An Annotations Approach to Peer Tutoring

John Champaign and Robin Cohen
jchampai@cs.uwaterloo.ca, rcohen@ai.uwaterloo.ca
David R. Cheriton School of Computer Science, University of Waterloo

Abstract. In this paper we detail a preliminary model for reasoning about annotating learning objects and intelligently showing annotations to users who will benefit from them. Student interactions with these annotations are recorded and this data is used to reason about the best combination of annotations and learning objects to show to a specific student. Motivating examples and algorithms for reasoning about annotations are presented. The proposed approach leverages the votes for and against an annotation by previous students, considering whether those students are similar or dissimilar to the current student, in order to determine the value of showing this annotation to the student.

1 Introduction

McCalla's ecological approach to e-learning systems [8] is described as "attaching models of learners to the learning objects they interact with, and then mining these models for patterns that are useful for various purposes." Using techniques inspired by collaborative filtering, the basis of this approach is to identify which users in a system are similar to each other, to then preferentially recommend learning objects that similar students have found useful. Learning objects are used in the ecological sense, in that instead of attaching static ontologies to objects as metadata, models of previous interactions (i.e. presenting learning objects to students) are used in their place. These models are then actively interpreted, with the meaning derived by real-time processes, as information about the object is needed.

Since McCalla's ecological approach is intended to primarily be a general philosophy for designing intelligent systems, individual researchers may then create their own algorithms and systems to embody this approach.

To honour the basic evolutionary approach, we are particularly interested in exploring the use of student annotations: allowing students to leave short comments on learning objects they are interacting with. More than simply tags, this could be a question or a commentary about what they're learning. Subsequent students would identify which annotations they found useful, which would then be intelligently shown to similar students. Asynchronous collaboration or, at least, to allowing the interactions of the student in the past to inform the interaction with the current student, honours the ecological approach [8].

In developing a model for reasoning about which learning objects and annotations to show to students, we are thus focused on mining the repository to improve the educational data sets of use in the tutoring process.

2 Background

In McCalla's ecological paradigm for intelligent tutoring system, the basis for tutoring is a repository of learning objects (which, for example, could be a chapter from a book, a video or an exercise which is completed and evaluated), which are provided to students to interact with. Information about these interactions is used to reason about the best learning objects to be shown in the future to other students, based on the utility of past interactions of similar students with the various objects in the system.

In previous research, we explored how best to select learning objects to present to students by reasoning about the benefits derived by previous students from these learning objects [4]. Beyond implicitly reasoning about interactions, in this work we consider extending this paradigm so as to allow students to explicitly leave information for future students. This takes the form of "annotations", or short text messages, which are attached to the learning object and intelligently shown to (or hidden from) future students. As a motivating example, suppose a student Carol in a Computer Science 101 course was struggling with the concept of procedures. She suddenly realizes that procedure is another name for a function, which is the term her high school teacher had used. She leaves a short annotation ("OMG, I realized half way through the lesson that procedures are the same as functions!!! duh! :-)") on the learning object she was studying with this insight and carries on with her lessons. Future students are shown this annotation and given the option of endorsing it (by clicking on a "thumbs up") or denouncing it (with a "thumbs down"). Over time, the system learns that the annotation tends to be useful to students with a background that used the term "function" instead of "procedure" (and shows it to them), but not to others (and hides it from them).

3 Our Approach

For the actual mechanics of our approach, we use a set of algorithms for dealing with the annotations. Our approach is as follows. We model the overall reputation of a student, initially set as 0.5 and afterwards based on the extent to which the student's previous annotations have been found useful by other students. The authoring student's reputation is used as the initial reputation of an annotation, which is afterwards independently tracked and modeled with its own reputation. Students' ratings of the value of the annotation serve to adjust the overall reputation of the annotation (and indirectly, the annotation author's reputation). Once a student has provided a set of ratings on annotations, it is then possible to reason about their similarity to other students, based on mutually rated annotations. This allows the probability that an annotation be shown to be determined by both the overall quality of that annotation, as rated by all students, adjusted by the similarity between the current student and the students who have previously rated the annotation.

3.1 Annotations Algorithms

```
Function calculateStudentReputation
      Arguments:  Student
      if (numberOfAnnotationsCreatedByStudent == 0)
            // If the student hasn't made any annotations,
            // assign their reputation to 0.5
```

```
            Reputation = 0.5
      else
            Reputation = 0
            // Calculate the average reputation of all annotations
            // made by the student
            for each previous annotation left by student
                  reputation += annotation.reputation
            end for
            Reputation /= numberOfAnnotationCreatedByStudent
      Return:  Reputation (0.0:1.0)
Endfunction
```

Example:

Bob has left his first annotation (a question asking for the definition of a term on the slide) on a CS 115 slide and no one has rated it yet. His reputation is 0.5.

Carol has left three annotations, two on slides and one on a video of the CS 115 instructor explaining Scheme structures (each one discussing how the concepts relate to more advanced concepts she has read independently). After receiving ratings from students presented with the learning object these annotations have reputations of 0.476, 0.551 and 0.704. Carol has a reputation of 0.577 (the average of these 3).

```
Function makeAnnotation
      Arguments:  Student, Learning Object, Annotation
      student.reputation =calculateStudentReputation(student)
      annotation.reputation = student.reputation
      // The annotating student will be considered to implicitly give
      // their own annotations a positive recommendation
      evaluateLearningObjectAnnotation(student, annotation)
      learningObject.attach(annotation)
      Return:  nothing
Endfunction
```

Example:

Allison and Carol each leave an annotation on a new set of example exercises that have been posted (Allison asks if the solutions have been posted, and Carol replies that they will be posted on the March 18th after the assignment due date). Allison's new annotation starts with a reputation of 0.5 (no one has rated his first annotation yet – Carol didn't bother) and Carol's has a reputation of 0.557 (there haven't been any additional ratings to her annotations since her reputation was calculated).

```
Function similarity[1]
      Arguments:  Current Student, Annotation Student
      votedTogether = 0
      votedAgainst = 0
      for each annotation both students voted on
            if (currentStudent.vote == annotationStudent.vote)
                  votedTogether += 1
            else
                  votedAgainst += 1
```

1 Students are considered "similar" to one another if they tend to benefit in the same way from the same learning objects. This straightforward comparison can be expanded as future work.

```
      end for
      similarity = (votedTogether - votedAgainst)
      similarity /= (votedTogether + votedAgainst)
      Return:  Similarity [-1.0:1.0]
Endfunction
```

Example:

Bob has given a "Thumbs up" (indicating approval) to annotations A, B, F & G and a "Thumbs down" to annotations D, M & Y. Carol has given a "Thumbs up" to annotations F, M, N, and Z and a "Thumbs down" to C, G, and Y.

Bob and Carol's similarity is:

$$\frac{2(\textit{they agreed on their assessment of } F,Y) - 1(\textit{they disagreed on their assessment of } G)}{2 + 1(\textit{the number of annotations both have rated})}$$

or 0.5

```
Function show annotation
      Arguments:  learning object, student
      for each annotation attached to learning object
            // base chance of showing is based on author's
            // reputation AT THE TIME THEY MADE the annotation.
            votes_for = annotation.votes_for
            votes_against = annotation.votes_against
            for each vote on annotation
                  sim = similarity(currentStudent,
                                   annotationStudent)
                  if vote.for
                        votes_for += 1 * sim
                  else
                        votes_against += 1 * sim
            AT = arctan(votes_for - votes_against) / pi
            chanceToShow = 0.5 + AT
            randomNumber = generateRandomNumber[0.0 : 1.0]
            if (randomNumber < chanceToShow)
                  show annotation to student
                  // implicit in showing an annotation is that
                  // any parent annotations will also be
                  // shown, this is applied recursively up to
                  // a root annotation
      Return:  boolean
Endfunction
```

Example:

Bob is viewing the learning object annotation Z is attached to. It has a reputation of 0.670 and Carol is the only other student he has a similarity rating with who has previously rated it. She gave it a thumbs up, which leads to the annotation being given a reputation of 0.670 * (1+ (0.05 * 0.5)) = 0.687 for Bob. This means there is a 68.7% chance that this annotation will be shown to him when he uses the learning object.

In the above example, Bob's annotation has a (1 – (1 * 0.5 * 0.453)) or 0.774 (since it may be shown on its own, or will be shown if Carol's reply is shown).

```
Function evaluate learning object annotation
      Arguments:  Learning object, student
      if (student votes on some annotation)
            votedAnnotation.attach(vote, student)
            votes_for = tallyOfForVotes(annotation)
            votes_against = tallyOfAgainstVotes(annotation)
            AT = arctan(votes_for - votes_against) / pi
            annotation.current_reputation = 0.5 + AT
      end if
      Return:  nothing
Endfunction
```

Example:

While Bob uses the learning object, he gives the annotation Z a "Thumbs Down" (he doesn't find it useful). This lowers the annotation's reputation from 0.670 to 0.638. His similarity to Carol is also lowered (now that they disagree on 2 annotations) and they will now have a similarity of 0 (their preferences will no longer influence one another). If either gives another rating that conflicts with the other, they will begin to negatively recommend annotations for one another (what one likes will be less likely to be shown to the other).

The formula which determines whether an annotation is shown or not:

$$\textit{Probability Annotation is Shown} = \frac{1}{2} + \frac{\arctan\left(votes_{for} - votes_{against}\right)}{\pi} \quad (1)$$

has a number of attractive properties.

3.2 Larger Number of Votes Given Greater Weight than Smaller Number

Consider 3 students as voting in favour of an annotation and 5 against it and contrast this with 30 students voting in favour and 50 voting against it. In each case, 37.5% of the student found the annotation useful. However, given that 10 times as many students have voted on the annotation in the second case, we want to take into account the greater certainty in the outcome given the larger number of voters.

Contrast m votes for and n votes against and $k \cdot m$ votes for and $k \cdot n$ votes against.

Formula (1) is a linear scaling of arctan($votes_{for} - votes_{against}$). Given that arctan is a strictly increasing monotonic function, arctan(k (m – n)) > arctan (m – n) and our approach will give a greater weight to larger groups of voters.

3.3 Probability Approaches But Doesn't Reach 0 and 1

Consider:

$$\lim_{x \to \infty} f(x) = \frac{1}{2} + \lim_{x \to \infty} \frac{\arctan\left(votes_{for} - votes_{against}\right)}{\pi} = 1$$

$$\lim_{x \to -\infty} f(x) = \frac{1}{2} + \lim_{x \to -\infty} \frac{\arctan(votes_{for} - votes_{against})}{\pi} = 0$$

Given that arctan is a strictly increasing monotonic function, we know that the probability that an annotation will be shown or not approaches, but never reaches 0 and 1. This is worthwhile as the usefulness of an annotation may be discovered at a later date, and it is then given a chance to be promoted. Conversely, if a well-regarded annotation is shown to be vacuous, the community has a chance to immediately begin decreasing its prominence.

Consider the example of a clarifying annotation where a student made the connection, in a video that explains parameters, that procedure another name for a function. After this annotation was given high ratings, a new article was added which explained functions and clearly presented the various terms such as routine, function, procedure or method. After having read this article, students begin finding the previously useful annotation redundant and it begins to receive negative ratings from current students. The system is immediately responsive to this and each negative vote decreases the probability that this (once highly-regarded) annotation is shown to current students.

4 Validation of Approach

Our intention is to validate this work using simulated students.

Let knowledge be defined as the known concepts in the domain under consideration (the course the ITS endeavours to educate the student on).

$$k = \{set\ of\ known\ concepts\}$$

After an interaction between student s and learning object l, there will be a set of relationship such that:

$$if\ k_s \in k_l\ then\ k_s = k_s \cup k_{new}\ with\ probablity\ p$$

e.g. suppose learning object abc had the relationship:

$$if\ k_s \in \{B,J,U\}\ then\ k_s = k_s \cup \{M\}\ with\ probablity\ 0.25$$

This would imply that if a student using this learning object had attained concepts B, J and U, then upon completion of using this object he would have a 25% chance of attaining concept M.

Let overall knowledge (K) be represented as a percentile, considered roughly analogous to the student's expected mark given their current understanding.

$$K = \frac{(Known\ Concepts)}{(All\ Concepts)}$$

e.g. suppose an ITS had 26 concepts, each represented by a letter the alphabet. Given a student who had obtained concepts B, J, M and U, their overall knowledge would be:

$$K = \frac{(\{B,J,M,U\})}{(A,B,C,\ldots Z)} = \frac{4}{26} = 15\%$$

The goal of the system is to maximize the average K of student's using the system.

5 Annotation

An annotation will modify the relationships of a learning object in one of two ways.

1. Create a new relationship with a substitute, removed concept
2. Increase or decrease the probability of attaining the new concept for an existing relationship

5.1 Example

Student Amy annotates a chapter from a text book that was assigned to her by the system. Her annotation has the effect of creating a new relationship, based on the above example but instead of requiring an understanding of variables, constants, and functions it will now require an understanding of variables, constants and procedures (in order to understand the concept of recursion). Students who use this new learning object with the annotation attached, have two "paths" to obtaining the concept of recursion.

From a "real life" perspective, this could be viewed as Amy relating the learning object to an alternative background (in some way showing that functions are analogous to procedures and the annotation allows students with this alternative background to comprehend the chapter).

Student Bob annotates a video about data structures. His annotation (incorrectly claiming a B+ tree is a B tree written in C++) has the effect of adjusting the probability of an existing relationship. This annotation makes it 10% less likely that students seeing the learning object with Bob's annotation will attain the new concept compared to student who experience the learning object without Bob's annotation. Bob has confused students and prevented them from properly understanding what the video is trying to convey. The system should stigmatize this annotation and prevent it from being shown.

The system will run, and use the reputation and previous ratings to determine which annotations are shown to a student.

6 Related Work

Peer tutoring has been explored by a number of previous researchers. Some work, such as [10] and the COMTELLA project of [11], have investigated annotation techniques such as folksonomies and user tagging. While on the surface, this may seem similar to our work, there are important distinctions. With tagging, the purpose it to have users categorize items in ways that are meaningful for them, with the goal of sidestepping many of the problems inherent with ontologies (as articulated in [8]). In contrast, our approach endeavors to not just help students find an appropriate learning object, but to actually clarify that object and allow students to share insights with one another. Other works, such as [9][10], have been more explicit about arranging peer-tutoring. In their COPPER system, they arrange for students to practice conversations with one another, taking into account each student's level of proficiency, previous interactions and how they can best learn from one another. While our approach is a far less intense interaction than peer-tutoring that reasons about groups and gives them task in order to learn from

one another, our approach has the benefit of allowing asynchronous learning. Students may be able to benefit from annotations left by students who are no longer even in the course.

Other work [11][12] has been done considering text produced by learners, specifically the notes they take. They used these notes and text retrieval techniques to implicitly derive information about the student and to build a profile and social network about them. In contrast, we take an intensely pragmatic view of annotations and don't try to decipher the meaning. Instead, our approach reasons directly about which annotation will help a student learn, and ignores the actually underlying content of the annotations.

iHelp [2] is a project related to COMTELLA that involves reasoning about matching stakeholders (such as students, markers, tutorial assistants and instructors) in order to get the right information to the right person (both in public and private discussions). In [3] the authors extend iHelp to explore the value of tools such as chat rooms where learners are automatically drawn when using learning objects, shared workspaces where multiple learners can edit the same source code while discussing it and visualization tools for indicating a particular student's degree of interaction with her classmates. All of this done to encourage "learner collaboration in and around the artefacts of learning". In contrast, our work seeks to provide repositories of useful information from past students, rather than provide tools to assist in the interactions between current students. In many cases these “past students” may be a classmate who used the learning object the day before, while in others it might be a former student who has since graduated and left the school.

The work of John Lee et al. on the *Vicarious Learner* project [7], investigates how to automatically identify worthwhile dialogs to show to subsequent students, by determining the “critical thinking ratio” of a dialog, generated using a content analysis mark-up scheme. This ratio is determined from the positive and negative aspects within a discussion, with the assumption that discourse patterns provide signs of deeper levels of processing by learners and lead to a “community of enquiry” which benefits students. Dialogs with higher ratios could then be considered as valuable to show to new students. Our work differs from theirs in that we are interested in messages that have been explicitly left for future students and tied to a particular part of the course, rather than data-mining past interactions between students. Additionally, our approach is able to leverage similarities between students, in order to have a user-specific process for deciding which annotation should be shown. It may be interesting to integrate Lee et al.'s automated analysis of the critical thinking of text, as a component of deciding whether an annotation should be shown to a student.

6.1 Incentives

A criticism may be leveled that students won't be interested in helping their classmates (or admitting their ignorance) and therefore won't leave annotations. In the case that the intrinsic benefit aren't enticing to a group of students, various approaches, such as [5][6] could be used to encourage participation. The authors' feeling is that such explicit systems for coercing participation will not be needed in many learning contexts. Learning could be considered an inherently social process that naturally develops as sharing information with other student. Intrinsically this can be thought of as developing

social capital by sharing information, leading to greater trust and respect within the community.

7 Conclusion and Next Steps

We have demonstrated in this work the foundation for an approach for allowing students to explicitly share insights from their educational experiences to similar students going through the same process. Our next step will be to validate this approach, using simulated students following our approach, compared with both random and ideal provision of learning objects to students. Following this, we intend to corroborate our results with a study on real students.

There are various directions as well for extending our current model for reasoning about the annotations to be shown to students. Currently, which annotation is shown to a student is sensitive to the reputation of the annotation and the similarity of the current student to those who have rated this particular annotation. For future work, it is worthwhile to be modeling more extensively the student providing the annotation, assigning a greater weight to those annotations provided by students with greater learning proficiency. It may also be useful to be examining the level of achievement of the student who provided the annotation, considering as more valuable those students who are at a higher level of learning.

In our previous work [4], we presented an algorithm to determine which learning object to present to a student, based not only on their similarity to previous students but also on the extent to which those students benefited in their learning, when using that object. The methods proposed in that model for capturing the gains in learning of students could form a useful starting point for determining how to incorporate the learning proficiency of students into our algorithms for determining which annotations to show.

Another direction for future work is to examine alternative formulae for managing the votes for and against (beyond our current proposal to employ an arctan function). One possibility would be to examine alternative methods for converting the interval to a [0,1] range. Another direction would be to examine in more detail the statistical confidence between the votes that are being registered, as a method of determining the importance of the votes towards the decision of showing a particular annotation. In addition, we are currently troubleshooting the proposed arctan function, as it is desirable to better incorporate the annotator's reputation and still maintain the desired [0,1] range.

It is also worthwhile to be exploring more sophisticated metrics for determining the similarity between two students. Current research in collaborative filtering approaches for recommender systems has suggested more sophisticated techniques for making a recommendation, such as: statistical collaborative filtering, cluster models, and Bayesian networks[1]. We intend to examine whether these techniques may be applicable to our approach to annotations.

Finally, it would be useful to consider our proposal for presenting annotations to learning objects together with other algorithms for determining which learning objects should be presented, as part of the overall tutoring of that student. Our own investigations into curriculum sequencing [4] and peer-based development of the corpus of learning objects [5] would be particularly relevant, here.

References

[1] J. S. Breese, D. Heckerman and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998, p. 43-52. San Francisco

[2] H. Bretzke, and J. Vassileva, Motivating Cooperation in Peer to Peer Networks, *User Modeling*, 2003, pp. 218-227.

[3] C. Brooks, R. Panesar, and J. Greer. Awareness and Collaboration in the iHelp Courses Content Management System. *1st European Conference on Technology Enhanced Learning*, 2006. Crete, Greece.

[4] Champaign, J., and Cohen, R. A Model for Content Sequencing in Intelligent Tutoring Systems Based on the Ecological Approach and Its Validation Through Simulated Students. *Proceedings of the 23rd International FLAIRS Conference*, 2010. Daytona Beach, Florida (to appear).

[5] Champaign, J., and Cohen, R. A Distillation Approach to Refining Learning Objects. *The Third International Conference on Educational Data Mining*, 2010. Pittsburgh, Pennsylvania (to appear).

[6] Cheng, R., and Vassileva, J. Design and evaluation of an adaptive incentive mechanism for sustained educational online communities. *User Modeling and User-Adapted Interaction*, 2006, 16(3-4), p. 321-348.

[7] Lee, J., Dineen, F., and McKendree, J. 1998. Supporting Student Discussions: it isn‘t Just Talk. *Education and Information Technologies*, 1998, 3, 3-4, p. 217-229.

[8] McCalla, G. The Ecological Approach to the Design of E-Learning Environments: Purpose-based capture and use of information about learners. *Journal of Interactive Media in Education: Special Issue on the Educational Semantic Web*, 2004, 7, p. 1–23.

[9] Rambo, N. and Beahler, C. Knowledge-Based Information and Systems.

In *Public Health Informatics and Information Systems*, 2003, p. 352-375.

[10] Read, T., Barros, B., Bárcena, E., and Pancorbo, J. Coalescing individual and collaborative learning to model user linguistic competences. *User Modeling and User-Adapted Interaction*, 2006, 16(3-4), p. 349-376.

[11] Vassileva, J. Toward Social Learning Environments. *IEEE Transactions on Learning Technologies*, 2008, 1(4), p. 199–214.

[12] Wan, X. Jamaliding, Q., and Okamoto, T. Discovering Social Network to Improve Recommender System for Group Learning Support. *Computational Intelligence and Software Engineering*, 2009, Wuhan, China.