Title
Imputing KCs with Representations of Problem Content and Context

ISBN
9781450346351

Authors
Pardos, Zachary A
Dadu, Anant

Publication Date
2017-07-09

DOI
10.1145/3079628.3079689

Peer reviewed

Imputing KCs with Representations of Problem Content and Context

Zachary A. Pardos
UC Berkeley
Berkeley, CA
USA
zp@berkeley.edu

Anant Dadu
IIT (BHU) Varanasi
Varanasi, UP
India
anant.dadu.cse14@iitbhu.ac.in

ABSTRACT

Cognitive task analysis is a laborious process made more onerous in educational platforms where many problems are user created and mostly left without identified knowledge components. Past approaches to this issue of untagged problems have centered around text mining to impute knowledge components (KC). In this work, we advance KC imputation research by modeling both the content (text) of a problem as well as the context (problems around it) using a novel application of skip-gram based representation learning applied to tens of thousands of student response sequences from the ASSISTments 2012 public dataset. We find that there is as much information in the contextual representation as the content representation, with the combination of sources of information leading to a 90% accuracy in predicting the missing skill from a KC model of 198. This work underscores the value of considering problems in context for the KC prediction task and has broad implications for its use with other modeling objectives such as KC model improvement.

KEYWORDS

Representation learning; skip-gram; KC prediction; skill tagging; bag-of-words; ASSISTments

1 INTRODUCTION

Crowdsourcing educational content [1] for tutoring systems and open educational resource sites allow teachers to tailor pedagogy to their students and become an integral part of the instructional design of material that shapes the platform. Crowdsourced content has, however, lacked the same meta-information, often in the form of "tags," that platform curated content has included. These tags help organize the contributed content to be searched for by other users. On the ASSISTments tutoring platform, these tags are comprised of 198 knowledge components (KC), or skill names, enumerated by the platform designers, and are integral in generating reports for teachers, constructing mastery learning problem sets, and organizing content. In ASSISTments, where most content is now user contributed, continued surfacing of methods for imputing these tags is relevant to better integration of this content and propagation to other teachers.

The most straightforward approach to using existing tagged content to predict the tags of untagged content is to assume that regularities in the text of the content might be indicative of their appropriate tags. This intuition is most commonly applied in the form of learning a function between the word distribution of a document and its label. In this paper, we investigate problem context as a new source of information for skill tag prediction. The intuition is that the problem IDs encountered by the students to the left and right of the problem in question, could serve as an abstract signature of that problem's skill, which may generalize to other problems with a similar context. We explore the use of the skip-gram model for representing problems and their relationships to one another in Euclidian space learned from sequences of problem IDs, as encountered by students in ASSISTments.

2 BACKGROUND

Our task can be seen as beginning with a problem by skill association matrix, called a Q-matrix [4], where all the skill associations for a random subset of questions have been hidden. Prior work has used the text of problems to learn their skill labels using bag-of-words with an SVM [2], predicting the skill of the problem (out of 106) with 62% accuracy. Other work by [3] utilized the same technique and predicted the skill (out of 39) with a reported Kapa between .65 and .70. Other paradigms, like non-negative matrix factorization, have been applied towards learning the entire Q-matrix from scratch in testing scenarios where knowledge is not changing [5]. When multiple Q-matrix definitions are present, algorithms have been applied to permute them together to maximize performance prediction and thereby produce a hybrid single Q-matrix [6]. A recurrent neural network was used to learn pre-requisite relationships between skills in ASSISTments [7] by learn representations at the skill level and then querying the model to determine how responses to candidate pre-requisite skills relatively affected candidate post-requisite skill response predictions. This was a skill by skill matrix induction (or attribute to attribute as referred to in Psychometrics). All the methods mentioned apply various models

UMAP '17, July 09-12, 2017, Bratislava, Slovakia

ACM ISBN 978-1-4503-4635-1/17/07…$15.00
http://dx.doi.org/10.1145/3079628.3079689

to extract information from student responses. The method we introduce in this paper, explores the utility of context. A Bayesian Network model has recently been described [8] as capturing application context; however, the use of the word context in that work described the combining of skills with additional, more granular skills or difficulties associated with the problem and arranging these levels of latent nodes in a hierarchy of knowledge components [8]. Our use of the word context captures a more behavioral source of information.

We use a skip-gram model to form a continuous vector representation of each problem. The skip-gram model, and its Continuous Bag-of-Words (CBOW) counterparts, are the state of the art techniques used in computational linguistics, and more commonly known as word2vec [9], an optimized software package released by Google containing both models. These models are in many ways a simpler version of the earlier Neural Net Language Model (NNLM) [10] with the non-linear activations removed and a fixed context window instead of the Markov state of the NNLM. The major advantage of the proposed techniques is that it is very efficient in terms of computational complexity and since the representation is created from linear transformations, it is valid to manipulate with simple vector arithmetic. Skip-gram models are finding expanded use inside and outside of NLP, with applications to machine translation [11], computer vision [12], dependency parsing [13] [14], sentiment analysis [14], biological gene sequencing [15], exploring college course similarities [16], and recommendation systems [17]. Since, at their core, skip-grams were intended to represent words based on the many contexts they appear in and then reason about their relationships given their learned continuous embedding [18] [19]; they were an appropriate choice for our task of representing problems based on their problem contexts and reasoning about their relationships to one another and the skills they represent.

3 PLATFORM & DATA

ASSISTments is an online web platform used primarily for middle and high-school science and math problem solving. The knowledge component model used in the dataset we chose contained 198 unique skills covering mostly middle and high-school math. While it has been 10 years since the original design of its skill model [20], which contained 106 skills [20] (used in [2] [3]), the granularity of its present KC model is still set to match the granularity of the official platform curated questions it asks. Intelligent Tutoring Systems (ITS) like the Cognitive Tutor for Algebra prompts students to answer questions at the step level, which often corresponds to a highly granular knowledge component defined through the time intensive human process of initial cognitive task analysis followed by data-driven refinement. Due to this granular approach, knowledge component models in the Cognitive Tutor for Algebra exceed 2,000 unique knowledge components. ASSISTments has taken a more pragmatic approach to pedagogical design, prompting students with broader level questions than Cognitive Tutors and then breaking those questions down into sub-parts in the form of scaffolding if answered incorrectly or if help is pre-emptively sought. The reasoning for this approach, in part, is to allow teachers to become a primary source of content contribution through a content authoring system which does not require the same level of familiarity with cognitive theory and programming needed for Cognitive Tutors. It is worthy to note that while CTAT (Cognitive Tutor Authoring Tools) have much improved in their usability, they are still not at the ease-of-use level as ASSISTments' authoring tools and teachers do not currently have any means, to our knowledge, of adding and sharing their content in Cognitive Tutors. Of the nearly 80,000 top level problems in ASSISTments (non-scaffolding), 71% have no skill associated with them. ASSISTments does not track students' cognitive mastery per skill, as the Cognitive Tutors do [21], so this skill tagging is not as critical to learning in ASSISTments; however, the lack of a skill tag makes the item more difficult to find among other teachers and also prevents responses to that item from being included in a skill level report available to teachers. The missing skill information for these items also adds considerable noise to the many knowledge tracing based approaches used by researchers in dozens of papers studying these data, since these missing KC items and the learning which occurred during interactions with them are often filtered out.

We conducted our analyses on the ASSISTments 2012-2013 school year dataset[1] and considered only the following five attributes:

- *user id:* student unique anonymous identifier
- *problem id:* unique ID of original problems/questions
- *skill name:* The single[2] skill associated with the problem
- *correctness on first attempt:* The correctness of response given by the student to the problem on his or her first attempt and without first asking for help.
- *base_sequence_id:* The problem set ID containing the problem associated with the student's response. While problems can belong to multiple problem sets, a particular response belongs to a problem answered within the context of a problem set. Problem sets are the level at which teachers assign work to students.

Table 1: ASSISTments 2012-2013 Dataset description before and after filtering out problems with no KC.

	original	after removing items with missing KCs
Responses	6,123,270	2,630,080
Problems	179,999	50,988
Users	46,674	28,834
Problem sets (base sequence IDs)	110,648	102,104

[1] https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-with-affect

[2] ASSISTments' internal KC model allows for several skills to be tagged to the same problem; however, in this dataset, each problem is tagged to at most one skill. In personal communication with the ASSISTments Platform we learned that while the majority of problems are internally tagged with only a single skill, the additional tags of the few multi-tagged problems have been removed in this dataset, leaving only the first skill tag in terms of its arbitrary alpha-numeric order.

Table 1 summarizes the dataset. The average number of unique problems and skills attempted by each user was 86.93 and 12.53, respectively[3]. We also incorporated the problem text of each of the problems in our pre-processed dataset to predict the KC tags associated with them to replicate [2] on this dataset. There were two versions of the problem text used, one with HTML markup stripped out and one with the markup left inside. We included the HTML version in analysis because the tags include image meta information which might be relevant to classification. Out of the 50,348 problems in our pre-processed dataset, 50,339 had problem text information made available by ASSISTments.

Our dataset consists of student responses to problems, however, the problems they engage with is restricted to those contained within problem sets assigned by their teacher. The interface allows them to jump around between existing assignments but it is more common for students to complete a problem set before moving on to the next.

4 METHODOLOGY

Our primary methodology treats the sequence of problem IDs encountered by each student as training instances for our representation learning models. All problem IDs in the dataset are used in the unsupervised representation learning phase. We cross-validate the optimization of the skip-gram parameters and the prediction of the skill labels by problem and base sequence ID by associating learned vector representations with skills using the training set of problems and attempting to predict the skill of the problem representations in the test set. We explore manipulations of the representation for classification using various levels of machine supervision.

4.1 Representation learning with skip-grams

A skip-gram is a simple three-layer neural network with one input layer, one hidden layer, and one output layer (Figure 1). The input is a one-hot representation of the problem ID (dummy variablized) with a one-hot output for each of the specified number of problems in context. The number of problems in context is two times the window size, which is a hyper parameter of the model. The objective of the model is to predict the problems in context given the input problem. Since multiple problems are being predicted, the loss function (categorical cross-entropy) is calculated for each problem in context. The second major hyper parameter is the size of the hidden layer, which ultimately is equivalent to the number of dimensions of the resultant continuous representation since it is the weights associated with the edges stemming from the one-hot position of the problem to all the nodes in the hidden layer that serve as the vector after training. In a skip-gram, the output vector of the hidden layer is: $\mathbf{h} = \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{x}$

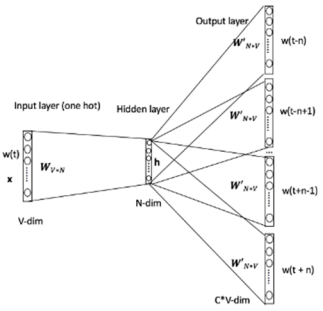


Figure 1. Skip-gram architecture

Where $\boldsymbol{x}$ is the one-hot input of a problem and $\boldsymbol{W}$ is the weight matrix of the connections from the one-hot to each of the nodes in the hidden layer. There is an output for each problem ID, the larger the value, the greater the chances of that problem being in context. Each problem ID, $\boldsymbol{i}$, is defined by: $z_i = \boldsymbol{W}_i'^{T}\mathbf{h}$

The final output is a softmax, which normalizes the outputs into probabilities: $\boldsymbol{y}^i = \frac{e^{\boldsymbol{z}^i}}{\sum_{j \in Problems} e^{\boldsymbol{z}^j}}$

With a loss being the sum of the categorical cross-entropy for every problem in the context window:

$$\mathrm{C} = -\sum_{\substack{j \in Problems \\ in\ context}} \log \boldsymbol{y}^j$$

To train the model, we passed one sequence of problem IDs per student as the input. Each row has all the problems attempted by the student in chronological order. We used two different ways of tokenizing problems. First, we used problem IDs as inputs irrespective of the student's first attempt correctness on them. In the second version, we concatenate the correctness of the response with the problem ID which created two representations for each problem ID. This allowed us to explore if problem and correctness context was important for skill classification.

4.1.1 Distance based classification

After learning the vector representations of problems using the skip-gram model, we created a centroid (skill vector) to represent each skill by averaging together the problem IDs in the training set associated with that skill. For predicting the test set problem IDs, we look to see which skill vector each test set problem vector is closest to. We compare using closeness measures of both Euclidian and cosine distance.

We explored several different metrics to optimize in order to determine the best set of skip-gram hyper parameters:

1. Variance: In this approach, we tried to minimize the average variance across each dimension for each KC.

$$costfunction(x) = argmin\left(\frac{1}{kc}\sum_{i=1}^{i=kc}\frac{1}{d}\sum_{k=1}^{k=d} var(x_{ik})\right)$$

$$var(x_d) = \frac{1}{pc}\sum_{j=1}^{j=pc}\left\|x_{dj} - m_d\right\|^2$$

here x is calculated for different parameters of the skip-gram, kc is the total number of unique knowledge components in the data, d is the total number of dimensions and x_{ik} is the vector of k^{th} dimension of problem vectors belonging to i^{th} knowledge

[3] Our analyses were conducted on a randomly chosen 80% user ids. The remaining 20% will be used for future studies

component, pc is the total number of problems, m_d is the mean of all the problem vectors for d^{th} dimension and x_{dj} is the value of the d^{th} dimension for j^{th} problem vector.

2. Cosine Distance: Minimizing the angular distance of problem vectors of the same skill:

$$costfunction(x) = argmin\left(\frac{1}{kc}\sum_{i=1}^{i=kc} median_{j\in pc_i}\{cosinedistance(x_j, m_i)\}\right)$$

$$cosinedistance(\bar{a}, \bar{b}) = 1 - \frac{\bar{a}.\bar{b}}{\|\bar{a}\|\|\bar{b}\|}$$

3. Euclidean Distance: Minimizes the Euclidian distance between problem vectors of the same skill:

$$costfunction(x) = argmin\left(\frac{1}{kc}\sum_{i=1}^{i=kc} median_{j\in pc_i}\{euclideandistance(x_j, m_i)\}\right)$$

$$euclideandistance(\bar{a},\ \bar{b}) = \|\bar{a} - \bar{b}\|$$

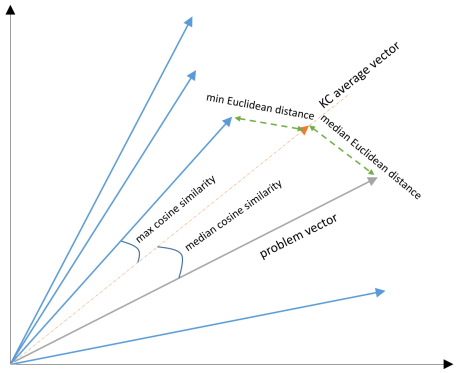


Figure 2: Geometric examples of three optimizations

Figure 2 shows the geometric meaning of our three distance based optimization approaches.

4.1.2 Supervised learning approaches

The distance based approaches were predicated on the assumption that problems of the same skill would cluster together naturally in the learned representation space. If they do not cluster together, the vector representation may nevertheless encode enough information to be classified in a supervised fashion. We use a neural network classifier for the supervised approach, which takes as input the vector representation and has the skill as the label. The neural net was a single hidden layer feed forward network with a 100 node hidden layer and used either a logistic or rectified linear unit function activation (relu). The relu function is defined by $f(x) = \max(0, x)$. The logistic function is defined $f(x) = \frac{1}{1+e^{-x}}$. The loss function used is cross-entropy with a softmax layer as the output layer.

4.2 Bag-of-Words

In this approach, we applied bag-of-words to convert the problem text description to a vector the size of the vocabulary. The problem text was converted into sparse matrix such that each word behaves like a feature. The weights are given to each word for each sample using term frequency and inverse document frequency. It is generally referred to as tf-idf transform. We treated word stemming (true/false) as a hyper parameter of the method. Stop word removal was not an effective filtration based on training set prototyping, so it was not employed in the experiments. Keeping or stripping the text of HTML tags was another hyper parameter. We tested two different classification algorithms for mapping from bag-of-words to label; neural networks and Naive Bayes. There were eight combinations of hyper parameters in total for the bag-of-words models.

4.3 Evaluation

In total, our experiments included two approaches (1) skip-grams and (2) bag-of-words. For skip-grams, hyper parameters (windows size & vector length) were searched using four different methods (a) vector variance minimization, (b) cosine angle minimization, (c) Euclidian distance minimization, and (d) validation set error minimization – this method created a 20% subset of the training set to serve as a validation set and chose the hyper parameters which optimized skill prediction accuracy on this validation set. Two prediction approaches were tried for predicting the skill of the problems in the test set: (1) neural network based classification of the vector to the skill and (2) distance based approach to vector/skill association. A 5-fold cross-validation (CV) was used throughput, with the entire evaluation process repeating within each CV phase. Two different CVs were tried, one where problem IDs were randomly placed in one of 5 CV bins, and the other where base sequence IDs (problem sets) and their problems were randomly placed into one of 5 CV bins. The reason for the base sequence ID CV was the suspicion that copies of the same problem (with different numbers filled in) may show up within a problem set labeled with a different problem ID. This is frequently done in ASSISTments "Skill Builder" problem sets which allow students to keep practicing problems until they answer N correct in a row (often 3 or 5 as set by the teacher). In this case, the bag-of-words classification would be made too easy since some problems with the exact same text would appear in the train and test. This also makes skip-gram classification easier since the prediction could simply use the skill of the problems immediately surrounding the problem. The base sequence ID CV split is meant as a more rigorous test of the generalization ability of the experiments. The evaluation procedure flow diagram can be seen in the Appendix Figure at the end of the paper.

5 EXPERIMENTS AND RESULTS

The results in this section report the cross-validated accuracy with which the algorithms' top prediction of skill matches the skill associated with the problem IDs in the test folds of the CV. The exact same CV assignments were used across all experiments. All bar charts in this section represent the average accuracy of experiments collapsed on the value represented by each bar. The companion table ebfore each chart gives the exact experimental parameters of the top 5 performing models.

5.1 Distance based approach

This approach evaluated how well the problem representations of the skip-gram would cluster together by skill and thereby be

classifiable using skill vector averages. Accuracies ranged from ~15% to ~34% on average in predicting the correct skill of problems out of 198 possible skills. The base sequence ID CV proved to be much more difficult a task, as anticipated, than problem level CV. Using correctness (C) information in the tokenization helped marginally but the distance measure for choosing the nearest skill and the optimization types did not make a difference, as can be seen in Table 2 and Figure 3 (all problem based CV type).

Table 2: Top 5 experiments using distance

Rank	*Optimization*	*Token*	*Acc*
1	Cosine	Correctness	0.3395
2	Euclidian	Correctness	0.3348
3	Variance	Correctness	0.3335
4	Euclidian	no Correctness	0.3313
5	Cosine	no Correctness	0.3252

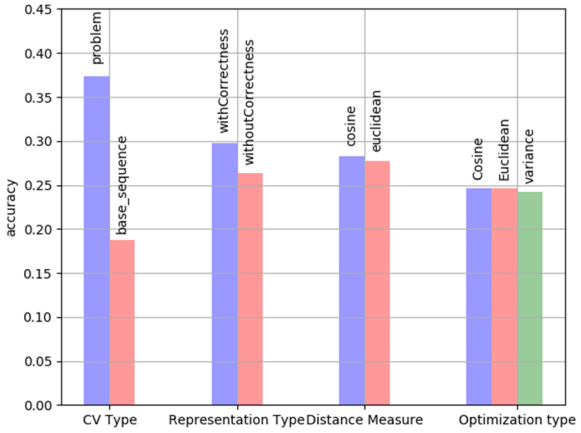


Figure 3: Avg. accuracies from distance experiments

5.2 Distance based (with validation set)

We tried another approach for hyper parameter searching to improve our model. In this approach, the training set was split into 20% sub-test set and 80% sub-train set and the entire prediction methodology was applied on the sub-test dataset as the test set. The parameters with minimum error were selected to build the actual model. Also note that within the validation optimization, we used the same type of prediction method that is set to be applied to the test set for the experiment. As can be seen in Table 3 and Figure 4, the accuracy increased by a significant margin to between ~25% to ~55% using a validation set to choose hyper parameters.

Table 3: Top 5 experiments using distance with validation

Rank	*Distance*	*Token*	*CV*	*Acc*
1	Euclidean	C	P	0.5597
2	Cosine	C	P	0.5490
3	Euclidean	noC	P	0.5000
4	Cosine	noC	P	0.4917
5	Euclidean	C	B	0.2941

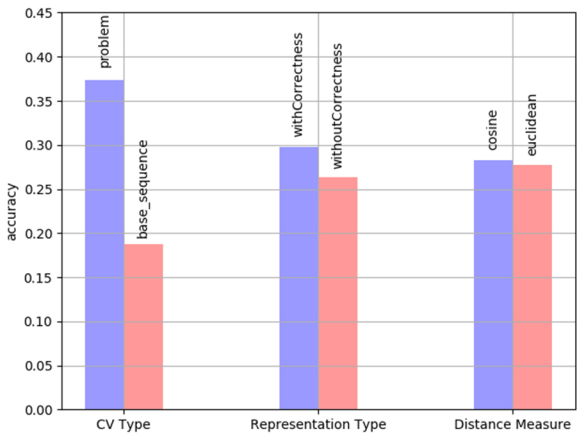


Figure 4: Avg. accuracies from distance experiments using a validation set to optimize hyper parameters

5.3 Supervised learning approach

In these experiments, instead of relying on the continuous vector representations of problems to cluster together by skill, the representations from the training sets are used as inputs to a neural network that learns to classify the skill of the vector representation. Experiments in this section significantly increase again, reaching a new height with individual experiment accuracies between ~50% and ~86%.

Table 4: Top 5 experiments using supervised classification

Rank	*Optimization*	*Token*	*CV*	*Acc*
1	Validation	noC	P	0.8643
2	Validation	C	P	0.8585
3	Variance	noC	P	0.8489
4	Variance	C	P	0.8418
5	Cosine	C	P	0.7086

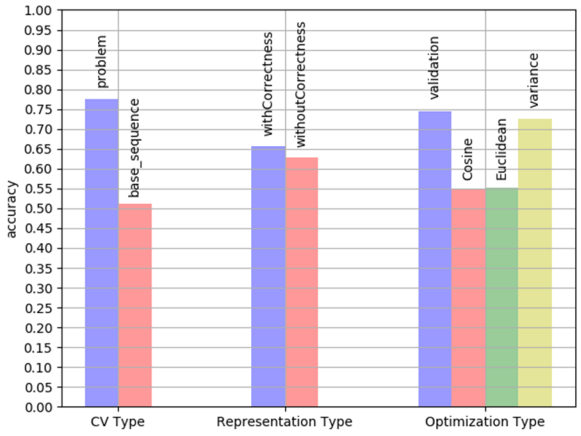


Figure 5: Avg. accuracies from supervised experiments

As seen in Table 4 and Figure 5, the best experiment used a validation set to optimize skip-gram hyper parameters, and in this experiment no correctness information (noC) was added to the problem token.

5.4 Summary and min-count parameter

The best supervised method (86%) beat the best distance based method (56%) by 30 percentage points. These were the accuracies of getting the skill correct on the first prediction but the accuracies improve somewhat if the correct skill can be within the top 5 or 10 predictions(recall @ 5 or 10), as shown in Figure 6. Also shown is the accuracy (y-axis) as the skip-gram parameter of min-count is varied (x-axis). Min-count specifies the minimum number of observations of a token for it to be included in the model. Skip-gram representations quickly lose their integrity when very low frequency words are included. It is therefore common practice to set a minimum count for each word. The tradeoff is that with a higher threshold, fewer words make it in. In the case of ASSISTments, a higher min count meant a higher number of problems wouldn't make it into the analysis. Figure 7 shows what percentage of the responses in the dataset are covered with various settings of min-count. When correctness is concatenated with the problem, min-count has an even more severe effect since a problem is only included in analysis if both the correct and incorrect response concatenated token appears with a frequency above min-count. We chose a min-count of 5 for all experiments in the previous results sections as it covers 99.02% of the data without using correctness and 86.63% with correctness, while keeping a reasonable minimum token frequency.

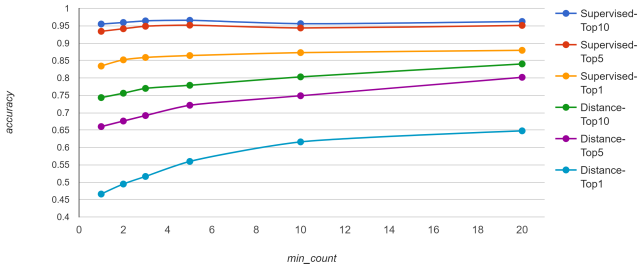


Figure 6: Accuracy variation by min-count for the best supervised learning and distance based models.

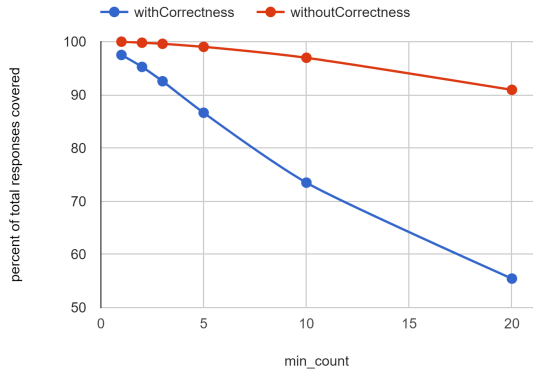


Figure 7: Percent of the dataset covered with various settings of min-count for with and without correctness

5.5 Bag-of-words results

The bag-of-words approach is the only method in this paper which uses the content of the problem (text) to make classifications of the problem's skill. Both a simple 100 node single hidden layer feed-forward neural network (NN) and a Naïve Bayes model were evaluated. The models' accuracies ranged from ~50% to 88% as seen in Figure 8. Neural networks outperformed Naïve Bayes by 25 percent. As with the representation learning approach, problem level CV was easier to predict than base sequence level CV. While filtering out HTML was better on average in these experiments, the best performing models were the ones that kept this markup which often contained information about the problems' images.

Table 5: Top 5 experiments for bag-of-words. The CV type for all experiments was 'problem' level

Rank	*Alg.*	*Stemming*	*Parsing*	*Acc*
1	NN	No ST	HTML	0.8818
2	NN	ST	HTML	0.8810
3	NN	No ST	No HTML	0.8646
4	NN	ST	No HTML	0.8629
5	NB	No ST	No HTML	0.7502

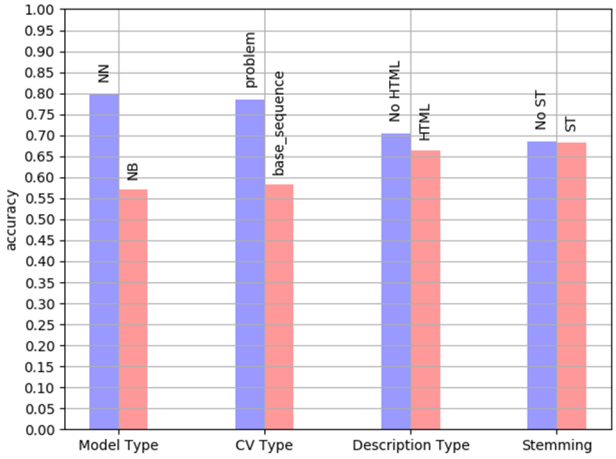


Figure 8: Avg. accuracies from bag-of-words experiments

5.6 Ensemble of content and context models

To improve overall prediction, we combined the two methods by taking the representation vectors from the best skip-gram model and concatenated them with the best bag-of-words vectors for each problem. We chose the best experiment parameters for both the CV types for all the different model approaches. We then trained the supervised neural net on the concatenation of the vectors to learn their skills from the training set problems. We summarize our results for kc tagging in Table 6. All the best models shown use the validation optimization type. The token type is with Correctness concatenation except for the combined and supervised model with the problem CV, which performed best without correctness information. For the bag-of-words model, no word stemming was used and HTML was kept as part of the BOW. When the content and context were combined, the accuracy increased by 2.12% to 90% for the CV split type of problem and a

similar amount for the base sequence CV. The plot in Figure 9 shows the accuracies (recall @ N) for the different models.

Table 6: Top models from the four categories, including the ensemble (combined) category for both base sequence (B) and problem (P) level CV

Rank	*Model*	*Acc (B)*	*Acc (P)*
1	Combined	0.7322	0.9030
2	BOW	0.7260	0.8818
3	Supervised	0.6547	0.8643
4	Distance	0.2941	0.5597

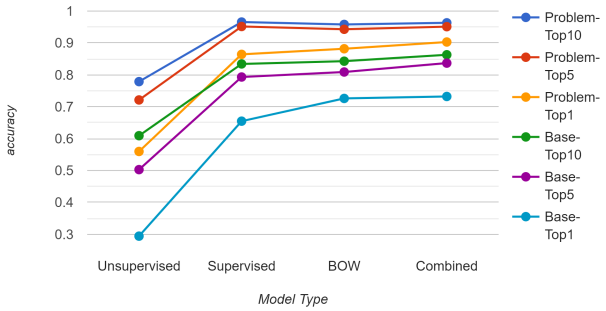


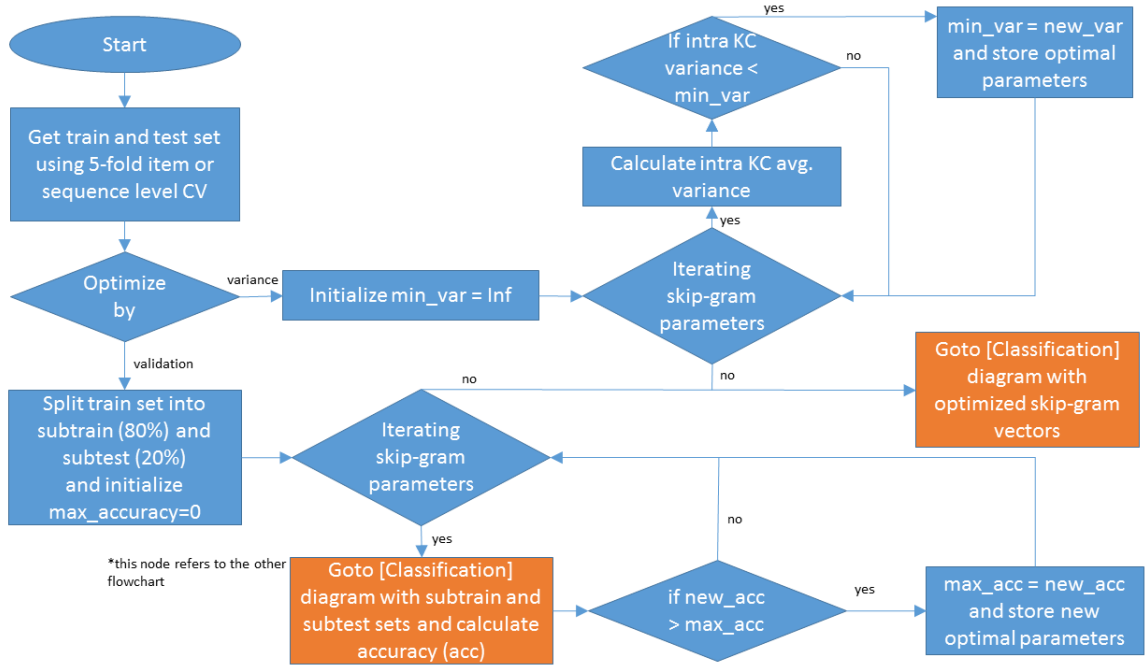
Figure 9: Avg. top N (recall @ N) accuracies for the best models + ensembles from all categories

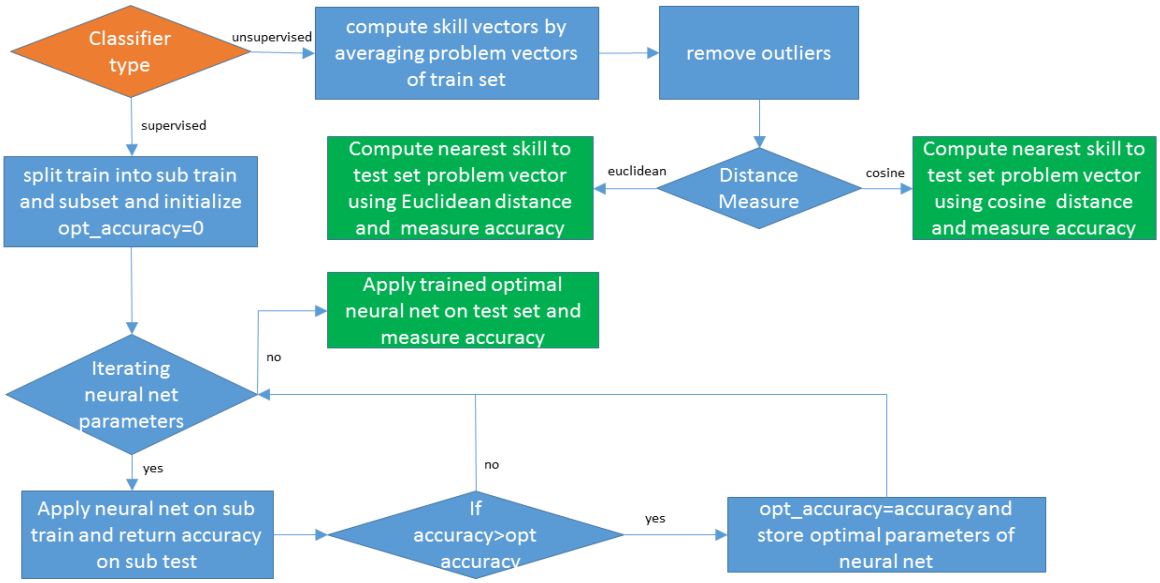
6 CONCLUSION & DISCUSSION

In this paper, we explored the novel predictive value of problem sequence context for imputing the KC of a problem and compared it to the value of using the problem text to classify its KC. We found that the relevant information to the task in the problem context (86% accuracy) was on par with the information from the problem's text (88% accuracy) with the combination of the two producing results which were better still (90%). The skip-gram model clustered problems of the same skill moderately close to one another, allowing for 56% accuracy with a distance based approach; however, adding a non-linear classification step to classify the vector was needed to boost accuracy up to the bag-of-words level. The increase from 25% to 56% accuracy by using a validation set suggests that the skip-gram can overfit with respect to our task and that being mindful of its generalization on a hold-out is prudent. The best performing context models did *not* use correctness information, a relative surprise given the number of KC model refinement techniques that use correctness almost exclusively.

In the case of a newly added problem, the bag-of-words approach has the clear advantage of not needing any response or interaction data to be collected before making a classification. The skip-gram, on the other hand, would require at least 5 students to encounter the new problem and then encounter more problems afterwards to fill a windows size of N. This may occur fairly quickly, but is not guaranteed to. On the other hand, the contextual skip-gram approach generalizes to problems of any type, including problems with only video, images, or interactives that are not clearly amenable to the bag-of-words approach. What this work underscores is that sequence context has a role to play in KC modeling work, including the potential to learn how to improve KC models by analyzing learned problem representations.

Acknowledgements. We would like to thank Seth Adjei for his assistance with the ASSISTments problem text data and description of how the skill tags were created for the public 2012-2013 dataset.





Appendix Figure. Flow diagrams documenting the evaluation process for all experiments in the study. The top diagram depicts the skip-gram hyper parameter search process while the bottom diagram depicts the process for KC classification.

7 REFERENCES

[1] D. Porcello and S. Hsi, "Crowdsourcing and curating online education resources," *Science,* vol. 341, pp. 240--241, 2013.

[2] 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,* pp. 442--459, 1993.

[3] M. KarlovčecMariheida, M. Córdova-Sánchez and Z. A. Pardos, "Knowledge Component Suggestion for Untagged Content in an Intelligent Tutoring System," in *International Conference on Intelligent Tutoring Systems*, pp. 195-200, 2012.

[4] C. P. Rosé, P. Donmez, G. Gweon, A. Knight, B. Junker, W. W. Cohen, K. R. Koedinger and N. T. Heffernan, "Automatic and Semi-Automatic Skill Coding With a View Towards Supporting On-Line Assessment.," in *AIED*, pp. 571-578, 2005.

[5] M. C. Desmarais, "Mapping question items to skills with non-negative matrix factorization," *ACM SIGKDD Explorations Newsletter,* vol. 13, no. 2, pp. 30-36, 2012.

[6] H. Cen, K. Koedinger and B. Junker, "Learning factors analysis–a general method for cognitive model evaluation and improvement," in *International Conference on Intelligent Tutoring Systems*, pp. 164-175, 2006.

[7] C. B. J. H. J. G. S. S. M. G. L. J. &. S.-D. J. Piech, "Deep knowledge tracing," in *Advances in Neural Information Processing Systems (pp. 505-513)*, 2015.

[8] Y. Huang, "Deeper knowledge tracing by modeling skill application context for better personalized learning," in *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pp. 325-328, 2016.

[9] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781,* 2013.

[10] Y. Bengio, R. Ducharme and P. Vincent, "A Neural probabilistic language model," *Journal of Machine Learning Research,* vol. 3, pp. 1137-1155, 2003.

[11] T. Mikolov, Q. V. Le and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168,* 2013.

[12] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean and T. Mikolov, "Devise: A deep visual-semantic embedding model," in *Advances in neural information processing systems*, pp. 2121-2129, 2013.

[13] M. Bansal, K. Gimpel and K. Livescu, "Tailoring Continuous Word Representations for Dependency Parsing," in *ACL (2)*, 2014.

[14] D. Santos, C. Nogueira and M. Gatti, "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts," in *COLING*, 2014.

[15] E. Asgari and M. R. K. Mofrad, "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics," *PloS One, 10(11), e0141287,* 2015.

[16] Z. A. Pardos and A. Nam, "The School of Information and its Relationship to Computer Science at UC Berkeley," in *Proceedings of the iConference*, Wuhan, China, 2017.

[17] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop*, 2016.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111-3119, 2013.

[19] T. Mikolov, W.-t. Yih and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations.," *HLT-NAACL,* vol. 13, pp. 746-751, 2013.

[20] L. Razzaq, N. Heffernan, M. Feng and Z. A. Pardos, *Journal of Technology, Instruction, Cognition, and Learning,* vol. 5, no. 3, pp. 289-304, 2007.

[21] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Modeling and User-Adapted Interaction,* pp. 253-278, 1994.