A Sequence Based Recommender System for Learning Resources

Dean Cummins *Kalina Yacef* *Irena Koprinska*

School of Information Technologies
University of Sydney
NSW 2042 Australia

(dcummins, kalina, irena) @it.usyd.edu.au

Abstract *This paper presents a novel approach for recommending sequences of resources for users to view based on previous user feedback. It considers the order in which resources are viewed to be important in delivering the next set of suggestions and tries to learn these dependencies from users' ratings. Although we describe our approach in the context of e-learning, it can be applied to other domains where ordering is important. We also propose a novel algorithm for learning the dependencies between the resources. Preliminary results are encouraging: they show that, after a threshold in quantity of feedback, our algorithm provides better results than standard collaborative filtering.*

Keywords Digital Libraries, Document Management, Information Retrieval

1 Introduction

Online learning resources can be very convenient to help users learn or practice a topic. Like any other resources that are part of a very large set which can potentially grow infinitely, such as movies or books, users appreciate some guidance in selecting the appropriate resource. There are two major ways to provide this guidance: one is based on *annotation* and the other on *collaborative filtering*. The first one typically relies on resource content metadata and requires the resource structure to follow some standards. Thus, a significant human effort is required to annotate and structure the resources. The second way relies on previous users' ratings to make recommendations, and this is where our research falls in. In particular, in this paper we study whether users' ratings could be used to achieve reusability of learning resources.

There are already a number of techniques and algorithms to build collaborative recommender systems (two famous systems are MovieLens [16] and Amazon [2]). However, there are two interesting aspects that distinguish learning resources from other items:

(i) the order in which the learning resources are seen is important: there is no point in seeing the difficult material without seeing the basics first;

Submission for the Eleventh Australasian Document Computing Symposium (ADCS 2006). Copyright for this article remains with the authors.

(ii) there is an end to the process of seeing learning resources. Learners typically see a small, finite number of resources for a given topic, whereas there is no end to the process of seeing movies or reading books.

These two aspects make the recommendation problem interesting and novel. We propose a sequence-based recommender approach to make suggestions to learners on what resources to use in order to learn about a given topic. Importantly, our approach aims at suggesting one or more learning paths. Learners seek new resources on the open web and add them to the pool of resources, which then grows over time. As the resources are procured from the open web and are referenced simply by a URL, the goal of reusability is achieved.

The next section discusses related research on resource reusability in online learning. Section 3 then formally defines the problem and presents our approach, highlighting the issues and challenges that lay ahead. Section 4 introduces the dependency learner algorithm, on which our approach relies on. Section 5 presents the experiments conducted to evaluate the algorithm and discusses the results. Finally Section 6 concludes the paper.

2 Background and existing work

In order to promote the reusability and identification of useful educational content on the web, the term *learning object* was coined [18]. The broad idea is centered on classifying suitable resources as learning objects and having them available in purpose built systems for teachers and learners, such as a Learning Object Repository [17]. These repositories aim to be a searchable catalog, enabling the sharing and reusibility of learning objects.

There are many definitions of learning objects and they are not all consistent [18]. The IEEE Learning Technology Standards Committee (LTSC) [10] defines learning objects as *any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning*. Others define it with a much smaller scale, such as a *digital learning resource that facilitates a single learning objective which may be reused in a different context* [15]. Regardless, an analysis of online repositories [7, 13] clearly shows learning objects to fit the first definition, or more specifically *any digital resource that can be reused to support learning* [21].

While all potentially useful resources available on the web fit these general definitions of learning objects, they are not in these learning repositories. It takes large amounts of human effort to identify, annotate and place these resources into these repositories. To perform this for all resources on the web is a gigantic and infeasible task for a small group of humans to do manually. This means that many potentially useful resources are left untouched if instructors and learners rely on these repositories alone. A learning system which does not require resources to exist in purpose-built repositories and does not require annotation would clearly be more generic and have a larger number of resources to choose from.

2.1 Annotation/metadata approaches

For learning objects to be shared and reused, current approaches involve adding some structure or metadata to the learning objects so that both humans and machines can understand them. This is similar to the semantic web vision in which information is annotated with well defined meaning so that computers and humans can better work in cooperation [3]. Mohan [15] argues that learning objects themselves should play a greater role in the search process and should intelligently interact with learning systems to provide instruction. This would allow intelligent search agents to identify pools of learning objects that are suitable for specific instructional goals. For this to be realised much work has focused on creating and implementing metadata standards and supporting ontologies.

Current popular metadata standards include Dublin Core [5] and IEEE's Learning Object Metadata (LOM) [11]. These are supported by a large number of metadata creation software, repositories and learning systems. The reality is that metadata creation is a lengthy, boring and tedious process with an undesirable high human cost involved [4]. Consequently elements are often overlooked or used incorrectly and inconsistently [8]. It was further identified that to promote semantic interoperability, common vocabularies or ontologies are required. However this is currently not the case [15, 19, 4].

Without consistently and extensively annotated learning objects, current learning systems cannot interoperate with each other and make effective use of the vast amount of available resources. The goal of achieving common standards and interoperability of all available learning objects therefore does not appear to be realistically achievable in the near future.

2.2 Collaborative filtering approaches

Another approach commonly used for sharing and reusing documents is to use social or collaborative filtering. For instance, recommendation systems aim to provide the user with choices or recommendations based on how users rated the resources, hence can even provide personalised recommendations. Current popular applications of these systems include MovieLens [16] and Amazon [2]. More recently we have seen collaborative filtering applied to electronic documents such as document searches [12] and research papers recommendations [14, 20]. However these approaches are focused on recommending single, unrelated resources whereas we are interested in recommending resources which may have implicit orderings within them.

For example, if we implemented a recommender system for resources using traditional techniques, resources would be suggested based on previous ratings by all learners and learners similar to the current learner, without taking into account sequencing between resources. More specifically, consider three resources A, B and C. A and B are two resources which deal with basic concepts for some topic, and C covers more advanced material. A traditional recommendation list will potentially include all three items, (A, B, C), ranked by the popularity of previous ratings given by learners. A learner may pick any of the three resources with the impression that top ranked resources are more useful and liked by other learners.

However, we wish to take into account that there may be implicit dependencies between resources. As C deals with more advanced material, it would be appropriate to recommend resources A and B to the learner before recommending C, regardless of whether C has been rated higher overall. Once the learner has seen A or B and found them useful then C should be recommended, thus presenting the resources in an order that takes sequencing into account.

We can also extend this recommendation by making available to the learner full paths of resources such as $(A \rightarrow C, B \rightarrow C)$ ahead of time. This provides the learner with a more structured environment that is similar to traditional custom learning systems but using a recommendation based approach. Learners are then able to visualise suggested learning paths, not just lists of resources.

3 The approach

In contrast with using metadata or standard collaborative filtering, our approach learns and extracts the dependencies that exist in the learning resources based on user ratings. Our approach does not rely on human annotation of learning resources and aims at discovering not only popular resources but also popular paths. Once we have mined these potential dependencies, we combine them into a dependency graph that can then be used by an intelligent recommender system to suggest resources as shown in Figure 1.

In our proposed system, the learner is suggested a set of learning paths (i.e. sequences of learning resources as identified in the dependency graph), has

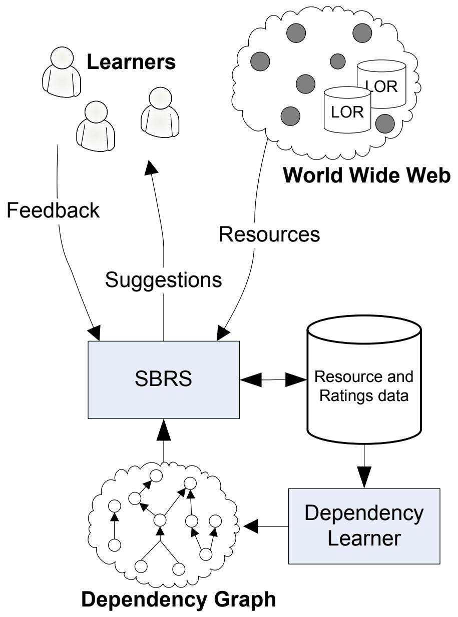

Figure 1: Proposed Sequence Based Recommender System

the possibility to view the other resources as well as to pick new resources from the open web. After viewing each resource, the learner is asked to provide a rating (feedback), which is added to the resource rating database. The system has a dependency learner, described in Section 4, which builds and maintains the dependency graph of resources.

3.1 Proposed Sequence Based Recommender System

We describe the main sections of our proposed sequence-based recommender system.

Growing the pool of resources. To take advantage of the vast amount of resources on the world wide web, we propose that learners and instructions be given the ability to add resources into the system that they believe will be useful for the given topic, similar to social bookmarking sites such as Del.ico.ous [6]. This will likely grow a pool of relevant resources with a high degree of quality as it has already been hand picked by a human. Naturally, spam and irrelevant resources may be added to this pool, however if these are not selected and are consistently rated poorly by learners they can be filtered out. We believe this to be a great strength of this approach and will allow the system to evolve over time as the number of resources increases. The system can also link into existing learning object repositories such as [7, 13] and select resources by the appropriate categories. While this may introduce resources that are not relevant for the given topic, these will be soon identified through the continuous feedback by learners interacting with the system.

Obtaining the feedback. When learners are presented with resources, they will be required, or greatly encouraged, to provide short, instant feedback in the form of rating. For the dependency extraction to work, it is required to know if the resource was useful or if it relies on content beyond what the learner believes he/she should know and as such, the feedback needs to reflect this. A star *usefulness* rating will perhaps be sufficient and this can be converted into a utility value as expected by the dependency extraction algorithm.

Mining the dependencies. Given a set of learner ratings, we need to extract any potential dependencies. We have developed an algorithm for mining a ratings based dataset to extract potential dependencies. Our approach has been inspired by the data mining problems Association Rule Mining and Sequential Pattern Analysis and is described later in Section 4.

Generating intelligent suggestions. Once the dependencies have been extracted and merged into a graph, the system can then make suggestions based on:

- Resources the learner has seen
- Resources other learners have liked
- The dependency graph

The dependency graph drives the suggestion process as this is the core component that produces the learning path for the learner.

The following section describes a scenario of how the system works on a small example.

3.2 Sample scenario of recommendations

We provide a description on how we propose the sequence based recommender system will recommend resources to a learner. For our sample scenario, we use the dependency graph as shown in Figure 3, which is simple yet contains different types of dependencies (AND, OR dependencies as well as independant resources).

Our pool of resources contains eight resources which are labeled A to H. Each edge in this graph represents a dependency between resources. For example, E depends on either D or C. A special case is the dependency for C which depends on both A and B, but not in any particular order. Each dependency represents what the learner should have seen in order to find the current resource useful. All of these resources are targeted at a particular topic.

A learner who wishes to learn about this topic logs into the system. Initially, the learner has not seen any resources and so the system will select suitable resources for the learner to begin with. The dependency graph is consulted and resources at the leaves, A, B and D are suggested to the learner. We also indicate

Seen	$\emptyset$
Suggested	$A, B, D\ (C, E)$
Other	F, G, H

(a) Iteration 1

Seen	B
Suggested	$A\ (C, E)$
Other	D, F, G, H

(b) Iteration 2

Seen	B, G
Suggested	$A\ (C, E)$
Other	D, F, H

(c) Iteration 3

Seen	B, G, A
Suggested	$C\ (E)$
Other	F, H

(d) Iteration 4

Seen	B, G, A, C
Suggested	E
Other	F, H

(e) Iteration 5

Seen	B, G, A, C, E
Suggested	$\emptyset$
Other	F, H

(f) Iteration 6

Figure 2: A trace of a learner's interactions with the system

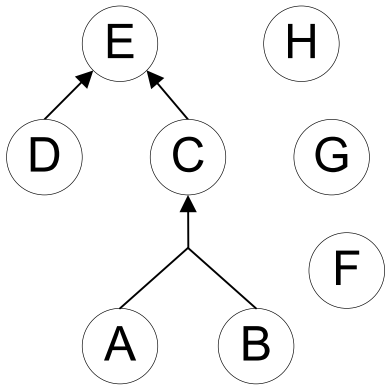

Figure 3: A Dependency Graph

resources that the learner should not see at this point in time, C and E, next to these suggestions in brackets. Finally, any resource that is not connected to the graph is added to an *other* list, so they are not hidden from the user. Figure 2(a) shows the current state of the suggestions for the user.

The learner selects resource B out of the suggested resources. As the dependency graph contains a dependency $\{A, B\} \rightarrow C$, it will suggest A next as B is thought to complement A. This is reflected in Figure 2(b). Figure 2(c) shows that the learner selected G, which does not change the suggestions. Once the learner has seen the suggested resource A (Figure 2(d)), C is selected (Figure 2(e)) and finally, E (Figure 2(f)).

The scenario described shows how we can leverage the dependency graph to present the learner resources they can follow in a suitable sequence. As this is a recommender system and it requires input from its users to make better suggestions over time, it is important for the users to have access to the full set of resources, not just the ones the system believes should be recommended. Over time, as ratings are collected from learners, the system should be able to learn and solidify the relationships between the resources.

For example, it might be the case that G is a resource that is only useful if seen after E as it contains advanced material which requires the knowledge taught in E. It also might be the case that D is not a very useful resource or is unrelated to the topic at hand and thus should be removed from the system. After a number of learners have used the system and provided feedback showing that D is not a very useful resource and that H is useful if viewed after E, the dependency graph should be modified to take the form of Figure 4.

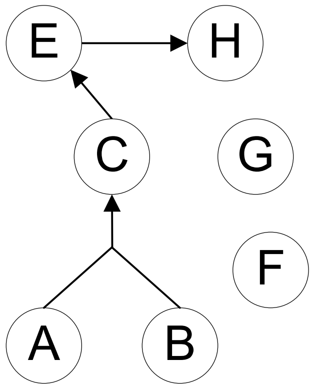

Figure 4: Modified Dependency Graph

Thus this approach differs from traditional recommender systems in that we take sequencing into account. It also differs from other learning based systems in that it works on the feedback (ratings) obtained from the learner, not through metadata records or static learning paths.

We will next describe the important pillar of this approach, the algorithm that mines the dependencies in the learner ratings of resources.

4 Dependency learner algorithm

We describe the main ideas behind our algorithm and how it is used to extract dependencies from datasets. While the algorithm we have implemented is based on this approach, it is similar to Association Rule Mining [1] in that it extracts dependencies satisfying a minimum support and confidence threshold. This makes it more robust so it can handle incomplete datasets (datasets which may not have all the information necessary to extract all dependencies with reasonable confidence) and noise. For the purpose of this paper, we focus on describing a simpler algorithm that extracts dependencies from a consistent dataset with no noise.

ID	Feedback
1	A_+, B_+, C_+
2	A_+, C_-, B_+
3	B_-, A_+, C_-
4	B_-, C_-, A_+
5	C_-, A_+, B_+
6	C_-, B_-, A_+

Table 1: Sample dataset

The task Given a dataset of sequences (records), where each element is an item-utility pair, the goal is to find the dependencies between the items.

For example, given the dataset in Table 1 which contains 6 sequences and 3 items $\{A, B, C\}$, we want to find the dependencies between these items. For the e-learning domain, the items corresponds to resources and the utility values to ratings. Each rating is either a positive (+) or negative (-) rating as denoted by the subscripts in the dataset. For example, if record 1 in the dataset corresponded to the ratings history of a learner, the learner will have viewed resources A, B and C in this order and rated them all positively. The learned dependencies can be represented as a graph where the nodes corresponds to items and the links to dependencies.

4.1 Algorithm rationale

The main idea of the algorithm is to take into account the items rated positively before each positive or negatively rated item in each sequence. We illustrate three different cases for some item i.

1. Any positively rated items that appear before a positively rated i *may collectively be* a dependency for i. For example, in record 1, C has been rated positively thus A or B, both A and B or neither may be a dependency for C.

2. Any positively rated items that appear before a negatively rated i implies that these are not dependencies for i. For example, in record 2 C was given a negative rating. A appears before C with a positive rating which implies that the existence of A is not enough to result in a positive C. There is not enough evidence that C depends on A.

3. Any negatively rated item j that appears before i is ignored. This is because j may have been rated negatively as *its* dependencies did not appear before it. Record 3 has a negatively rated item B appearing before a negative C. We can not exclude B from being a dependency of C as it might be the case that a positive A needs to occur before B to result in a positive B. This situation is actually supported by record 1.

4.2 Algorithm Description

The algorithm consists of the following steps:

For each item $i \in I$, the set of items:

Step 1. Create two *projected datasets* P_{i+} and P_{i-} for the positively and negatively rated items, i_+ and i_-, respectively. A projected dataset for item i is the set of positive items that occur before i in each sequence of the source dataset.

Step 2. Remove from P_{i+}, itemsets in P_{i-} as they do not impact on the ratings of i, creating the potential dependency set D_i.

Step 3. Find the smallest set of itemsets from D_i which describes the dependencies.

Finally the identified dependencies are merged into a dependency graph. This is a directed graph with each node corresponding to an item and each edge to a dependency. The graph is created such that the items on each path from a leaf to some item i is contained within the dependency set D_i.

4.3 Example

In this section, we apply the dependency learning algorithm on the dataset in Table 1.

Step 1. The first step is to take each item and find the items with a positive rating that preceded them in each record. This is done for both positively and negatively rated items. For example, the first record contains a positive B and there is only one positive item, A, preceding it. Thus considering only record 1, $P_{B+} = \{A\}$. This relationship can be used to imply a dependency between A and B, $(A \rightarrow B)$. Note, that if in a subsequent record, we also find the $(B \rightarrow A)$ dependency, then this will cancel out the previous one resulting in no dependency between A and B.

The project datasets for the three items, A, B and C are shown in Figure 5.

For our example:

$P_{A+} = \{\emptyset\}$, $P_{A-} = \{\emptyset\}$
$P_{B+} = \{A, A, A\}$, $P_{B-} = \{\emptyset\}$
$P_{C+} = \{(A, B)\}$, $P_{C-} = \{A, A\}$

Step 2. Next, we work out D_i which is defined as P_{i+} with all itemsets of P_{i-} removed. As i is not dependent on itemsets in P_{i-}, then by removing these from P_{i+} we only consider the items that may be dependent on i.

For example, if $P_{A+} = \{B, C, D\}$, $P_{A-} = \{B, C\}$, then $D_A = \{D\}$ as this is the only set of items that matters as A can not be dependent on $\{B, C\}$.

For our example:

$D_A = \{\emptyset\} - \{\emptyset\} = \{\emptyset\}$
$D_B = \{A, A, A\} - \{\emptyset\} = \{A, A, A\}$
$D_C = \{(A, B)\} - \{A, A\} = \{B\}$

Step 3. Now we take the smallest subset of the potential dependencies, D_i. This involves finding the

ID	P_{A+}	P_{A-}
1	$\emptyset$	
2	$\emptyset$	
3	$\emptyset$	
4	$\emptyset$	
5	$\emptyset$	
6	$\emptyset$	

(a) Projected datasets for A

ID	P_{B+}	P_{B-}
1	A	
2	A	
3		$\emptyset$
4		$\emptyset$
5	A	
6		$\emptyset$

(b) Projected datasets for B

ID	P_{C+}	P_{C-}
1	$\{A, B\}$	
2		A
3		A
4		$\emptyset$
5		$\emptyset$
6		$\emptyset$

(c) Projected datasets for C

Figure 5: Projected datasets for the dataset in Table 1

smallest set of items which cover the full set of dependencies for the given item. Given any sequence X and Y if X is contained within Y, we can remove Y.

For example, given that the current item is A, and $X = \{B, C\}$ and $Y = \{B, C, D\}$ where $\{X, Y\} \in D_A$, we can remove Y as X is contained within Y. This means that A will be positive if positive items B and C exist before it in sequence. As this subsequence exists in both X and Y, we simply keep the smaller of the two and discard the other.

For our example:

$D_A = \{\emptyset\}$

$D_B = \{A\}$

$D_C = \{B\}$

Finally, we build the graph of dependencies. Every item i is dependent on each item within Y given $Y \in D_i$. (Note that if there are multiple dependencies, $|D_i| > 1$, i will be positive if at least one of the dependencies exists before i in a sequence.)

For our example:

$A \rightarrow B$

$B \rightarrow C$

Thus the dependency graph is $A \rightarrow B \rightarrow C$.

5 Experiments

Evaluating our approach and proposed system involves a live user experiment [9] which is timely to perform. Before conducting this expensive stage in our research, we first needed to thoroughly evaluate our algorithm in a simulated environment in order to 1) assess whether our algorithm can learn a dependency graph based on simulated user ratings and 2) gain understanding on the impact of numerous factors on its effectiveness and efficiency. We have conducted a comprehensive set of experiments of which we will present some results.

We have developed a simulation framework in which an abstract version of the resource suggestion problem can be modeled. The simulation framework requires a dependency graph to be defined which reflects the *real* dependencies of the resources. We then pass a number of artificial learners through the simulation which provides a rating for each resource depending on what the artificial learner has seen. Each dependency graph contains a set of *goal* nodes in which the artificial learner aims to reach - this is akin to finishing a topic by looking at all suitable resources until the topic is learnt. Once a goal node is reached, the artificial learner stops requesting resources and exits the simulation. Artificial learners select suggested resources 85% of the time, otherwise selects a resource from the *other* category. (Refer to Figure 2 for the three types of recommendations our system presents). Currently we deal with a consistent scenario, such that if an artificial learner has seen and liked all of the dependencies for some resource r it will rate r as being positive. We have performed two main types of experiments - those that explore how well the dependency graph can be learnt and those that compared different recommendation techniques. The selected experiments presented in this paper were run 100 times with the results averaged.

The first set of experiments we ran aimed to explore our algorithms ability to learn dependencies. For each run of the experiment, we compared the learnt dependency graph with the real dependency graph, identifying correct, missing and incorrect edges. Results are presented for three basic types of dependency graphs - a linear path, a binary bottom up tree and a binary top down tree as shown in Figure 6.

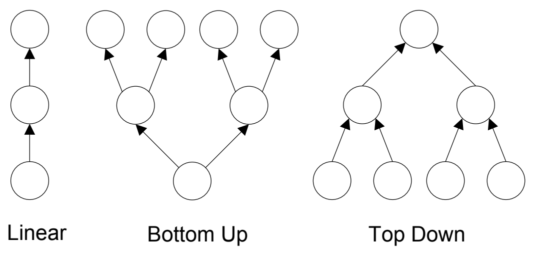

Figure 6: Graph structures experimented with.

The linear path contains a number of resources which all need to be viewed in a specific (linear) order to reach the goal. A binary bottom up tree contains a single leaf (root) node, branching out and ultimately leading to one of many goal nodes. A binary top down tree contains many leaf nodes leading to a single goal node. Figure 7 and Figure 8 show the recall and precision results for each learnt dependency graph. In this experiment, the dependency graph is relearnt every 5 artificial learners. The linear graph contains 10 nodes, while the binary trees have a depth of 3 (thus 4 nodes need to be viewed to reach a goal, however there are 8 different paths to this goal).

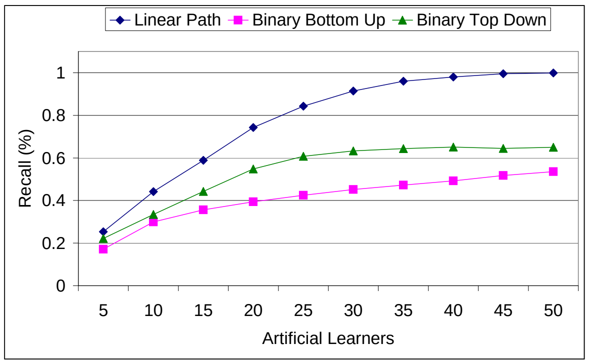

Figure 7: Recall (proportion of correctly learnt edges out of all correct edges) for three different dependency graphs.

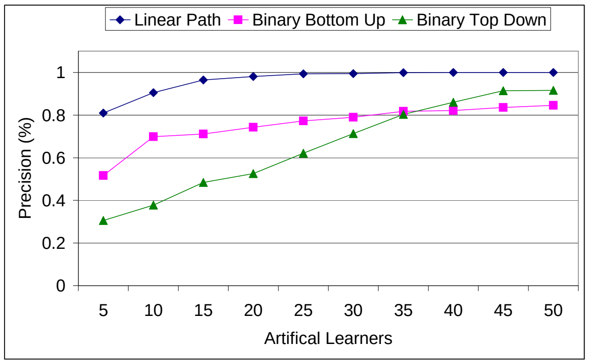

Figure 8: Precision (proportion of correct edges out of all learnt edges) for three different dependency graphs.

The graphs show that the learnt dependency graph is improving over time as more data is collected, particularly in the trivial linear path case. When multiple paths to a goal node exist, as in the case for the binary trees, it is harder to learn the full graph as once a path to a goal node has been identified, other paths are not generally learnt. This is a side effect of the artificial simulation - we propose that when selecting resources, real learners will use additional contextual information such as a resources title and description in addition to the recommendations made by the system. This is one critical aspect that requires future research.

The other main set of experiments we performed focused on comparing different recommendation techniques. We present some results based on a binary bottom up tree (described previously) with a depth of 3. Figure 9 shows how many resources an artificial learner needed to select before reaching its goal. This is something we would like to minimise, to avoid presenting redundant or useless resources to the learner. Figure 9 shows the percentage of resources that the artificial learner rated positive, out of all resources it selected. The system should avoid recommending (by explicitly discouraging) resources which should not be selected as its dependencies have not yet been selected. We see that randomly suggesting resources delivers poor results, due to the unlikely nature that suggestions will be made in a suitable order. Furthermore, suggesting resources based on its popularity alone (top rated) tends to result in nearly all resources being selected in order to reach a goal. (Resources at the lower level of a dependency graph will be rated positive more often than those at the top). Supporting the recommendation process via the learnt dependency graph clearly outperforms both random and top rated as it suggests a minimal number of resources that allows the learner to reach its goal.

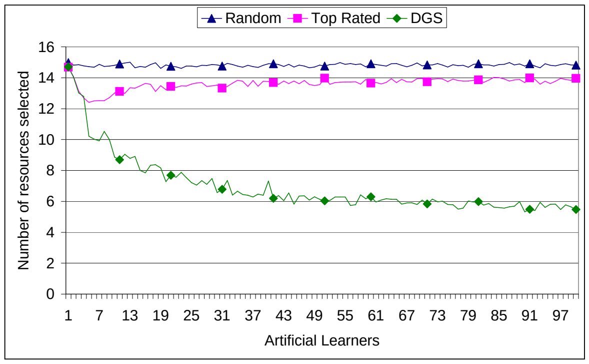

Figure 9: Number of resources selected in order to reach the goal.

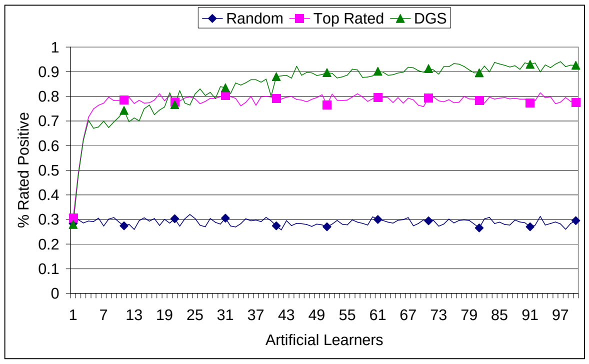

Figure 10: Percentage of resources rated positive out of those selected.

Thus, the simulation results show that the proposed approach can learn dependencies, and paves the way for a trial with real users.

6 Conclusion

We have presented an approach for recommending sequences of learning resources based on previous users' ratings. The novelty of this approach resides in the creation and use of a dependency learner algorithm. This work is still in progress and avenues for future work are numerous. We have implemented the algorithm and ran tests on simulated data which show very encouraging results. Whilst the e-learning context gave us the motivation for this work, it clearly can be applied to any other domains where the order in which users see resources is important.

References

[1] Rakesh Agrawal, Tomasz Imielinski and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA, 1993. ACM Press.

[2] Amazon. http://www.amazon.com/, Accessed 24th April 2006.

[3] T. Berners-Lee, J. Hendler and O. Lassila. The semantic web. *Scientific American*, Volume 284, pages 34–43, 2001.

[4] C. Brooks and G. McCalla. Towards flexible learning object metadata. In *Int. J. Cont. Engineering Education and Lifelong Learning*, Volume 16, pages 50–63, 2006.

[5] Dublin Core. http://dublincore.org/, Accessed 11th Oct 2006.

[6] del.icio.us. http://del.icio.us/, Accessed 11th Oct 2006.

[7] Edna. http://www.edna.edu.au/, Accessed 11th Oct 2006.

[8] N. Frizen. Final report on the "international lom survey", 09 2004.

[9] J.L. Herlocker, J.A. Konstan, L.G. Terveen and J.T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, Volume 22, Number 1, pages 5–53, 2004.

[10] IEEE. http://www.ieee.org/, Accessed 11th Oct 2006.

[11] IEEE-LSTC. The learning object metadata standard. http://ieeeltsc.org/wg12LOM/lomDescription, Accessed 25th April 2006.

[12] S. Jung, J. Kim and J.L. Herlocker. Applying collaborative filtering for efficient document search. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI04)*, pages 640–643, 2004.

[13] MERLOT. http://www.merlot.org/, Accessed 11th Oct 2006.

[14] S. Middleton, D. De Roure and N. Shadbolt. Capturing knowledge of user preferences: ontologies in recommender systems. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, pages 100–107, New York, NY, USA, 2001. ACM Press.

[15] P. Mohan and C. Brooks. Learning objects on the semantic web. In *Advanced Learning Technologies, 2003. Proceedings. The 3rd IEEE International Conference on*, pages 195–199, 2003.

[16] MovieLens. http://movielens.umn.edu/, Accessed 24th April 2006.

[17] F. Neven and E. Duval. Reusable Learning Objects: a Survey of LOM-Based Repositories. 2002.

[18] P. Polsani. Use and abuse of reusable learning objects. *Journal of Digital Information*, Volume 3, Number 4, February 2003.

[19] L.P. Santacruz-Valencia, I. Aedo, A. Navarro and C.D. Kloos. An ontology-based mechanism for assembling learning objects. In *Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/ELearning on Telecommunications Workshop*, pages 472–477, 2005.

[20] T. Ya Tang and G. McCalla. Mining implicit ratings for focused collaborative filtering for paper recommendations. In *UM03' Workshop on User and Group models for web-based adaptive collaborative environments*, 2003.

[21] D. Wiley. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. 2002.