors working together perform better than any of the algorithms. [11] Further limitation arises from the cold start problem which is one of the fundamental problems of the collaborative filtering approach. It describes that a lack of data in the dataset leads to insufficient, bad suggestions or to a total lack of the ability to create suggestions. Both systems therefore need a certain base set of data to produce hints. This means that there needs to be at least a sample solution for each task to serve as a basis for hint generation. [16]

ITAP is further limited to syntactically correct and parseable programs as the program needs to be parsed in order to check its correctness. In situations with syntax errors it therefore ca not perform at all. [13]

## 5 Results

The classification shows that the "task-based guidance" and "multi-step-solution task" group, which also includes the introduced programming ITS, is most likely to benefit from the concepts of iSnap and ITAP, as these kinds of tasks require a sequence of steps that must be completed in order to solve a specific problem.

### 5.1 Proposing Collaborative Filtering as a Foundation of a Domain-Agnostic Hint Generation

Linearized, multi-step (LMS) tasks share similarities with simple programs, where each step in the solution can be compared to a code instruction. Therefore the whole program can be encoded in the form of a linear graph or tree. Multiple solutions, and multiple snapshots of one solution can therefore be mapped into a graph similarly to programs which are mapped into solution space graphs in ITAP and iSnap. This suggests that similar, hint factory based approaches can be applied to LMS tasks, introducing the ability of on-demand hint generation to other educational domains. Potentially, such algorithms have similar cost-ratio and teaching-quality impacts as those of iSnap and ITAP, as the technology needs little adaption as the viewed solution space and solution representation is resemblant of that of iSnap and ITAP.

### 5.2 Challenges

It can be assumed that requirements and challenges of the proposed applications are similar to those of iSnap, ITAP and other collaborative filtering based approaches to novice programming environments. Sample solutions would be required to avoid the cold-start problem. Testing routines for students' solutions would also be needed to grade the solutions and to filter out wrong solutions from the solution space. Furthermore a formal and interpretable description language would be needed to formalize the student solutions. Whilst these may already exist for certain tasks and systems, the creation of a domain-agnostic language would be needed to create truly domain-agnostic tutoring systems.

## Conclusion

Current research into programming ITS based on the collaborative filtering approach show that by using crowd-sourced solutions personalized guidance and on-demand hints can be automatically generated. This measurably increases the learning experience and significantly reduces the costs of creating interactive, hint generating programming environments. The discussed algorithms focus on graph and tree based representations of task solutions and solution spaces. Applying these solutions to other domains therefore infers the need of multi-step tasks whose solutions can be represented by trees or graphs. Furthermore the ability to collect solutions of other students is a hard requirement. Especially ITS for formalized settings like science-education in which tasks often require multiple steps to solve, e. g. physics problems or multi-step transformations in mathematics education could benefit from the existing research. Furthermore, logic and modeling tasks like the creation of software engineering models could also be candidates for ITS with hint generation based on collaborative filtering. However, the set of potential applications is not exclusive to science education but rather all ITS that fit the above mentioned criteria can make use of these principles. Further work in this field is required to create implementations specific to other problem domains or even domain agnostic models. Additionally studies that show the performance and gains of these methods are required.

## Acknowledgements

The author would like to thank the i9 - Learning Technologies Research Group for their guidance and support and specifically the advisors and organizers of this proseminar, namely Christoph Greven and Sven Judel.

## References

1. Antonija Mitrovic, e.a.: Thermo-tutor: An intelligent tutoring system for thermodynamics. In: 2011 IEEE Global Engineering Education Conference (EDUCON). pp. 378–385. IEEE (2011)
2. Barnes, T., Stamper, J.: Toward automatic hint generation for logic proof tutoring using historical student data. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) Intelligent Tutoring Systems. pp. 373–382. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
3. Brzoza, P., Lobos, E., Macura, J., Sikora, B., Zabka, M.: Formath - intelligent tutoring system in mathematics. In: CSEDU (2012)
4. Corbett, A.T., Anderson, J.R.: Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 245–252. CHI '01, ACM, New York, NY, USA (2001). https://doi.org/10.1145/365024.365111, http://doi.acm.org/10.1145/365024.365111
5. Hagemann, N., O'Mahony, M.P., Smyth, B.: Module advisor: Guiding students with recommendations. In: Intelligent Tutoring Systems. pp. 319–325. Springer International Publishing, Cham (2018)
6. Min, W., Rowe, J.P., Mott, B.W., Lester, J.C.: Personalizing embedded assessment sequences in narrative-centered learning environments: A collaborative filtering approach. In: Artificial Intelligence in Education. pp. 369–378. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
7. Murray, T.: Authoring Intelligent Tutoring Systems: An analysis of the state of the art. International Journal of Artificial Intelligence in Education (IJAIED) **10**, 98–129 (1999), https://telearn.archives-ouvertes.fr/hal-00197339, part II of the Special Issue on Authoring Systems for Intelligent Tutoring Systems (editors: Tom Murray and Stephen Blessing)
8. Nkambou, R., Mizoguchi, R., Bourdeau, J.: Advances in Intelligent Tutoring Systems. Springer Publishing Company, Incorporated, 1st edn. (2010)
9. Price, T.W., Dong, Y., Barnes, T.: Generating data-driven hints for open-ended programming. In: Proceedings of the 9th International Conference on Educational Data Mining. pp. 191–198. International Educational Data Mining Society (2016)
10. Price, T.W., Dong, Y., Lipovac, D.: isnap: Towards intelligent tutoring in novice programming environments. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. pp. 483–488. SIGCSE '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3017680.3017762, http://doi.acm.org/10.1145/3017680.3017762

11. Price, T.W., Zhi, R., Dong, Y., Lytle, N., Barnes, T.: The impact of data quantity and source on the quality of data-driven hints forprogramming. In: Penstein Rosé, C., Martínez-Maldonado, R., Hoppe, H.U., Luckin, R., Mavrikis, M., Porayska-Pomsta, K., McLaren, B., du Boulay, B. (eds.) Artificial Intelligence in Education. pp. 476–490. Springer International Publishing, Cham (2018)
12. Reddy, S., Labutov, I., Joachims, T.: Learning student and content embeddings for personalized lesson sequence recommendation. In: Proceedings of the Third (2016) ACM Conference on Learning @ Scale. pp. 93–96. L@S '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2876034.2893375, http://doi.acm.org/10.1145/2876034.2893375
13. Rivers, K., Koedinger, K.R.: Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. International Journal of Artificial Intelligence in Education **27**(1), 37–64 (Mar 2017). https://doi.org/10.1007/s40593-015-0070-z, https://doi.org/10.1007/s40593-015-0070-z
14. Sarwat, M., Mokbel, M.F.: Encyclopedia of Database Systems - Collaborative Filtering, pp. 1–5. Springer New York, New York, NY (2017)
15. Sarwat, M., Mokbel, M.F.: Encyclopedia of Database Systems - Recommender Systems, pp. 1–5. Springer New York, New York, NY (2017)
16. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. in Artif. Intell. **2009**, 4:2–4:2 (Jan 2009). https://doi.org/10.1155/2009/421425, http://dx.doi.org/10.1155/2009/421425