Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference

# Improving Classification of Natural Language Answers to ITS Questions with Item-Specific Supervised Learning

**Benjamin D. Nye, Mustafa Hajeer,** and **Zhiqiang Cai**
University of Memphis
Institute for Intelligent Systems
bdnye@memphis.edu

**Abstract**

In a natural language intelligent tutoring system, improving assessment of student input is a challenge that typically requires close collaboration between domain experts and NLP experts. This paper proposes a method for building small item-specific classifiers that would allow a domain expert author to improve quality of assessment for student input, through supervised tagging of a small number of good and bad examples. Our approach then generates large sets of item-specific features, which are reduced by algorithms (e.g., Support Vector Machines, Ridge Regression) designed for high-dimensional feature sets. The feasibility of this approach is evaluated using an existing data set collected from an intelligent tutoring system based on AutoTutor. Evaluations of this technique show that it performs about as effectively as iteratively hand-tuned regular expressions for evaluating student input, which is a time-consuming process. This method also outperforms general Latent Semantic Analysis for evaluating the quality of input after approximately 8 tagged examples are available. By 16 tagged examples, it improves the correlation with human raters to over R=0.50, as compared to about R=0.25 for LSA alone. As such, this approach could be used by ITS authors to improve the quality of natural language understanding for their content by classifying a fairly small number of student inputs. Future directions for this work include integrating this approach into an authoring system and exploring directions for decreasing the number of tagged examples that are needed to provide effective classifications.

## Introduction: NLP Authoring for ITS

Natural language processing (NLP) for an intelligent tutoring system (ITS) has distinct challenges when classifying student answers, particularly when making right/wrong judgments (Dzikovska et al. 2009; Graesser et al. 2004). Three major competing design criteria are present: accuracy of classification, content author expertise, and consistent module behavior. Accuracy is important because learners, by definition, have fragile knowledge. An ITS must give accurate feedback to the learner, which requires accurate classification of student input. A second issue is that domain pedagogy experts have the best skill set to create ITS content. However, except in rare cases (i.e., an ITS for NLP), these experts have little to no expertise about improving dialog systems. As such, authors typically need to work iteratively with NLP experts to improve the classifications by the central ITS dialog system or must "dumb down" their content for better NLP accuracy. Finally, for a production-scale ITS, content consistency and versioning is very important. When the dialog system for the ITS is completely centralized, this creates major problems for testing content: to fix one content module, you must make changes to the central NLP system that could (potentially) break existing and rigorously-evaluated content modules.

These competing criteria leave ITS designers with difficult choices. While it is clearly important to make global NLP improvements, any classifier will always have flaws, bugs, and gaps. Since content authors often have little direct recourse to fix these problems, feedback and dialog quality take longer to improve. Ideally, a domain pedagogy expert should be able to "fix" the NLP for specific ITS items that work poorly and misclassify the quality of student answers (e.g., "good" answers are treated as "bad"). Supervised learning is a clear candidate to solve these problems. While a domain pedagogy expert may struggle to improve NLP algorithms or logic, their expertise makes them ideal for flagging good or bad answers, tagging specific misconceptions, and marking other classification judgments about student input.

## Concept: Item-Specific Supervised Learning

Our proposed solution is to complement global NLP algorithms for evaluating student input by adding a locally-trained supervised learning algorithm trained on examples for that specific problem. This approach has a few advantages. First, content authors will focus on improving accuracy for the worst-performing items. This is effectively human-driven active learning: authors will tend to tag more examples for poorly-labeled items, reducing uncertainty. Active learning approaches allow a small number of tagged examples to produce significant improvements (Tong and Koller 2002), which is similarly a goal for this work. Second, this approach is a good fit for online algorithms (or offline algorithms fast enough to run in real-time). Authors can classify examples and see the improvements to the system in real-time. This means that authors can improve the ITS module without waiting for an NLP expert to improve the classifications. Finally, it ensures the consistency of existing ITS modules. Since the supervised learning is used for item-specific improvement, it does not impact any other item. These supervised tags could also be used to update the system's global NLP algorithms, but our approach means that global changes can be deferred until the next major version of the system.

In this paper, we focus on evaluating the potential benefits of this approach to classifying human input to an ITS dialog system. This work follows in the footsteps of a significant body of research on short response evaluation, including a recent SemEval (Semantic Evaluation) Task on Student Response Analysis (Dzikovska et al. 2013). Compared to this prior body of work, a major focus of this analysis is to consider the number of training examples that are needed to improve the accuracy of classifying human input for an item. For authors to rely on this method, benefits must be observable after a fairly small number of tagged examples (e.g., 10-12). A second ideal characteristic for this approach would be that the worst-performing items for the global classifier show significant in accuracy. Assuming that these conditions hold, the proposed methodology should be an effective method for enhancing ITS natural language understanding.

## Method: Classifier Evaluation

Before piloting this approach with real-life ITS authors, it is important to establish that this approach meets these feasibility conditions (i.e., small number of samples and improves low-performing items). To look at this issue, this paper simulates the process of supervised learning by building classifiers on an established data set of human-coded student answers to ITS questions. By looking at which items benefit and the number of tagged examples needed to show improvements, we can estimate the effort required for the authors.

In this paper, we are applying this method to improve AutoTutor answer classifications. AutoTutor is a natural language ITS that has tutored domains such as computer literacy, physics, and research methodology (Graesser et al. 2004; Nye, Graesser, and Hu 2014). AutoTutor converses with users through an expectation-misconception dialog framework: learners are first given an open ended question (typically a "Why" question) and need to explain the main facets of an ideal explanation (expectations). In the process, AutoTutor scaffolds the learner by asking more specific answers such as hints (leading questions) or prompts (fill-in-the-blank) type questions.

In prior research on an ITS for research methodology based on AutoTutor called Operation ARIES: Acquiring Research Investigative and Evaluative Skills (Millis et al. 2011), two raters tagged the quality (good vs. not-good) for a student answers to hint questions (Cai et al. 2011). In that paper, the ratings were used to evaluate the ability of different linear models to predict the average tagger rating, which was on a six-point scale. The most effective models considered Latent Semantic Analysis (LSA) cosine similarity, item-specific regular expressions, and LSA match against the full pool of student answers (Landauer, Foltz, and Laham 1998; Cai et al. 2011). Of these, regular expressions require some iterative improvement and are quite difficult for authors without programming or NLP experience. Our approach tries to harness the pool of student answers more effectively by supporting supervised tags for student answers.

To evaluate our approach, we first compared against earlier benchmarks (Cai et al. 2011) for predicting the 6-point human rater scale. These earlier approaches required expert-designed regular expressions that were specific to each item. For the second analysis, we simplified our classification scheme to a binary good vs. bad judgment. Ridge Regression with feature selection and support vector machines (SVM) were used to classify examples (Cortes and Vapnik 1995; Tikhonov et al. 1995). Ridge regression is similar to standard linear regression, but with a smoothing term that limits overfitting. Feature selection also dropped low-value parameters. SVM maps all features into a higher-dimensional space, where hyperplanes are used to separate classes. SVM scales efficiently, so these local classifiers can process a very large number of features (e.g., a nearly exhaustive set of keywords), with weight given to features that are most predictive. These classifiers were trained using SciKit-Learn Python library (Pedregosa and Varoquaux 2011).

A goal of this work is to support very large feature sets (much larger than the number of tagged examples), which are then pruned to the small number that are useful to discriminate between student answer categories (in this case, good vs. bad). In the present paper, the set of features includes: 1) LSA cosine similarity scores to the "ideal answer" defined by the question author, 2) the logarithm of the number of words in a statement, 3) every word that occurs in at least 5% of good or bad answers (treated as potential keywords), 4) any existing ordered n-grams of length 2 or 3 of every keyword, and 5) the count of negation terms in the answer (e.g., "no," "not," "n't" endings), and 6) a binary flag for an even or odd number of negation terms. The semantic match and logarithm of words were chosen because related research showed that these features were highly-predictive of later test performance (Nye et al. 2014). N-grams were included due to their success in inclusion in prior successful classifiers, such as entries into the SemEval Joint Student Response Analysis Task (Heilman and Madnani 2013; Dzikovska et al. 2013). Negations were included because these play an important role for meaning, and are not handled by traditional LSA approaches.

This approach to keyword detection plays a similar role as regular expression creation, since it infers candidate keywords for good or bad answers. All analyses were run using two different tokenization schemes for keywords. The first method used the Treebank tokenizer and Porter stemmer from the Natural Language Toolkit (Porter 1980; Loper and Bird 2002). The second method used a simpler (and much faster) regular expression word-splitting approach. Also, the log of the answer word count is used rather than the raw number of words, since this has been more predictive of student performance (Nye et al. 2014). The goal of this large number of features is to capture any key difference between different answer categories. Future work is expected to iteratively expand the feature set, while applying classifiers that reduce the number of features in the final model.

## Data Set and Results

The data set used for evaluating our method is described in Cai et al. (2011), which analyzed the assessment of user input by 21 college students into the Operation ARIES tutoring system for research methodology and critical thinking (Millis et al. 2011). While that paper looked at students' long answers and brief answers, in this paper we are focusing on the 1141 long answers, where each answer was a reply to an AutoTutor main question, which tend to be open-ended and require the student generate an explanation (Graesser et al. 2004). Each student answer was matched against one of 32 ideal answers (i.e., question types).

Two independent coders rated the quality of these answers against an associated ideal answer, with "1" as completely wrong and "6" as excellent. These raters tended to score answers fairly highly (average score of 4.7) and had a reasonable level of correlation (Pearson's R=0.69). The rater scores were considered from two perspectives. First, a classifier was trained to predict the average rater score for an input. This outcome data was used to compare performance against the work done by Cai et al.. However, the main goal of this work is to classify the answer type (e.g., identify a "good answer"). As such, a transform was applied to the ratings, where an average rating less than 5 was coded as "not good" and an average $>= 5$ was coded as "good." Two of these ideal answers were excluded from all analyses because they had insufficient examples of bad answers (i.e., ratings on all but one answer averaged over 5). Post-hoc exploratory analyses were performed for different thresholds (4, 4.5, 5.5), but these analyses did not produce any new intuitions compared to the presented results.

| Question | Answer | Avg. Rating | LSA Score |
|---|---|---|---|
| A. Could you define what a theory is? | A1. A theory is an explanation as to why something might happen. | 6 | 0.61 |
| | A2. An explaination of why something happens | 6 | 0.06 |
| B. Do your best to provide a definition for pseudoscience. | B1. Pseudoscience means that the theory proposed cannot be proven false. | 6 | 0.66 |
| | B2. Pseudoscience is hypothesis that are disproved | 1 | 0.77 |

Table 1: Example Questions and Answers

A few examples of questions and associated student answers from this set are noted in Table 1. The question set is primarily focused on expressing distinctions or relationships between concepts, as opposed to procedural questions (e.g., calculations). The LSA match between each answer and the ideal answer is also noted. For example, the ideal answer for Question A was: "A theory is an organized set of principles that explain why something occurs and allow for the creation of new predictions. Scientific theories are tested using strict scientific techniques." This table shows some of the limitations of LSA, which would have two correct accepts (A1 and B1), one incorrect rejection (A2), and one incorrect acceptance (B2). One significant issue is that the students and raters assume a shared common ground for the question: A2 does not state the key word "theory," as it is implied by the question. B2 has a different issue: because LSA does not handle negations, it cannot distinguish between "disproved" and "cannot be disproved." Due to these issues and others, LSA and similar techniques must be complemented with other features.

### Results: Linear Regression Estimator

The results for the local classifier approach worked about as effectively as the results Cai et al., but without the need for hand-tuned regular expressions. For reference, Cai et al. reported Pearson correlations of R=0.28 for LSA and R=0.62 for item-specific regular expressions. Their linear regression using LSA and word count correlated at R=0.41 to the average rating, and the best-fit model with both LSA and regular expressions correlated at R=0.67 (using a split sample train-test design).

Using our local classifier approach explained earlier, we trained and tested Ridge Regression linear estimators using leave-one-out cross-validation. Leave-one-out was chosen due to the relatively small number of tags available for each item (between 17 and 40 samples). These were compared against the inferences provided by a global LSA score. Table 2 shows the results (correlations are Pearson's R). The LSA regression performed comparably to the Cai et al. LSA benchmark, while the trained local classifier performed nearly equivalently to the hand-tuned regular expressions (which had required multiple iterations of authoring).

| Measure | Overall $R$ | $Avg_i(R)$ |
|---|---|---|
| Ridge Regr. (Stemmed) | 0.67 | 0.60 |
| Ridge Regr. (Unstemmed) | 0.65 | 0.58 |
| LSA | 0.42 | 0.40 |
| LSA+Word Count (Cai et al.) | 0.41 | N/A |
| LSA+RegEx (Cai et al.) | 0.67 | N/A |

Table 2: Linear Estimator Results

Since some items ($i$) had more examples than others, the correlation was first calculated for each item-specific estimator and then averaged ($Avg_i(R)$). This shows slightly worse performance for all estimators, but the same trends hold. Finally, stemming provided a small benefit of on the order of 0.02. Across all analyses, this was about the relative level of benefit, so the unstemmed results will not be presented in later analyses. Overall, these results showed that despite using fairly "dumb" features (i.e., keywords), the Ridge Regression was able to match the performance of the hand-tuned regular expressions. Both estimators may be approaching an asymptote due to their source tags as well, since the human taggers only had an inter-rater correlation of R=0.69.

## Results: Classifying Good Answers

Next, we examined the accuracy at classifying "good" answers (rating $>= 5$). This analysis was done using both SVM (linear kernel) and Ridge Regression, but both provided comparable results with no obvious advantage for either. In this analysis, any item with less than 2 negative examples was excluded since the SVM algorithm would not work for those items. This meant that a total of 30 items (ideal answers) and 878 rated answers were analyzed.

A good baseline for this evaluation was difficult to establish, since LSA does not naturally provide a good/bad classification. Two approaches were used to form a baseline benchmark. The first method was a static threshold for all problems. However, Penumatsa, P. et al. (2006) reported that ideal LSA thresholds for classification accuracy tend to vary based on the ideal answer. As such, a second approach trained a second SVM that used only LSA as a feature. Unfortunately, the results for this LSA-trained benchmark were quite bad (Phi R=0.11) and even performed worse than many static thresholds. While a "floating threshold" is probably feasible, it may need significant inertia to prevent overfitting a limited sample. As such, rather than showing this second benchmark, two different LSA cutoff thresholds (0.4 and 0.7) for a "good" answer are reported.

The results for the classifier performance (with stemming) are shown in Table 3. To note, since the outcomes are dichotomous (0 vs 1) the correlations in this table are Phi coefficients, also known as Matthew's correlation coefficients (Cramér 1999; Matthews 1975). The average accuracy across items is also shown ($Avg_i(Acc)$). As with the prior analysis, adding localized features and supervised learning to the LSA classifier significantly improves the correlation to the tagged output and increases the overall accuracy. By comparison, the LSA threshold is very conservative and has poor accuracy when set at common levels of about 0.6 to 0.7. Reducing the threshold to 0.4 improves accuracy, but is likely to make the tutoring system provide positive feedback to some entirely incorrect answers. As such, this "improvement" in accuracy comes with significant practical implications. Overall, the locally-trained SVM is much more accurate and has a correlation with the human raters that is nearly twice as strong as LSA alone.

| Measure | Overall $R$ | $Avg_i(R)$ | $Avg_i(Acc)$ |
|---|---|---|---|
| SVM Classifier | 0.58 | 0.52 | 81% |
| LSA | 0.25 | 0.26 | 62% ($>0.4$)<br>37% ($>0.7$) |

Table 3: "Good Answer" Classifier Results

It is also particularly important to improve classification of items that are classified poorly by LSA alone. If this method only improves the well-classified examples, it would not be very useful for authors. In this analysis, a badly-classified item is one where the LSA threshold accuracy is worse than 70%. For those items (18 out of a total of 30), the average LSA correlation was still $Avg_i(R)$=0.25 but $Avg_i(Acc)$=51%. On the same same items, the SVM classifier had $Avg_i(R)$=0.53 and $Avg_i(Acc)$=78%. So then, this method does improve poorly-performing items, though these items still tend to perform slightly worse than other items. As such, this method offers a reasonable approach for an author trying to improve system's assessment of a particular problem that can be expected to correct over half of the incorrect classifications.

## Results: Benefit Per Supervised Tag

While this classifier improves accuracy and correlation with test classifications, it is also important to look at the number of tags needed to show a benefit. As noted, each item currently has 17 to 40 tagged answers. If it takes 30 tags to show noticeable improvement, authors will get frustrated before reaching that point. To look at this issue, we analyzed the average benefit for each additional supervised tag for an item.

Using a Monte Carlo method, 100 sets of tagged answers were drawn for each item for every number of tagged examples ranging from 6 to 26. Since leave-one-out validation was used, this corresponds to training sets ranging in size from 5 samples to 25 samples (per item), where each training set has 100 randomly-selected populations. This resampling-based approach should damp noise from the random selections. Also, the order of sample selection was randomized, to reduce potential anchoring effects from the human raters (e.g., rating later answers to the same question based on earlier ones). This methodology required at least 26 samples per item, limiting the analysis to the 23 items with at least that number (out of 32). Each item answer set was randomly chosen from that item's answers, using stratified sampling balanced the number of positive and negative examples. If insufficient positive or negative examples existed, samples from the remaining class were drawn as needed.

Figure 1 shows the average quality (Phi R) of the SVM classifier as a function of the number of samples, as compared to an LSA classifier with a static threshold of 0.4. Since the LSA classifier does not learn, any variation is due to random noise or unbalanced sample counts (i.e., insufficient negative examples for certain items). The same analysis was done with a Ridge Regression, which tended to perform slightly worse than SVM initially but followed the same general pattern. While the local classifiers correlate worse than LSA alone for very small sample sizes, they perform comparably by about 7 tagged examples and almost twice as well after 12 tagged examples. The majority of gains occur by 16 samples and plateau at about 23 samples for this data set and features. As shown in Figure 2, this improvement trend is also evident for the average accuracy. Unlike the overall correlation, accuracy is always higher because LSA does not use an optimal set of thresholds.

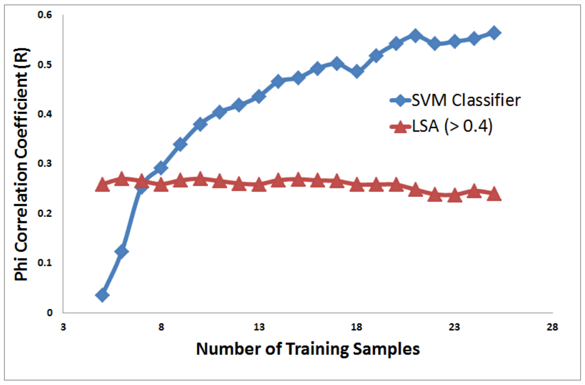

Figure 1: Classifier Quality (R) as a Function of # Samples

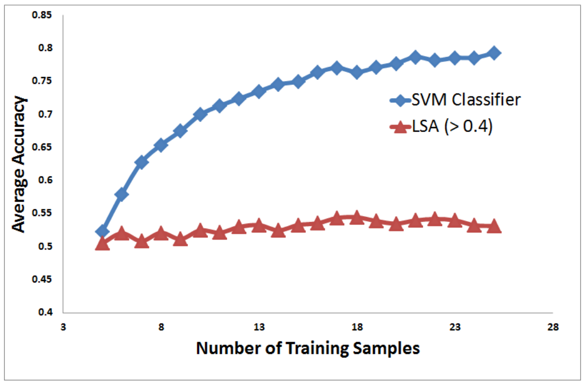

Figure 2: Classifier Accuracy as a Function of # Samples

## Discussion and Conclusions

Overall, this approach to analyzing student input to an ITS seems viable: it offers significant improvements for assessing student input, without requiring a large number of tagged examples. The performance of this model indicates that a hybrid approach is probably ideal, where an author needs to tag a fixed number of examples (e.g., 10) before the local classifier replaces the default LSA threshold. Future work might be able to reduce the number of samples required to reliably improve classifications, such as by using active learning techniques to suggest examples to tag, by adding additional features, or by enhancing local classifiers with a more advanced global model than LSA alone. Since this work is currently using Support Vector Machines, active learning is a potential next-step that could be used to suggest informative (i.e., close to an SVM margin) answers that a human author could tag (Tong and Koller 2002).

While this first exploration was promising, the method must be applied to more diverse sample sets. Due to the relatively small number of students and raters, there is a risk for overfitting regularities in the students' word choices or other imbalances in the data set. One direction for follow-up research is to study these issues, as well as the potential for active learning to mitigate such effects.

Additional features may be needed for the algorithm generalize to other types of questions. The current data set on experimental methods contained fairly distinctive terms and had limited logical entailment. By comparison, sets such as those used in the SEMEVAL student response analysis task involved questions about electrical circuits and other domains that might be less tractable for the evaluated feature set (Dzikovska et al. 2013). Possible features to add include explicitly-authored features or tags (e.g., regular expressions), feature clustering, and specialized feature detectors for domain-specific relationships (e.g., math formulas). For example, the regular expressions applied in prior work might be used as extra features for this model (Cai et al. 2011).

Another possible direction would be to cluster detected keyword features. For example, a brief look at the specific features for given classifiers tended to show multiple synonyms (or even misspellings) of similar concepts. Syntactic clustering (e.g., edit distances) and localized semantic clustering (e.g., words that serve a similar function for an item's answer) might reduce features and increase accuracy slightly. These types of features, among others, have been included in effective short-answer assessment ratings such as the ETS2 entry to SEMEVAL (Heilman and Madnani 2013) and the c-rater system (Sukkarieh and Stoyanchev 2009). Since science and mathematics are common ITS domains, specialized detectors related to formulas and other domain-specific features could also be valuable.

Finally, work needs to examine how these tagged examples can be used to improve the global classification algorithms for an ITS. If taken to scale, this approach would quickly build a corpus of tagged examples for supervised learning. However, each tagged example would be specific to a given ITS question. Research is needed to identify the best ways to harness these item-specific examples for more general supervised learning that improves unsupervised assessment of student inputs to new questions in an ITS.

## Acknowledgments

This research was supported by the Office of Naval Research STEM Grand Challenge (N00014-12-C-0643). The ARIES data set was collected with the support of the Institute of Education Sciences (R305B070349). The views expressed in this paper are those of the authors alone.

## References

Cai, Z.; Graesser, A.; Forsyth, C.; Burkett, C.; Millis, K.; Wallace, P.; Halpern, D.; and Butler, H. 2011. Trialog in ARIES: User input assessment in an intelligent tutoring system. In Chen, W., and Li, S., eds., *Proceedings of the 3rd IEEE International Conference on Intelligent Computing and Intelligent Systems*, 429–433. Guangzhou: IEEE Press.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Cramér, H. 1999. *Mathematical methods of statistics*, volume 9. Princeton University Press.

Dzikovska, M. O.; Callaway, C. B.; Farrow, E.; Moore, J. D.; Steinhauser, N.; and Campbell, G. 2009. Dealing with interpretation errors in tutorial dialogue. In *Proceedings of* *the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '09, 38–45. Stroudsburg, PA, USA: Association for Computational Linguistics.

Dzikovska, M. O.; Nielsen, R. D.; Brew, C.; Leacock, C.; Giampiccolo, D.; Bentivogli, L.; Clark, P.; Dagan, I.; and Dang, H. T. 2013. SemEval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In *Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics*, 263–274. Stroudsburg, PA: Association for Computational Linguistics.

Graesser, A. C.; Lu, S.; Jackson, G. T.; Mitchell, H. H.; Ventura, M.; Olney, A.; and Louwerse, M. M. 2004. AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, and Computers* 36(2):180–192.

Heilman, M., and Madnani, N. 2013. ETS: domain adaptation and stacking for short answer scoring. In *Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics*, 275–279. Stroudsburg, PA: Association for Computational Linguistics.

Landauer, T. K.; Foltz, P. W.; and Laham, D. 1998. An introduction to latent semantic analysis. *Discourse Processes* 25(2-3):259–284.

Loper, E., and Bird, S. 2002. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 63–70. Association for Computational Linguistics.

Matthews, B. W. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta -Protein Structure* 405(2):442–451.

Millis, K.; Forsyth, C.; Butler, H. A.; Wallace, P.; Graesser, A. C.; and Halpern, D. F. 2011. Operation ARIES! a serious game for teaching scientific inquiry. In Ma, M.; Oikonomou, A.; and Lakhmi, J., eds., *Serious Games and Edutainment Applications*. London, UK: Springer. 169–195.

Nye, B. D.; Hajeer, M.; Forsyth, C.; Samei, B.; Hu, X.; and Millis, K. 2014. Exploring real-time student models based on natural-language tutoring sessions: A look at the relative importance of predictors. In *Educational Data Mining (EDM) 2014*. 253–256.

Nye, B. D.; Graesser, A. C.; and Hu, X. 2014. AutoTutor and Family: A review of 17 years of science and math tutoring. *International Journal of Artificial Intelligence in Education* 24(4):427–469.

Pedregosa, F., and Varoquaux, G. 2011. SciKit-learn: Machine learning in Python. *The Journal of Machine Learning* 12:2825–2830.

Penumatsa, P.; Ventura, M.; Graesser, A. C.; Louwerse, M.; Hu, X.; Cai, Z.; and Franceschetti, D. R. 2006. The right threshold value: What is the right threshold of cosine measure when using latent semantic analysis for evaluating student answers? *International Journal on Artificial Intelligence Tools* 15(05):767–777.

Porter, M. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.

Sukkarieh, J., and Stoyanchev, S. 2009. Automating model building in c-rater. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, 61–69.

Tikhonov, A. N.; Goncharsky, A. V.; Stepanov, V. V.; and Yagola, A. G. 1995. Regularization methods. In *Numerical Methods for the Solution of Ill-Posed Problems*, 7—-63.

Tong, S., and Koller, D. 2002. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research* 2:45–66.