Knowledge Component Suggestion for Untagged Content in an Intelligent Tutoring System

Mario Karlovčec, Mariheida Córdova-Sánchez, Zachary A. Pardos

Abstract. Tagging educational content with knowledge components (KC) is key to providing useable reports to teachers and for use by assessment algorithms to determine knowledge component mastery. With many systems using fine-grained KC models that range from dozens to hundreds of KCs, the task of tagging new content with KCs can be a laborious and time consuming one. This can often result in content being left untagged. This paper describes a system to assist content developers with the task of assigning KCs by suggesting knowledge components for their content based on the text and its similarity to other expert-labeled content already on the system. Two approaches are explored for the suggestion engine. The first is based on support vector machines text classifier. The second utilizes K-nearest neighbor algorithms employed in the Lemur search engine. Experiments show that KCs suggestions were highly accurate.

Keywords. Intelligent Tutoring Systems, Text Mining, Knowledge Components, TextGarden, Lemur, Bag-of-Words

1 Introduction

When designing exercises within the learning software, appropriate knowledge components should be assigned to them. "A knowledge component is a description of a mental structure or process that a learner uses, alone or in combination with other knowledge components, to accomplish steps in a task or a problem." [1] The process of assigning knowledge components to the exercises can be a time consuming job, since the number of possible knowledge components can be very large. In order to help the tutor or course designer in writing exercises we have proposed two approaches that suggest knowledge components. The first approach is based on text mining [2] and SVM classification algorithm and the second is based on a search engine with a KNN classification algorithm [3]. These two approaches can be used for a system that could encourage the course designers to assign knowledge components to new exercises they design, as well as to existing exercises that do not have knowledge components assigned to them.

2 Related Work

This work continues the line of research proposed by Rose *et al*. [4] and expands on the prior art by applying a variety of optimizations as well as evaluating the algorithms on numerous KC models of varying granularity. The work by Rose et al. presented KC prediction results on a model of 39 KCs but skill models have since increased in complexity. We investigate how KC prediction accuracy scales with larger KC models and which algorithms adequately meet this challenge.

The necessity of associating knowledge components with problem solving items is shared by a number of tutoring systems including The Andes physics tutor [5], The Cognitive Tutors [6] and the ASSISTments Platform [7]. The Andes and Cognitive tutors use student modeling to determine the amount of practice each individual student needs for each KC. The student model that these tutors use is called Knowledge Tracing [6], which infers student knowledge over time from the history of student performance on items of a particular KC. This model depends on the quality of the KC model to make accurate predictions of knowledge.

The KC association with items in a tutor is typically represented in an *Item* × *KC* lookup table called a Q-matrix [8]. Methods such as Learning Factors Analysis [9] have been proposed to automate the improvement of this Q-matrix in order to improve the performance of the student model. Recently, non-negative matrix factorization methods have been applied in order to induce this Q-matrix from data [10]. The results of this work are promising but its applications so far are limited to test data where there is no learning occurring and only to datasets with only around five KCs, where these KCs represent entirely different high level topic areas such as Math and English which do not intersect. All the student modeling and Q-matrix manipulation methods have so far not tapped any information in the text of the items they are evaluating. This paper will make the contribution of looking at this source of information for making accurate KC predictions. While this paper focuses on text mined KC suggestion to aid content developers, this technique is relevant to those interested in Q-matrix improvement as well.

3 The ASSISTments Platform

The dataset we evaluated comes from The ASSISTments Platform. The ASSISTments platform is a web based tutoring system that assists students in learning, while it gives teachers assessment of their students' progress. The system started in 2004 with a focus on 8th grade mathematics, in particular helping students pass the Massachusetts state test. It has since expanded to include 6th through 12th grade math and scientific inquiry content.

A feature that sets ASSISTments apart from other systems is its robust web based content building interface [7] that is designed for rapid content development by system experts and teachers alike. Teachers are responsible for a growing majority of the content in ASSISTments. While the content has been vetted and verified as being of educational value by ASSISTments system maintainers, the content often lacks meta

information such as KC tagging as this is an optional step in content creation. An ASSISTments administrator must add this tagging or leave it blank which can cause a lack of accuracy in student model analysis of the data and also inhibits the system from reporting KC information to teachers. The tagging has to be performed by selecting from the large list of KC, which are organized into 5 categories and sorted alphabetically within the categories. Untagged content in ASSISTments is a growing phenomenon with only 29% of the content possessing KC tags as of this writing. Accurate KC suggestion would expedite the processes of content tagging and encourage external content builders to tag their content.

4 Data

The dataset used for testing the performance of the proposed approaches was taken from tagged content on the ASSISTments Platform during the 2005-2006 school year. The ASSISTments Platform has three KC models consisting of varying degrees of granularity. The first two models, containing 5 and 39 KCs, use KC names corresponding to the Massachusetts state math standards. The system's finest-grained KC model contains 106 KCs which were created in-house [11]. The KCs from the 106 model have a hierarchical relationship to the 39 KC and 5 KC models. This allows content to be tagged only with the 106 KCs and then inherit the KCs from the other models in the hierarchy. While tagging with the 106 model is preferred, content builders can choose from KCs from any model to tag their content.

5 Approaches

In order to solve the problem of assigning appropriate KCs by providing automatic suggestion system in the process of exercise design, two approaches are suggested: a text mining approach using the SVM classifier and the search engine based approach with the KNN classifier.

5.1 Text Mining Approach with SVM classifier

One approach was based on text mining and building SVM classification model using the Text Garden [12] utility. It has been shown [2] that the SVM is an appropriate method for text classification. The main reasons include the ability to handle high dimensional input space and suitability for problems with dense concepts and sparse instances. The classification model was built based on set of labeled exercises. We wanted to test the influence of stop words removal and stemming on the classification problem, so four different classification models were built, covering all combinations of applying these standard text processing techniques.

5.2 Search Engine and KNN approach

The second approach was to use the Lemur Toolkit [3] with a K Nearest Neighbors (KNN) classification algorithm. KNN is a commonly used algorithm that finds the K documents closest (most similar) to the document being tested. The Lemur Toolkit is an open source search engine. The questions in the training set were indexed using Lemur. The text of the test set questions were then used as queries against the indexed questions. The top k (in this case k=200) most relevant search results, most relevent to the query, were retrieved (along with their KC tag). Each retrieved document was assigned a score based on its rank (e.g. the score of the top document is 200, the score of the second retrieved document is 199, and so on). We calculate a score for a tag as

$$tag\_score(t) = \frac{a}{\Sigma\, score(t) + b}$$

where $\Sigma\, score(t)$ is the summation of all document scores with tag *t*, and *a* and *b* were both chosen to be two times the KC model size. This is done to predict KCs using a weighted measure of the frequencies of tags (i.e. KCs) and their retrieval ranks. Lastly, for each unlabeled question (query), the tag with the highest *tag_score* is assigned to it.

6 Results

Testing was performed using the 5 folds cross validation method. All experiments used accuracy as the goodness metric. For both approaches, testing was performed on the three different knowledge component models: the largest model with 106 KCs, the 39 KC model and the 5 KC model. In [13], the automatic text generation of mathematical word problems is performed. The paper shows that leaving out the common text processing techniques, namely stop word removal and stemming, can increase the performance of text categorization. To take into account the findings of that paper, we tested each dataset with four different text processing setting: (1) without applying stop-words removal and stemming (SVM); (2) applying only stop-words removal; (3) applying only stemming; and (4) applying both stop-words removal and stemming.

Table 1. Experimental results of proposed approaches for suggesting knowledge components

Dataset	*Number of suggestions*									
	SVM					KNN				
	1	2	3	4	5	1	2	3	4	5
106 KC	0.607	0.739	0.784	0.809	0.823	**0.574**	**0.736**	**0.796**	**0.835**	0.865
106 KC ST	**0.621**	**0.749**	**0.798**	**0.824**	**0.842**	0.567	0.728	0.795	0.834	**0.866**
39 KC	0.683	0.815	0.863	0.895	0.914	**0.666**	0.815	0.854	0.898	**0.914**
39 KC ST	**0.689**	**0.818**	**0.870**	**0.901**	**0.916**	0.653	**0.829**	**0.865**	**0.907**	**0.914**
5 KC	0.814	**0.943**	**0.969**	0.981	1.000	0.762	0.919	**0.976**	0.993	1.000
5 KC ST	**0.815**	0.938	**0.969**	**0.983**	1.000	**0.784**	**0.923**	**0.976**	**0.996**	1.000

The experimental results of both approaches are shown in Table 1. The table shows

accuracy results given KC suggestions ranging from 1 to 5. The accuracy when suggesting 5 KCs, for example, is the percentage of exercises where the correct KC was among the top 5 suggested KCs. For the 5 KC model, 5 suggestions always results in 100% accuracy. Each row represents a different dataset with classification algorithm and text processing settings used in the experiment. 106KC, 39KC and 5KC are labels for the different knowledge components models. SVM and KNN are labels for the two different classification algorithms. ST indicates applying stemming. Each column of the table represents different number of suggestions. Performance of stop-words removal did worse than stemming. Performance of stop-words removal in addition to stemming also did worse than just stemming. These results were not shown in the table for space reasons. The results in this table are represented with sensitivity. Sensitivity in information retrieval is recall for the binary classification problems. It is the probability that a relevant KC is suggested for the exercise. The Kappa value for the 39 KC and 106 KC model was 0.669 and 0.610 respectively.

7 Discussion

Results of the experimental testing indicate that the proposed approaches are suitable for practical usage. Table 1 show the results, which are grouped according to the model of KC used for testing. The SVM classifier with stemming performs the best for every KC model. The dataset with 106 KC model is the hardest challenge for the proposed approach, but this is the KC model for which the system can be mostly useful in practical application. If only one KC is suggested for the 106 KC model, it would be the correct one in 62.1% of the cases. Suggestion systems usually suggest more than one option, if the number of these suggestions is 5; the correct KC is among these 5 in 84.2 % of the cases, or in 88.9% of the cases if there are 10 suggestions. If the number of suggestions increases, the probability that the correct KC is among them naturally grows, but the effort required from the user to choose the correct KC among the suggested also increases. Comparing the results with all four combinations of typical text processing procedures applied (stop-word removal and stemming), not removing stop-words and performing stemming improves the accuracy of the system as suggested by [13]. However this improvement is much less significant than in the referenced paper. The improvement for the best options in comparison with the worst option - removing stop-words and no stemming, were around 2%.

Results indicate that both suggested approaches are suitable for practical usage, since they would decrease significantly the number of KCs to be used for labeling, without compromising much on efficiency (i.e. failing to show the correct labels).

Acknowledgements: This research was supported by the National Science Foundation via the "Graduates in K-12 Education" (GK-12) Fellowship, award number DGE0742503. We would like to thank the additional funders of the ASSISTments Platform found here: http://www.webcitation.org/5ym157Yfr. We would also like to thank Geoff Gordon, Ken Koedinger, John Stamper and the other organizers of the Pittsburgh Science of Learning Center EDM summer program.

References

1. Pittsburgh Science for Learning Center, "LearnLab," [Online]. Available: http://www.learnlab.org/research/wiki/index.php/Knowledge_component. [Accessed 15 11 2011].
2. T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *Proceedings of ECML-98, 10th European Conference on Machine Learning*, Chemnitz, 1998.
3. University of Massachusetts and Carnegie Mellon Uni-versity, "The Lemur Project," www.lemurproject.org.
4. C. Rosé, P. Donmez, G. Gweon, A. Knight and B. Junker, "Automatic and Semi-Automatic Skill Coding with a View Towards Supporting On-Line Assesment," in *Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology*, Amsterdam, 2005.
5. A. S. Gertner and K. VanLehn, "Andes: A coached problem solving envi-ronment for physics," *In G. Gautheier & C. Frasson & K. VanLehn (Eds.), Intelligent Tutoring Systems: 5th international Conference, ITS 2000,* pp. 133-142, 2000.
6. K. R. Koedinger, J. R. Anderson, W. H. Hadley and M. A. Mark, "Intelligent tutoring goes to school in the big city," *International Journal of Artificial Intelligence in Education,* vol. 8, no. 1, pp. 30-43, 1997.
7. L. Razzaq, J. Patvarczki, S. F. Almeida, M. Vartak, M. Feng, N. T. Heffernan and K. R. Koedinger, "The ASSISTment builder: Supporting the Life-cycle of ITS Content Creation," *IEEE Transactions on Learning Technologies Special Issue on Real-World Applications of Intelligent Tutoring Systems,* vol. 2, no. 2, pp. 157-166, 2009.
8. M. Birenbaum, A. E. Kelly and K. K. Tatsuoka, "Diagnosing knowledge states in algebra using the rule-space model," *Journal for Research in Mathematics Education,* vol. 24, no. 5, pp. 442-459, 1993.
9. H. Cen, K. Koedinger and B. Junker, "Learning Factors Analysis - A general method for cognitive model evaluation and improvement," in *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 2006.
10. M. C. Desmarais, "Conditions for effectively deriving a Q-Matrix from data with Non-negative Matrix Factorization," in *In Proceedings of Educational Data Mining 2011*, Eindhoven, Netherlands, 2011.
11. L. Razzaq, N. T. Heffernan, M. Feng and Z. A. Pardos, "Developing Fine-Grained Transfer Models in the ASSISTment System," *Journal of Technology, Instruction, Cognition, and Learning,* vol. 5, no. 3, pp. 289-304, 2007.
12. Artificial Intelligence Laboratory - Institute Jozef Stefan, "Artificial Intelligence Laboratory," [Online]. Available: http://ailab.ijs.si/tools/text-garden/. [Accessed 15 11 2011].
13. S. Cetinas, L. S. Si, Y. X. Ping, D. Zhang and J. P. Young , "Automatic Text Categorization of Mathematical Word Problems," in *Proceedings of the Twenty-Second International FLAIRS Conference*, Florida, 2009.