Daniel Schiffner (Hrsg.): Proceedings of DeLFI Workshops 2018
co-located with 16th e-Learning Conference of the German Computer Society (DeLFI 2018)
Frankfurt, Germany, September 10, 2018

Branched Learning Paths for the Recommendation of Personalized Sequences of Course Items

Christopher Krauss[1], Andreas Salzmann[2] and Agathe Merceron[3]

Abstract: Current research in Learning Analytics is also concerned with creating personalized learning paths for students. Therefore, Recommender Systems are used to suggest the next object to learn or pre-computed paths are recommended. However, the time of the requested learning session and the freedom of choice are often not considered. Respecting certain course deadlines and providing the user with choice is a very important aspect of recommendations. In this work, we present an approach to creating personalized paths through knowledge networks. These paths are constructed by considering the time at which they are requested and suggest alternative routes to provide the user with a choice of preferred learning items. An evaluation gives precision measures that have been obtained with different lengths for the Top-N recommendations and different branching factors in the paths and compares the results with other Recommender Systems used in Adaptive Learning Environments.

Keywords: Learning Analytics, personalized learning paths, multi-modal routes

1 Introduction

Various educational stakeholders have identified adaptive learning as one of the most promising and at the same time most challenging trends of Learning Analytics [VMO12] [EFR15]. Thereby, various Recommender System (RS) techniques are used to provide a personalized learning experience to the user. This paper focuses on RS that recommends content in a specific course. The user gets the choice which subject to learn or skip. However, most of the research is focused on the recommendation of single learning items.

When users want to obtain a complete overview of all offered learning items or to see the recommended order of the next learning items, traditional Collaborative Filtering algorithms need to be extended. The identification of item sequences is typically handled by a special group of Recommender Systems, as it requires knowledge about the relations between items instead of just predicting a relevance score per item. Shen and Shen [SS04] introduced a prediction model with a sequencing rule algorithm by taking a topic ontology into account. When the system identifies that a user lacks knowledge, appropriate contents are recommended. In this work, a novel Learning Path algorithm builds on this foundation.

[1] Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, christopher.krauss@fokus.fraunhofer.de
[2] Mablo, Kurfürstenstraße 141, 10785 Berlin, andreas@mablo.net
[3] Beuth Hochschule für Technik, Luxemburger Straße 10, 13353 Berlin, merceron@beuth-hochschule.de

This algorithm recommends paths of learning items dependent of the current state of the learner.

The remainder is structured as follows: Chapter 2 introduces related work and, especially, different learning path strategies. Chapter 3 gives an overview of the developed algorithm and Chapter 4 presents a deep dive. Chapter 5 discusses the evaluation results and, finally, this paper ends with a conclusion and an outlook.

2 Related Work

A common technique for the presentation of item relations is to create a database with directed graphs, including nodes for items (which are, in our case, the learning items to be recommended) and edges for their relations [ABB03] [VS06] [YW09]. Thereby, different approaches are frequently utilized to create the needed item sequences. Six key approaches are described in the following:

1. Teacher's Sequence: Teachers and educational staff have the best knowledge of the taught topics and the intended knowledge transfer. Thus, they model item sequences manually and in a pedagogical way [DHK09].
2. Content-based Analysis allows for the generation of a topical structure without the supervision of humans. It rather analyzes the content, especially textual input, to automatically generate dependencies.
3. Constraint-based Approaches are similar to manually defined structures in allowing for more choice in the personal learning directions with some predefined restrictions [ABB03] [VS06]. For instance, the learners might appreciate the opportunity to freely choose the next learning item as long as all prerequisites are fulfilled.
4. The Knowledge-based Approach is based on the previously acquired knowledge of the learner, e.g., by presenting a survey at the beginning of a course or by letting the user answer some related questions when accessing the content. According to the determined knowledge, well-known items are filtered out, and items with new topics gain a higher relevance [YW09].
5. Activity-based Analysis: The set of user activities can be utilized to generate item dependencies based on the past consumption activities of learners [DHK09]. This approach requires a sufficient number of users and interactions.
6. Hybrid Combination: Similar to Hybrid Filtering, the hybrid combination of approaches represents the most powerful class, as it overcomes the weaknesses of single approaches.

Most publications allocate an important role to user feedback for learning path generations in terms of the learners' interests and knowledge levels. Drachsler et al. [DHK09] note that explicit user feedback, such as ratings or tags, helps to identify paths in learning networks

in a more efficient way than other input types. Voss et al. developed an adaptive sequencer that uses Matrix Factorization as the performance predictor [VSM15]. With the same approach, Schatten and Schmidt-Thieme tried to "keep the contents in the Vygotskis Zone of Proximal Development (ZPD) [VYG80], i.e., the area where the contents neither bore or overwhelm the learner" [SS14]. In each calculation step, the system selects contents with a predicted performance score that is most similar to the user's modeled score. Researchers of the Worcester Polytechnic Institute [XWB15] enhance long-term retention of acquired knowledge by creating a Personalized Adaptive Scheduling System for retention tests. Another example of learning paths was realized by Nabizadeh et al. [NMP17]: They restricted the item sequence to those learning items that allow "obtaining the maximum possible learning score in a limited time" [NMP17]. While path creation algorithms are frequently utilized for the prediction of learning sequences (cf. [ABB03] [VS06] [YW09] [NMP17]), none of them consider the presentation of alternative routes with path branches. However, the idea of a Recommender System is to offer different choices in a particular situation. This work focuses on an approach that incorporates paths with branch alternatives which are then presented to the user.

3 Approach and Context of Evaluation

Our approach to predicting these personalized learning paths consists of two separate steps: The first step handles the definition of the knowledge graph that represents possible routes through the items. A hybrid combination of teacher-based, constraint-based and user-interaction-based approaches are utilized for building the knowledge graph. The direct transitions (edges) from one item (node) to another comprises a probability that describes the percentage of past transitions. Thereby, each historical path of learners is analyzed to create a knowledge graph and the probabilities. In other words: the more often users studied items in a particular order, the higher is the probability that this part of the path will be recommended to other learners in the future.

The second step predicts an individual learning path for each learner in the given situation. Thereby, the past item accesses are considered for the requesting user to predict a personalized subpath. This future path starts at the last consumed item node and avoids other, previously seen, items. The idea is that the algorithm searches for the most efficient path through all left learning items in the before built knowledge graph.

The result is a set of individual paths that lead through all offered items exactly once. Repeating or skipping items is not possible when consequently following the recommended route. However, an extension of step 2 might neglect items that are marked as "known" by the learner - which is not implemented for this evaluation. This approach has been evaluated with the Advanced Web Technologies (AWT) course, a course for master students of computer science at the Technische Universität Berlin. Five technical

experts taught in 12 presence lectures nine topics that are of interest for future web developers – from web technology basics, such as HTML, over media delivery and protection, to personalization through Recommender Systems (cf. [KMA17] [KRA18]). A group of 145 students enrolled in the course and 99 of them used a novel web application to access the course materials. Their interactions with the web app are collected for the evaluation of the learning path approach. We presented almost 1,000 Learning Objects grouped in about 100 Learning Units (LU) of the 9 main chapters. Besides a few videos and some interactive exercises, Learning Objects are mainly sets of slides that can be downloaded [KMA17]. Given the number of about 100 Learning Units within the course, there are about 10^{157} possible combinations of item sequences ($100! \sim 9*10^{157}$) - not considering the about 1,000 Learning Objects that make up the 100 Learning Units. The learner, in turn, only needs to select one appropriate series, that is called Learning Path, to consume all given items.

4 Algorithm Design

The algorithm is inspired by routing plan algorithms for public transportation where trains follow particular schedules [ZA08]. The actual route of a passenger, however, is similar to the learner's individual learning path in that it needs to follow some constraints (e.g., the lecture schedule) but allows for decisions at multiple points in time in respect of alternative routes.

4.1 Dependency Graph

The actual algorithm splits step 1 into two parts. At first, the AWT Learning Units are transferred into a graph database as nodes. Let I be a set of items belonging to the course C. Prerequisites, given as attributes in the meta-data according to IMS LOM[4], are considered for a primary dependency graph. Prerequisites for the AWT Learning Units are manually defined and can be seen as constraints of a path that must not be violated - for instance the Learning Unit "Concepts of Recommender Systems" is a prerequisite for the Learning Unit "Technical Issues in Recommender Systems". The result is a dependency graph representing all allowed learning transitions. See Figure 1 for an example of the dependency graph with six items (LO1, ..., LO6) where LO2 is a prerequisite for LO3 (LO3 → LO2), LO3 is a prerequisite for LO4 (LO4 → LO3) and LO5 a prerequisite for LO6 (LO6 → LO5).

[4] IMS Global Learning Consortium: LOM - IEEE Learning Resource Meta-Data Specification, 2006, https://www.imsglobal.org/metadata/index.html

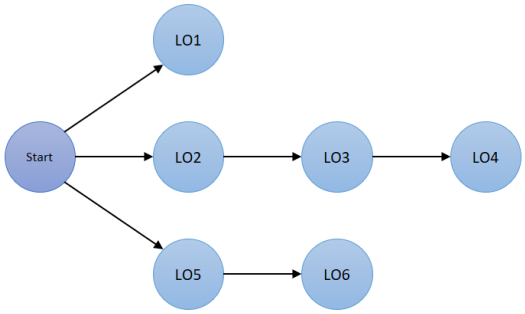

Figure 1: Example of a dependency graph with six Learning Objects and three defined prerequisites (LO4 → LO3, LO3 → LO2 and LO6 → LO5).

4.2 Knowledge Graph

Secondly, based on the defined prerequisite constraints, all allowed combinations of items in I are listed. The subset s of items in I represents one possible learning state with all so-far consumed items. S is the set of all possible states within the course:

$$S = \{s | s \subseteq C\}.$$

Each transition from one state to another is represented by a directed edge $e_{s1,s2}$ in E except where the transition violates the prerequisite definition. An edge connects two states s1 and s2 where one item is added to s1 which results in state s2. E is the set of all edges.

The directed transition edges are extended by an attribute that indicates the number of other learners who used the same transition in the past. More precisely, it represents a ratio of historic transitions $e_{transitions}$ between exactly the two states (s1 and s2) and the number of all historic transitions $s_{transitions}$ leaving s1. The result of the probability function p(e) determines the transition probability per edge e of historic movements of all learners and is given in percent:

$$p(e) = \frac{e_{transitions}}{s_{transitions}}.$$

As an intermediate result, a knowledge graph K is generated that represents all possible states and transitions of the dependency graph between a start state B, which does not contain any item, and a target state T = C, which contains all items of the course.

$$K = (B, T, S, E, p).$$

Figure 2 visualizes the example knowledge graph with the six items of the dependency graph example. This graph is computed offline and stored in a database to efficiently calculate individual paths in the future.

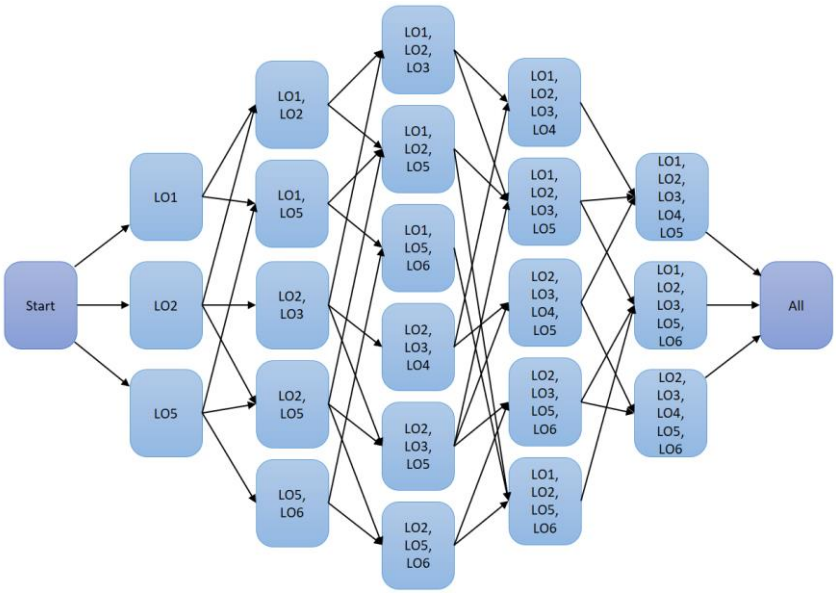

Figure 2: Example of a knowledge graph with the same six example Learning Objects of the dependency graph example.

4.3 Path Generation

When now a learner requests a path recommendation, all of the so far accessed items of that user are considered. The learner's accessed items correspond to a state in the knowledge graph K reflecting a new starting point $B_{u,t}$ for user u at time t. When the user, for instance, accessed LO3, LO2 and LO4, the state {LO2, LO3, LO4} is the starting point for the routing algorithm. If the user violated the prerequisite definition and studied another item first, the state that shows the highest agreement of known items is considered as the start state.

The routing algorithm identifies an ideal route r from the start to the target point T in the knowledge graph (marked with "All" in Figure 2). The target point T reflects all studied items in the course and thus the course goal. R is the set of all possible routes between B and T. Thereby, all probabilities of the taken edges on the route are summed (a multiplication would penalize smaller edge probabilities and was not evaluated in this research). E_r is the number of edges of the route r. The path with the highest total probability $p_{total}(r)$ should be preferred.

$$p_{total}(r) = \sum_{e=0}^{E_r} p(e).$$

The routing plan algorithm of Zografos and Androutsopoulos [ZA08] was used in the evaluation. It analyzes backwards the transitions from T to $B_{u,t}$ by considering the probability p(e) per edge. Moreover, as the algorithm is designed for public transportation, it allows incorporating waiting times, when connecting. Transferred to the learning domain, this feature is used to penalize switches of the higher-level topics as the learner might better focus when working on similar topics at a time and encourage switches first when all low-level items belonging to one high-level item are processed. The result is a list of alternative routes with a number of branches b per node. Only those b branches are considered that show the highest transition probability p(e) from one edge to another. Instructors might adjust the total number of considered and presented branches. The implementation is realized as a Java server with the graph database neo4j[5].

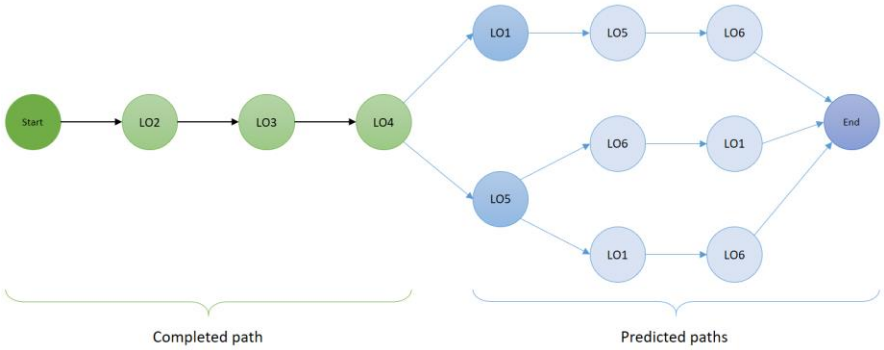

Figure 3: Example of a path presented to the learner when LO2, LO3 and LO4 have been consumed.

5 Evaluation

The 16 weeks of the AWT course data is used as data set. The course ran from October 24, 2016, until February 12, 2017. The test data comprises all 99 active users and 8,241 activity statements on 106 Learning Units. The data is split 15 times into training and test data set by following the "increasing time-window cross validation" of Campos et al. [CDC14], where the time threshold shifts by seven days. Thereby, with each split, the duration of the training dataset increases by seven days, and the test dataset decreases by the same amount. Items are considered as relevant that have been processed by the same user after the point of time of the recommendation (see Krauss [KRA18] for details on this evaluation setting). Besides the split per week, which lead to 15 different time-dependent results, different sizes of branches per node are considered, which are given as b (b=1, 2, 3, 5, 10 and 15). Moreover, the Top-N lists contain in different settings 3, 5, 10 or 15 items. The evaluation was performed in 360 iterations (15 splits * 6 sizes of b * 4 lists).

[5] neo4j. See https://neo4j.com/ (Accessed: 22.08.2017).

Since the number of branches b and recommendations N are not necessarily equal, the Top-N list consists of the items with the lowest distance (number of edges) to the last accessed item (state $B_{u,t}$). When deciding for items with the same number of edges to the last accessed item, those with the highest transition probability are preferred. This is determined iteratively, starting with direct connected items. Given the example of b=3 and N=4, the 3 direct connected items are set on the Top-N list in the first iteration. Afterward, all 9 items that are connected via 2 edges to the start item are considered as the missing Top-N items. Thereby, the algorithm prefers the items with the highest transition probability. This is repeated until the Top-N list consists of N items where every item of the Top-N list must be unique. For the defined evaluation setting and in order to produce comparable results with the other approaches, the connections between the items (the actual learning path) is not of interest for the measurements (just for the selection process). For the evaluation, only the fact is evaluated that an item is part of the Top-N list.

The precision value represents the ratio of relevant items that have been recommended to all recommended items. Relevant items have been processed by the user after the point of time of the recommendation. Due to the vast amount of experiments, Figure 4 shows the precision of the main settings averaged over the whole course period. This is done to better compare the different settings. In general, the fewer the number of elements in the Top-N list, the more precise are the recommendations (Top-3 performs best when b>2). Over half of the recommended items are relevant (precision of up to 0.58). The number of branches does not have such a massive impact on the precision measure as the Top-N elements have. Nevertheless, it seems clear that b=10 is the best setting - and b=1 and b=2 perform almost identically low.

We compared this approach to other approaches, known from the literature, in the same evaluation setting and utilizing the same data set (each in the most precise setting): The Slope One algorithm by Lemire et al. [LM05] reached a precision of only 0.459. An extension of this approach with time-weights from Jiang and Lu [JL13] improves this precision to 0.465. And the approach of Hermann [HER10] leads to a precision of 0.512

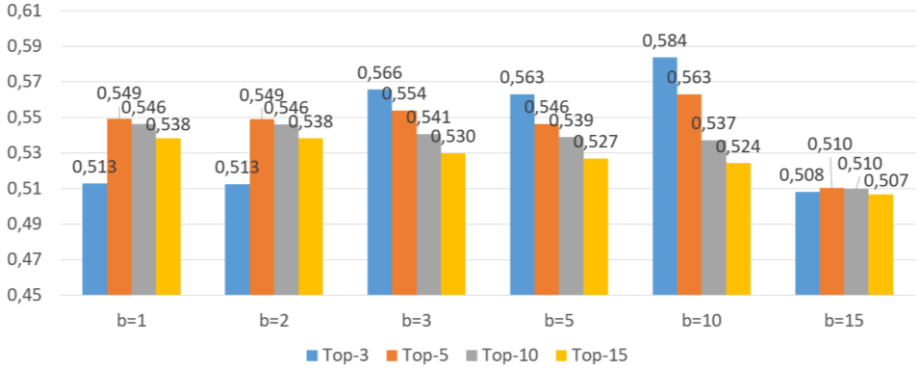

Figure 4: Precision for different settings of the Top-N path recommendations.

under the same conditions. It is important to notice that the evaluation procedure of Krauss [KRA18] is designed to better tell about timely effects in real courses and, thus, the results are not comparable to results of common evaluations settings and datasets of other domains which often exceeds precision values of 0.8.

6 Conclusions

The precision of the introduced Learning Paths based on the activities of other learners is 7 to 13% better than the precision of the other evaluated Collaborative Filtering algorithms. However, the most significant strength comes from the cold start phase, where the predicted Learning Paths rely on item metadata that have been entered by educational staff. Thus, even at the beginning of the course, the recommendations are highly relevant. The drawback comes from its static concept where every item is presented exactly once.

The reasons why all evaluated approaches reach low precision values of under 0.6 lies in the setup of the evaluation framework and in the composition of the activity data. The algorithms are not trained with a random selection of activity data as known from the n-fold cross-validation, but in a strict chronological order which higher weights temporal effects, such as the cold start, little activity during the course and a massive learning phase at the end. In the evaluated dataset, 65% of the students only infrequently learned with the system [KMA17] – which is typical for university courses. However, this circumstance also reduces the average precision. Moreover, items for the course introduction are often skipped by learners – but frequently recommended by the algorithms. An extension of the algorithm would, for this reason, also consider item skipping and repetitions.

Acknowledgments

The authors would like to thank the instructors of the AWT course and the whole Smart Learning Team. This work is supported by the German Federal Ministry of Education and Research grant number 01PD14002D and 01PD17002D.

References

[ABB03] Atif, Y., Benlamri, R., & Berri, J. (2003). Learning objects based framework for self-adaptive learning. Education and Information Technologies, 8(4), 345-368.

[CDC14] Campos, P. G., Díez, F., & Cantador, I. (2014). Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Modeling and User-Adapted Interaction, 24(1-2), 67-119.

[DHK09] Drachsler, H., Hummel, H. G., & Koper, R. (2009). Identifying the goal, user model and conditions of recommender systems for formal and informal learning. Journal of Digital Information, 10(2).

[EFR15] Erdt, M., Fernandez, A., & Rensing, C. (2015). Evaluating recommender systems for technology enhanced learning: a quantitative survey. IEEE Transactions on Learning Technologies, 8(4), 326-344.

[HER10] Hermann, C. (2010, June). Time-based recommendations for lecture materials. In EdMedia: AACE World Conference on Educational Media and Technology.

[JL13] Jiang, T. Q., & Lu, W. (2013). Improved slope one algorithm based on time weight. In Applied Mechanics and Materials (Vol. 347, pp. 2365-2368). Trans Tech Publications.

[KMA17] Krauss, C., Merceron, A., An, T. S., Zwicklbauer, M., Steglich, S., & Arbanowski, S. (2017, October). Teaching Advanced Web Technologies with a Mobile Learning Companion Application. In Proceedings of the 16th ACM mLearn World Conference.

[KRA18] Krauss, C. (2018, June). Time-Dependent Recommender Systems for the Prediction of Appropriate Learning Objects. Dissertation, Technische Universität Berlin, Deposit Once (http://dx.doi.org/10.14279/depositonce-7119), 2018.

[LM05] Lemire, D., & Maclachlan, A. (2005, April). Slope one predictors for online rating-based collaborative filtering. In Proceedings of the 2005 SIAM International Conference on Data Mining (pp. 471-475). Society for Industrial and Applied Mathematics.

[NMP17] Nabizadeh, A. H., Mário Jorge, A., & Paulo Leal, J. (2017, July). RUTICO: Recommending Successful Learning Paths Under Time Constraints. In Adjunct Publication of the 25th UMAP Conference (pp. 153-158). ACM.

[SS04] Shen, L. P., & Shen, R. M. (2004, August). Learning content recommendation service based-on simple sequencing specification. In International Conference on Web-Based Learning (pp. 363-370). Springer, Berlin, Heidelberg.

[SS14] Schatten, C., & Schmidt-Thieme, L. (2014, April). Adaptive Content Sequencing without Domain Information. In CSEDU (1) (pp. 25-33).

[VMO12] Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I., & Duval, E. (2012). Context-aware recommender systems for learning: a survey and future challenges. IEEE Transactions on Learning Technologies, 5(4), 318-335.

[VS06] Viet, A. N., & Si, D. H. (2006, September). ACGs: Adaptive Course Generation System-An efficient approach to build E-learning course. In Computer and Information Technology, 2006. CIT'06 (pp. 259-259). IEEE.

[VSM15] Voß, L., Schatten, C., Mazziotti, C., & Schmidt-Thieme, L. (2015). A Transfer Learning Approach for Applying Matrix Factorization to Small ITS Datasets. International Educational Data Mining Society.

[VYG80] Vygotsky, L. S. (1980). Mind in society: The development of higher psychological processes. Harvard university press.

[XWB15] Xiong, X., Wang, Y., & Beck, J. B. (2015, March). Improving students' long-term retention performance: a study on personalized retention schedules. In Proceedings of the Fifth International LAK Conference (pp. 325-329). ACM.

[YW09] Yang, Y. J., & Wu, C. (2009). An attribute-based ant colony system for adaptive learning object recommendation. Expert Systems with Applications, 36(2), 3034-3047.

[ZA08] Zografos, K. G., & Androutsopoulos, K. N. (2008). Algorithms for itinerary planning in multimodal transportation networks. IEEE Transactions on Intelligent Transportation Systems, 9(1), 175-184.