Adaptive Engine for MOOC

**Ilia Rushkin, Glenn Lopez,
Andrew Ang, Yigal Rosen**
Cambridge, USA

ABSTRACT
UPDATED—August 11, 2017. Description of an adaptive recommendation engine for serving assessment questions in a MOOC. Prototype in R exists.

1. INTRODUCTION
We would like to create a simple adaptive recommendation engine for an edX MOOC, capable of deciding what item to serve to a user next based on the user's history. Primarily, we are interested in serving assessment items (for this reason, we may use "question" and "item" interchangeably), but instructional materials are also possible. These can be any XBlocks instructional webpages or videos intended to be mixed with assessment items ("if a student has trouble with that question, let them read this" etc.)

We use a variety of Bayesian Knowledge Tracing (BKT) model to estimate the students' state. What makes our situation special is that, as we learned from the adaptive pilot and just generally from seeing MOOCs,

1) Questions in the course differ widely in nature, and in particular in difficulty. Thus, we cannot assign the same values of guess, slip and transit probabilities to them, even if they are all tagged with the same knowledge component.

2) Tagging is complicated: some of the questions are tagged with multiple knowledge components (this is the term we prefer; in other works it is also called "skills" or "learning objectives").

3) In a self-paced MOOC environment, there is a need for a causal structure in the knowledge components: we should not serve to a user items tagged with a knowledge component, if the user has shown lack of knowledge of other knowledge components that are pre-requisite to that one. In the simplest case, it can be dictated by a simple ordered list (the natural order of learning the content of the course), but it could also be a detailed graph of pre-requisite relationships among knowledge components.

4) In a MOOC, the number of students is high, so we can afford to define a model with a large number of parameters and optimize them based on the student interaction data.

2. RELATED LITERATURE
TBD

3. MODEL DESCRIPTION

Knowledge tracing
Let there be Q questions in the course ($q = 1, 2, ... Q$), tagged with N knowledge components ($i = 1, 2, ... N$), or KCs for short, with pre-requisite relationships traced among them. These relations will be involved in the recommendation part of the engine, but not in knowledge tracing. We assume that the mastery of each KC by each course user is a binary latent variable – the user either has learned it or not – and we update the mastery matrix p, where the element p_{ui} is the currently estimated probability that the user u has the mastery of the KC i. We could define the mastery threshold $p^* \in [0, 1]$, and if $p_{ui} \geq p^*$, we say that the mastery of i by the user u is sufficiently certain and no longer needs verification. We could initialize the mastery probability matrix $p = p^{(0)}$ (students' prior knowledge), after which, when the user has been served a question. When the user submits an answer to the question, it gets a correctness value (score) $C_q^{(u)} \in [0, 1]$ and we update the mastery probability of each KC (i.e. this user's row of the matrix p).

The Bayesian updating is easier to write in terms of odds, or even logarithmic odds, rather than the probability p:

$$\mathcal{O}_{ui} = \frac{p_{ui}}{1 - p_{ui}}, \quad L_{ui} = \log \mathcal{O}_{ui}, \quad L^* = \log \frac{p^*}{1 - p^*} \quad (1)$$

So we will translate the transit, guess and slip probabilities into odds as well:

$$o_{qi}^{trans} = \frac{p_{qi}^{trans}}{1 - p_{qi}^{trans}}, \quad o_{qi}^{guess} = \frac{p_{qi}^{guess}}{1 - p_{qi}^{guess}}, \quad o_{qi}^{slip} = \frac{p_{qi}^{slip}}{1 - p_{qi}^{slip}}, \quad (2)$$

Then the update procedure requires defining the likelihood ratios for the case of incorrect (0) and correct (1) answer:

$$x_{qi}^0 = \frac{p_{qi}^{slip}}{1 - p_{qi}^{guess}}, \quad x_{qi}^1 = \frac{1 - p_{qi}^{slip}}{p_{qi}^{guess}} \quad (3)$$

These matrices encode the relevance of a problem q to an KC i. If the problem is irrelevant to a KC, the probability of correct or incorrect score should be independent of that KC. This will be the case if $p_{qi}^{slip} = 1 - p_{qi}^{guess}$, in which case $x_{qi}^0 = x_{qi}^1 = 1$. We propose to define the relevance matrix, used in the recommendation part of the engine, as:

$$k_{qi} = \log x_{qi}^1 - \log x_{qi}^0 = -\log o_{qi}^{guess} - \log o_{qi}^{slip}, \quad (4)$$

a sum of logarithmic odds of non-guessing and non-slipping. While the tagging matrix is binary (a KC is either linked to a question or not), the relevance matrix shows the weight of the link: how much of an evidence for the KC mastery the question provides. The multiplicative factor earned by the mastery odds is:

$$x_{qi} = x_{qi}^0 \left(\frac{x_{qi}^1}{x_{qi}^0} \right)^{C_q^{(u)}} \tag{5}$$

For binary (0 or 1) scores, this factor should equal x_{qi}^0 or x_{qi}^1. How we interpolate between these for fractional scores is a matter of choice. For instance, an alternative definition could be a linear interpolation $x_{qi} = x_{qi}^0 + C_q^{(u)}(x_{qi}^1 - x_{qi}^0)$. We preferred the multiplicative interpolation by looking at the location of the "borderline" score, for which $x_{qi} = 1$, reresenting the boundary between correctness and incorrectness. For instance, as a back-of-the-envelope estimate, let the guess and slip probabilities have equal values (typically, they are not too different). In Eq. 5, this sets the borderline score at a reasonable 0.5, whereas in case of linear interpolation the borderline score in such a situation equals the slip (= guess) probability, which is likely too low.

The posterior odds, with the evidence of the submitted problem, become $\mathcal{O}_{ui} \to \mathcal{O}_{ui} x_{qi}$. Additionally, we modify the mastery odds due to transfer of knowledge ($p_{ui} \to p_{ui} + (1 - p_{ui}) p_{qi}^{trans}$), so the full update procedure is:

$$\mathcal{O}_{ui} \to o_{qi}^{trans} + (o_{qi}^{trans} + 1) \mathcal{O}_{ui} x_{qi} \tag{6}$$

This is a type of Bayesian Knowledge Tracing. The main modification is that we allow fractional scores (by means of interpolation) and questions tagged with multiple KCs, so that the parameters carry the index i. If a problem is tagged with several KCs ($k_{qi} > 0$ for more than one value i), we essentially view the problem as a collection of sub-problems, each tagged with a single KC. The predicted odds $\mathcal{O}_{qu}^{predict}$ correct answer by user u on a question q is found as

$$\mathcal{O}_{qu}^{predict} = \prod_i \frac{\mathcal{O}_{ui}(1 - p_{qi}^{slip}) + p_{qi}^{guess}}{\mathcal{O}_{ui} p_{qi}^{slip} + 1 - p_{qi}^{guess}}, \tag{7}$$

which is to say that we take the ratio of the probability that each sub-problem is answered correctly to the probability that each sub-problem is answered incorrectly (since we must remove from the ensemble the possibilities of correct answer on some but not all sub-problems). The predicted probability of the correct answer to a problem is found from the odds by the standard transformation as: $\mathcal{O}_{qu}^{predict} / (\mathcal{O}_{qu}^{predict} + 1)$.

The outlined procedure is multiplicative in nature. A natural idea would be to replace it with an additive one by working with logarithmic odds L_{ui}, which we do in the recommendation part of the engine. It would be clearly preferable from the computational point of view in the knowledge-tracing part as well, if it was not for the knowledge-transfer step: in the additive formulation this step would involve an exponentiation and a taking a logarithm.

For terminological simplicity we referred to the content items as questions. However, the model can accomodate instructional items as well, e.g. videos or text. We can adopt a rule that, if an item q is instructional, the outcome of user's interaction with it is always "correct". A way to think of it is to imagine that q includes an assessment part of trivial difficulty. The slip probabilities $p_{qi}^{slip} = 0$, the guess probabilities now have the meaning of the probability of not learning an KC from the item, and so we set them to $p_{qi}^{guess} = 1 - p_{qi}^{trans}$. Also, in case of an instructional item, unlike an assessment item, we may or may not want to mark it as "seen", because it is allowed to serve it to a user multiple times (or perhaps we should impose a limit on how many times it can be served).

If the matrix p or other parameter matrices contain zeros or ones it is possible to encounter 0/0 indeterminacies. One way to preclude these is adopt a small cutoff, e.g. we can set $\varepsilon = 10^{-10}$, and make all elements of the parameter matrices $p^{slip}, p^{guess}, p^{transit}$, as well as the initial mastery probability $p^{(0)}$, be in the interval $[\varepsilon, 1 - \varepsilon]$.

Recommendation part

The strategy we use for recommending the next item is a weighted combination of four sub-strategies. Each sub-strategy comes in with an importance weight (the vector of these weights is a governing parameter of the adaptive engine). In particular, if we want to eliminate one sub-strategy, we can simply set its weight to 0. When the user has requested

The pre-requisite relationships among the KCs are naturally visualized as a directed acyclic graph, and is stored as an $N \times N$ matrix w of pre-requisite strengths, w_{ij} representing the strength of the graph edge from KC j to KC i (j is a pre-requisite for i). We define this strength to be on the scale from 0 to 1. If there are no connections, w a zero matrix.

For each KC and for each user, we can define the pre-requisite readiness:

$$r_{ui} = \sum_{j=1}^N w_{ij} \min(0, L_{uj} - L^*). \tag{8}$$

An element r_{ui} has value 0 if the user has sufficiently mastered all KCs pre-requisite for the KC i, and less than 0 if the matery probabilities for some pre-requisites are not yet certain. If the pre-requisite strength w_{ij} is weaker, it enters r_{ui} with a smaller weight, in a sense allowing less certain mastery of less important pre-requisites. If all the pre-requisites are ascertained, $r_{ui} = 0$, otherwise it is less than 0. We can deviate from this slightly and introducing a forgiveness parameter $r^* \geq 0$, we assume that a user u is sufficiently ready for learning an KC i if $r_{ui} + r^* \geq 0$. If $r^* > 0$, it means that we "forgive" some degree of uncertain knowledge of pre-requisites.

To recommend the next question for a student, we subset the matrix k_{qi} to only those questions (matrix rows) that belong to the current homework and that a user u has not seen yet. Thus, we obtain a user specific matrix $k_{qi}^{(u)}$. We define the non-negative user-specific vectors of "remediation", "continuity", "preparedness", and "difficulty appropriatness" (in terms of

difficulty level of the problem $d_q \in [\varepsilon, 1-\varepsilon]$):

$$R_q^{(u)} = \sum_{i=1}^{N} k_{qi}^{(u)} \max\left(0, L^* - L_{ui}\right) \tag{9}$$

$$C_q^{(u)} = \sum_{i=1}^{N} k_{qi}^{(u)} k_{q_{last},i}, \tag{10}$$

$$D_q^{(u)} = -\sum_{i=1}^{N} k_{qi}^{(u)} \left| L - \log \frac{d_q}{1 - d_q} \right| \tag{11}$$

$$P_q^{(u)} = \sum_{i=1}^{N} k_{qi}^{(u)} \min(0, r_{ui} + r^*) \tag{12}$$

where q_{last} is the last item the user saw. To bring them all to comparable scale, we normalize each vector with non-zero range to range:[1]

$$R_q^{(u)} \rightarrow \frac{R_q^{(u)}}{\max(R_q^{(u)}) - \min(R_q^{(u)})}, \quad \text{and similar for others,} \tag{13}$$

where it is understood that the renormalization is performed only if the denominator is not 0.

These expressions are invented, they can be modified in many ways. More competing considerations can be added to the list at will. The idea is that we want to maximize some linear combination of them. As defined here, $P_q^{(u)} \leq 0$, $R_q^{(u)} \geq 0$, $D_q^{(u)} \leq 0$. A larger remediation value for a question means that this question focuses on those KCs for which the user's probability of mastery is currently low. We introduce a vector of importance weights $W = (W_r, W_c, W_d, W_p)$ (defined up to normalization) which measures the relative importance of each: the next item q to serve should be the one which maximizes the combination $W_r R_q^{(u)} + W_c C_q^{(u)} + W_d D_q^{(u)} + W_p P_q^{(u)}$.

The serving stops naturally in two cases: if we exhausted the available questions (the matrix $k^{(u)}$ has no rows) and 2) if $R_q^{(u)} = 0$ for all q, which means that the user has reached the mastery threshold p^* on all KCs relevant for the available questions. Additionally, we may or may not adopt the policy that questions with zero remediation value (i.e. mastery threshold reached for all tagged KCs) should not be served ("stop on mastery").

4. USER EXPERIENCE

We use an LTI tool, multiple instances of which can be plugged in as adaptive modules in different parts of the course. E.g. at the end of each course chapter there is an assessment module. The instance of the LTI tool (referred to as "LTI box") is a frame with navigation controls, inside which an edX Xblock of an item is iframed. Once the user submitted the item (if the item has a submit button, i.e. it is a question), clicking on "Next Item" causes the next item to be iframed, and so on, creating a service sequence of items. The adaptivity means that the length and the order of items in the sequence are

---

[1]This step is debatable. May not do it, or do it differently.
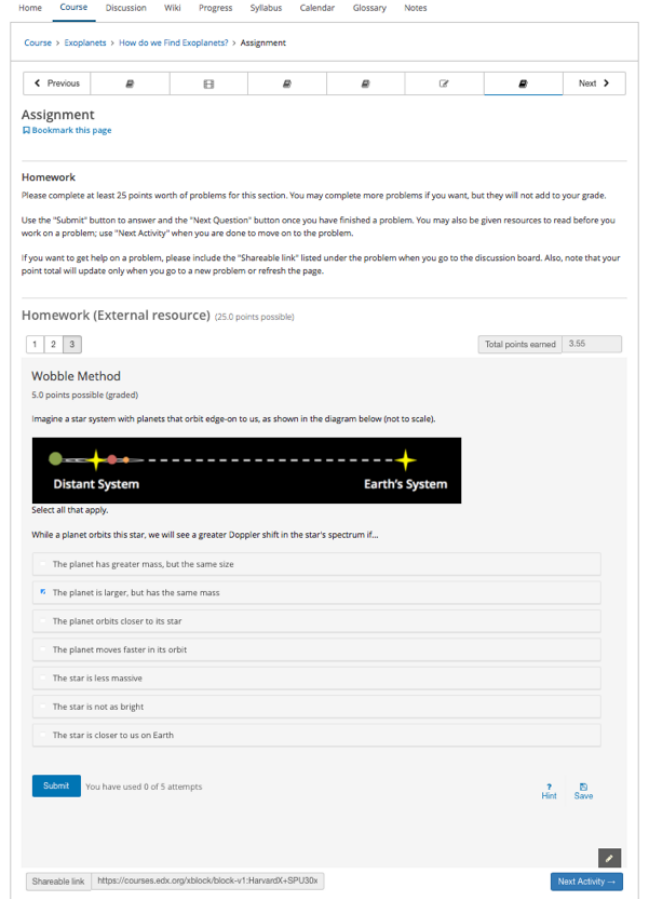
Figure 1. Appearance of an LTI box in a course. At top of the box there is a navigation bar, user can freely move along the instantiated service sequence of items. At the bottom is the button for requesting to continue the sequence and a shareable link to the currently displayed item, so it can be used, for example, in forum discussions.

personalized. The same item cannot be served to a user more than once. The adaptive engine maintains a knowledge profile for each user, i.e. the estimated personal probability of mastery on each knowledge components. The user's knowledge profile is updated after every submit event by that user (including the submits that happen outside the LTI tool). However, the user's also need to be assigned a score for an adaptive assessment module. Such a score should be calculated in a way that looks fair and clear to the users, regardless of the differences in the personalized service sequences. The situations is easier when all questions assigned to one adaptive module are worth equal number of points. For each module, we set the minimum number of questions Q that we would like a user to do. Then, if p is the number of currently earned points in the module and q is the number of distinct items submitted (i.e. multiple submit attempts on the same item count as 1),

$$\text{score} = \begin{cases} \frac{p}{Q} & \text{if } q < Q \text{ and sequence not over} \\ \frac{p}{q} & \text{otherwise} \end{cases} \tag{14}$$

Each adaptive module comes with its own bank of items. I.e. a question may be served in adaptive module 1 but not in

adaptive module 2. But we also allow for a special category of items that can be served in any adaptive module, they just need to be marked as belonging to this type by the course developing team.

Students will be split 50/50 into Group A and Group B. The "stop-on-mastery" policy will not be implemented for either group (we do not refuse to serve a question on whose KCs the user's mastery probability exceeds p^*). In group A, $W_r = 5$ and $W_c = 1$ (i.e. remediation has higher importance than continuity). In group B this is reversed: $W_r = 1$ and $W_c = 5$. The other importance weights are the same for both groups: $W_d = W_p = 1$.

5. OPTIMIZATION OF BKT PARAMETERS

We will rely on a way to optimize our BKT parameters, inspired by the "empirical probabilities" method of [1].

At some point in time, when we decide to run the optimization, suppose that the items submitted by a user u are $\{q_j^{(u)}\}$ ($j = 1,...J^{(u)}$), indexed in chronological order. Let the correctness of answers be $C_j^{(u)}$. We denote $K_{ij}^{(u)}$ this student's knowledge of an KC i just before submitting the item $q_j^{(u)}$. Assuming that there is no forgetting, the knowledge is a non-decreasing function with values 0 and 1, so it is characterized simply by the position of the unit step: for j from 1 to some n_i knowledge is 0 and from $n_i + 1$ onward it is 1. We find which n_i gives the highest accuracy of predicting correctness from knowledge. The generalized number of errors on predicting the outcome based on mastery of a particular knowledge component are:

$$E_i^{(u)}(n) = \left(-\sum_{j=1}^{n} C_j^{(u)} \log o_{q_j,i}^{guess} - \sum_{j=n+1}^{J^{(u)}} (1 - C_j^{(u)}) \log o_{q_j,i}^{slip} \right),$$
(15)

where $n \in [0, J^{(u)}]$ and we adopt the convention that if the lower limit of a sum is greater than the upper limit, the sum is 0. We set the knowledge step where it minimizes the errors:

$$n_i = \mathrm{argmin}(E_i^{(u)}),$$
(16)

and construct the step-function $K_{ij}^{(u)}$ using it n_i. If there are multiple equal minima, and so multiple n_i, we take the average of the corresponding multiple step-functions (because of this, knowledge may now have fractional value). The resulting $K_{ij}^{(u)}$ is our empirical estimate of the knowledge of all KCs by the user u. Repeat the procedure for each user. Note that, if user's problems are irrelevant for an KC, we will find a steadily growing knowledge of that KC. This is not bad, however, since for each KC we will average only over the users who experienced some relevant problems. Namely, we can define the sets of users

$$\mathcal{U}_i = \{\forall u : \sum_{j=1}^{J^{(u)}} k_{q_j^{(u)},i} > \eta\},$$
(17)

$$\mathcal{U}_{qi} = \{\forall u : \sum_{j=1}^{J^{(u)}} k_{q_j^{(u)},i} \mathbf{1}(q_j^{(u)} = q) > \eta\}$$
(18)

where $\eta \geq 0$ is a constant we set as a measure of how much total relevance of a KC is enough for the user to be included into the ensemble for estimating the parameters of that KC. As the simplest choice, $\eta = 0$.

Since the order of items was chronological, $K_{i,1}^{(u')}$ estimates the prior knowledge of concepts by that user. Moreover, we can identify the occasions when slips, guesses or transfers of knowledge occured. Averaging these over the users we get the empirical matrices of prior knowledge, transit, guess and slip probabilities as ratios:

$$P_{u'i}^{(0)} = \frac{\sum_{u \in \mathcal{U}_i} K_{i,1}^{(u)}}{\sum_{u \in \mathcal{U}_i} 1}$$
(19)

(same priors for all users u', i.e. all rows of $P^{(0)}$ are identical)

$$P_{qi}^{trans} = \frac{\sum_{u \in \mathcal{U}_{qi}} \left(\sum_{j=1}^{J^{(u)}-1} (1 - K_{ij}^{(u)}) K_{i,j+1}^{(u)} \mathbf{1}(q_j^{(u)} = q) \right)}{\sum_{u \in \mathcal{U}_{qi}} \left(\sum_{j=1}^{J^{(u)}-1} (1 - K_{ij}^{(u)}) \mathbf{1}(q_j^{(u)} = q) \right)}$$
(20)

$$P_{qi}^{guess} = \frac{\sum_{u \in \mathcal{U}_{qi}} \left(\sum_{j=1}^{J^{(u)}} (1 - K_{ij}^{(u)}) C_j^{(u)} \mathbf{1}(q_j^{(u)} = q) \right)}{\sum_{u \in \mathcal{U}_{qi}} \left(\sum_{j=1}^{J^{(u)}} (1 - K_{ij}^{(u)}) \mathbf{1}(q_j^{(u)} = q) \right)}$$
(21)

$$P_{qi}^{slip} = \frac{\sum_{u \in \mathcal{U}_{qi}} \left(\sum_{j=1}^{J^{(u)}} K_{ij}^{(u)} (1 - C_j^{(u)}) \mathbf{1}(q_j^{(u)} = q) \right)}{\sum_{u \in \mathcal{U}_{qi}} \left(\sum_{j=1}^{J^{(u)}} K_{ij}^{(u)} \mathbf{1}(q_j^{(u)} = q) \right)}$$
(22)

Here again, we adopt the convention that if the lower limit of a sum is greater than the upper limit, the sum is 0 (this happens when $J^{(u)}$ is 0 or 1). The value of the denominator in each of these expressions is a measure of how much student information we have for estimating the probability. In case there is no data, the expression becomes a 0/0 indeterminancy. We should not want to update a probability in this case. Moreover, we impose a threshold M (e.g. 20) and say that we will not update a particular probability if the denominator in the corresponding equation is less than M. This is simple to enforce in practice: add to eqq. (19, 20,21,22) the rule that in each of them the denominator less than or equal to M (this roughly counts the number of terms in the denominator sum) should be replaced by 0. After that, any non-numeric (indeterminate or infinite) elements of the matrices $P^{(0)}$, P^{trans}, P^{guess}, P^{slip} should be replaced by the corresponding elements of the matrices $p^{(0)}$, p^{trans}, p^{guess}, p^{slip}.

We should also watch out for guess and slip probabilities of 0.5 and up. If this happens, we will not update with such values.

Now we can update the BKT parameter matrices with the estimates: $p^{(0)} = P^{(0)}$, $p^{trans} = P^{trans}$, $p^{guess} = P^{guess}$, $p^{slip} = P^{slip}$, and also convert probabilities to odds, i.e. $L_{ui}^{(0)} = \log(p_{ui}^{(0)}/(1 - p_{ui}^{(0)}))$, and the guess, slip, transit odds via (2).

In addition to that, we should also update some elements of the current mastery odds L. Namely, let us call "pristine" those elements of L, which have never been updated with a

non-zero shift via (6). All the pristine elements of should be replaced with the corresponding elements of $L^{(0)}$. This way, the optimized prior KC knowledge values will be used for all users yet to come to the course, but also for the existing users for those knowledge components that they have not yet explored.

6. MODEL PARAMETERS

As described in section 3, the model is governed by the following constants: the mastery threshold L^* (maybe 2.2, corresponding to 0.9 probability), the pre-requisite forgiveness parameter r^* (maybe 0), the vector of relative importances of recommendation sub-strategies W and the regularization cutoff ε (maybe 10^{-10}). In addition, as will be described in section 5, there are threshold for updating BKT parameters: η (maybe 0) and M (maybe 20).

The minimal information we need from the course developer or SME is

1. Item tagging (which item is tagged with which KC or KCs).

2. Item placement (which item should be served in which adaptive module; or belongs to the type servable in any module).

3. Item worth (which item is worth how many points). Note that the grading strategy we have currently demands equal point worth in all items that could be served in the same module. But this might change in the future.

4. Pre-requisite relations among knowledge components (at its simplest, the learning order of KCs, which KC should be learned before which). If this is not provided, the engine will assume that no pre-requisite relations exist and so will not use that as a factor in the recommendation decision.

More information is useful but not strictly required, so can be provided partially or not at all: probabilities of guessing (correct despite not knowing KC), slip (incorrect despite knowing KC) and transfer (learning KC from doing the item) per item per tagged KC. If we have a cold start, we start with cold guesses for all these parameters in the knowledge tracing, until enough user transactions are accumulated to replace these values with the estimates from the data. By default, the cold start parameters are the same across all questions, but if SME provides the information of this type, we can use it.

7. TESTING PREDICTIVE POWER OF KNOWLEDGE TRACING

We used the data from the HarvardX course Super-Earths v3 for testing the predictive power of the knowledge-tracing algorithm. The problems in this course tagged with 66 KCs by SMEs. Unfortunately, the number of problems was disproportionately small for this many KCs. 22 of the KCs had only one problem associated with them, 18 KCs had 2 problems, 13 KCs – 3 problems, 8 KCs – 4 problems, 2 KCs – 5 problems, 3 KCs – 7 problems. So we cannot expect much, but we want to at least see that nothing crazy happens when problems are tagged with more than 1 KC. Restricting ourselves to KCs with more than 3 problems (13 KCs) leaves 60 problems, 3 of which are tagged with 2 KCs, with 2,362 users attempting

them, a total of 56,178 submits. Alternatively, we tagged the items using an automated procedure. In this case the KCs are much better populated with problems (here, too, we dropped some KCs with too few problems, but this was insignificant). This gives us a data set with 133 problems, 2837 users, 19 KCs, and a total of 172,410 submits. This automated tagging did not contain multiple KCs per item.

For both taggings, we performed a 3-fold validation, splitting the list of users randomly into thirds, cyclicly using two of them as the training set and the remaining third as the validation set. To alleviate the random effect of some users having significantly more submits than others, we repeated the 3-fold procedure 10 times. Thus, we are calculated the accuracy measures from a vector of observed submit correctnesses and a vector of predicted submit correctnesses, and the length of each vector is 10 times the number of submits in our data set, i.e. 561,780 for SME-tagged data and 1,724,100 for auto-tagged data.

The algorithm was optimized with parameters $\eta = 0$, $M = 20$. We use five measures of prediction quality: the negative logarithmic likelihoods (total $-LL$, and also separately $-LL_+$ for correct and $-LL_-$ for incorrect answers; high $-LL_+$ is a sign of many false negatives, and high $-LL_-$ is a sign of many false positives), as well as the mean absolute error (MAE) and the root-mean-square error (RMSE):

$$-LL = -\frac{1}{2\log 2}\left(\frac{1}{|N|}\sum_{i\in N} x_i \log p_i + \frac{1}{|N|}\sum_{i\in N}(1-x_i)\log(1-p_i)\right)$$
(23)

$$-LL_+ = -\frac{1}{2\log 2}\left(\frac{1}{|N_+|}\sum_{i\in N_+} \log p_i\right) \qquad (24)$$

$$-LL_- = -\frac{1}{2\log 2}\left(\frac{1}{|N_-|}\sum_{i\in N_-} \log(1-p_i)\right) \qquad (25)$$

$$MAE = \frac{1}{|N|}\sum_{i\in N}|x_i - p_i| \qquad (26)$$

$$RMSE = \sqrt{\frac{1}{|N|}\sum_{i\in N}(x_i - p_i)^2}, \qquad (27)$$

where the notation is: N is the set of question-user records, divided into the set N_+ of correct responses and the set N_- of incorrect responses. The correctness of each question is $x_i = 0, 1$ and p_i is the predicted probability of correct response. Thus, larger value of $-LL_+$ means underestimating correctness (trend to false negatives), and larger value of $-LL_-$ means overestimating correctness (trend to false positives). As defined, all measures would equal 0.5 if prediction were done by a coin toss, and 0 if prediction was perfect ($x_i = p_i$). As a performance benchmark, we take the "mean-trained chance" prediction with p_i being the mean correctness in the training set (same value for all problems).

We see that the knowledge tracing always substantially outperforms prediction by coin-toss (which would have all five

Table 1. SME-tagged data (561,780 predictions).

	knowledge tracing	mean-trained chance
$-LL$	0.420	0.476
$-LL_+$	0.436	0.335
$-LL_-$	0.394	0.715
MAE	0.330	0.467
RMSE	0.447	0.483

Table 2. Auto-tagged data (1,724,100 predictions).

	knowledge tracing	mean-trained chance
$-LL$	0.379	0.366
$-LL_+$	0.319	0.166
$-LL_-$	0.612	1.142
MAE	0.295	0.326
RMSE	0.428	0.404

accuracy measures equal to 0.5). For the SME-tagged data knowledge tracing outperforms the prediction by mean-trained chance. The latter method simply always predicts that the answer will be correct. The only measure which is lower for the chance prediction is $-LL_+$, but that is inconsequential: indeed, since this method always predicts the outcome "correct", it has no false negatives at all. For the auto-tagged data, knowledge tracing outperforms prediction by mean-trained chance only according to MAE. But even if it loses to the mean-trained chance, it is valuable simply because it is adaptive: we could not base a recommendation engine on a prediction method which simply always predicts the same outcome.

If we predict correctness by rounding the probability of correctness (i.e. predicted correctness value is the more likely one of the two), we can see the confusion matrices (Tables 3 and 4).

Table 3. SME-tagged data (561,780 predictions). Accuracy (sum of diagonal terms in the confusion matrix) is 69.4% for knowledge tracing and 62.9% for mean-trained chance.

knowledge tracing	observed $-$	observed $+$
predicted $-$	28.7%	23.3%
predicted $+$	8.5%	39.5%

mean-trained chance	observed $-$	observed $+$
predicted $-$	0%	0%
predicted $+$	37.1%	62.9%

8. AUTOMATED CONTENT TAGGING

Tagging items with KCs is a laborious task for SMEs, so automation, even a partial one, is welcome. If SME tagging is unavailable, we can rely on the automated tagging. If not, automated tagging may serve as the starting point and guidance in the SME's tagging work.

We experimented with the possibility of using a natural language processing algorithm called STM (structural topic model) for this purpose. The textual content of a number of documents is viewed as a mixture of a number of topics, which are defined by the probabilities with which each word

Table 4. Auto-tagged data (1,724,100 predictions). Accuracy (sum of diagonal terms in the confusion matrix) is 74.6% for knowledge tracing and 79.5% for mean-trained chance.

knowledge tracing	observed $-$	observed $+$
predicted $-$	10.4%	17.7%
predicted $+$	10.1%	61.7%

chance	observed $-$	observed $+$
predicted $-$	0%	0%
predicted $+$	20.5%	79.5%

is employed. The number of topics is an input to the STM algorithm, which identifies the topics via an optimization technique. As an output, we obtain the topic properties (how common is each word in a topic), and a non-negative prevalence matrix θ_{qt}, which shows the weight ("prevalence") of a topic t in a document q. The normalization is $\sum_t \theta_{qt} = 1$ for any q. We can adapt this model to the tagging task. The documents are the questions (items) and the topics are the knowledge components. An item q is considered tagged with a KC i, if the value θ_{qi} is high enough: we set a threshold θ^*, and the tagging matrix is $T_{qi} = \mathbf{1}\theta_{qi} > \theta^*$. If $\theta^* = 0.5$, each item will be tagged with at most one KC, apart from the rare possibility of a tie between two KCs, but lower threshold makes multiple-KC tagging possible. To guarantee no untagged items, set $\theta^* \leq \min_q \max_i \theta_{qi}$. If STM is used solely on the basis of the item texts, it reduces to the correlated topic model (CTM). The advantage of STM is that it can include the item metadata as covariates. For instance, the section number, to which an item belongs in the course, may be an important covariate for tagging. Comparing such single-KC tagging with the human tagging of 39 problems from the Super-Earths v3 course, we found Cramer's V coefficient[2] >0.4.

What number of KCs to use may be left for SME to decide, but as a guidance, we can ask for the "clearest tagging", by which we mean that the prevalences of the tagged KCs should be high:

$$N = \operatorname{argmax}\left(\frac{1}{N'} \sum_{q=1}^{N'} \max_i \theta_{qi}^{(N')} \right), \qquad (28)$$

where the upper index in θ indicates that this matrix is the output of the STM with N' topics. This describes the unsupervised automated tagging. The supervised version uses pre-existing tagging of some items and extends it to all. In the simplest version, it will not expand the list of KCs. This can be achieved by training a classifying random forest (with the KCs as the response variable) on the pre-tagged items.

9. CONCLUSION

---

[2]A measure of similarity for categorical variables, between 0 and 1, an analog of correlation for quantitative variables.

REFERENCES

1. William J Hawkins, Neil T Heffernan, and Ryan SJD Baker. 2014. Learning bayesian knowledge tracing parameters with a knowledge heuristic and empirical probabilities. In *International Conference on Intelligent Tutoring Systems*. Springer, 150–155.