User Modeling and User-Adapted Interaction (2020) 30:51–80
https://doi.org/10.1007/s11257-019-09251-6

Going deeper: Automatic short-answer grading by combining student and question models

Yuan Zhang[1] · **Chen Lin**[1] · **Min Chi**[1]

Received: 5 September 2018 / Accepted in revised form: 16 November 2019 / Published online: 1 January 2020

Abstract

As various educational technologies have rapidly become more powerful and more prevalent, especially from the 2010s onward, the demand of automated grading natural language responses has become a major area of research. In this work, we leverage the classic student and domain/question models that are widely used in the field of intelligent tutoring systems to the task of automatic short-answer grading (ASAG). ASAG is the process of applying natural language processing techniques to assess student-authored short answers, and conventional ASAG systems often mainly focus upon student *answers*, referred as *answer-based*. In recent years, various deep learning models have gained great popularity in a wide range of domains. While classic machine learning methods have been widely employed to ASAG, as far as we know, deep learning models have not been applied to it probably because the lexical features from short answers provide limited information. In this work, we explore the effectiveness of a deep learning model, deep belief networks (DBN), to the task of ASAG. Overall, our results on a real-world corpus demonstrate that 1) leveraging student and question models to the conventional answer-based approach can greatly enhance the performance of ASAG, and 2) deep learning models such as DBN can be productively applied to the task of ASAG.

Keywords Automatic short-answer grading · Machine learning · Deep belief network

✉ Yuan Zhang
yzhang93@ncsu.edu

Chen Lin
clin12@ncsu.edu

Min Chi
mchi@ncsu.edu

[1] Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA

1 Introduction

Providing students with informative feedback in assessment settings (e.g., assignments, quiz and examinations) is one of the cornerstones of education. Many real-world assessment questions require students to author free text answers (e.g., fill-in-the-blank and essay questions). While such questions can represent complex learning goals more effectively than multiple choice questions (Anderson and Biddle 1975; Karpicke and Roediger 2008), evaluating free text answers is inherently challenging. The research to date can be categorized into one of two categories: *essay grading*, which typically focuses on the writing style, grammar and coherence of the text (Higgins et al. 2004; Pérez 2004); and *short-answer grading*, (Leacock and Chodorow 2003; Pulman and Sukkarieh 2005; Mohler and Mihalcea 2009), which focuses on the semantic content of the answer rather than general style (Burrows et al. 2015).

Our goal in this work is to investigate effective approaches to automatically grade student short answers, formally termed as automatic short-answer grading (ASAG). More specifically, we adopt the definition of *short answer* used by Burrows et al. (2015): 1) the answers are in natural language; 2) the length of answers ranges from one phrase to one paragraph (essays, by contrast, are much longer ranging from two paragraphs to several pages); 3) the answers require students to utilize external knowledge that is not provided by the question itself, rather than selecting or stating the correct answer(s) in a range of choices; 4) the focus is on the content of the answers rather than the grammar and/or writing style; and 5) the answers are objectively graded (i.e., correct vs. incorrect). Our approach focuses on two aspects which are motivated by prior research: exploring different *state feature spaces* and investigating different *machine learning* methods.

State feature space prior research on ASAG has primarily adopted *answer-based* approaches where researchers rely on various natural language processing (NLP) techniques to analyze the individual student answers. Some examples of complex NLP techniques are syntactic analyzers (Manning and Schütze 1999), rhetorical parsers (Marcu 2000) and semantic analyzers (Mohler and Mihalcea 2009). In the literature, these are sometimes referred to as 'deep' NLP approaches. Early versions of the C-rater (Leacock and Chodorow 2003) and PS-ME (Mason and Grove-Stephensen 2002) systems, for example, were supported by syntactic parsers, discourse parsers and grammatical structure analyzers. Nevertheless, it is often difficult to take full advantage of these deep NLP approaches given the nature of the ASAG task; students' answers are often too short to provide sufficient lexical features for analysis (Pulman and Sukkarieh 2005; Pérez-Marín et al. 2009). Thus, later answer-based approaches often focus on utilizing shallow NLP techniques with machine learning to compare individual student answers to one or more correct exemplars (Mohler and Mihalcea 2009; Graesser et al. 1999; Litman and Silliman 2004; Jordan et al. 2006; VanLehn et al. 2007; Dzikovska et al. 2010; Carterette et al. 2008; Luaces et al. 2017; Rodrigues and Oliveira 2014). Mohler and Mihalcea, for example, utilized measures of text similarity to compare candidate answers to correct answer(s) of the human experts (Mohler and Mihalcea 2009). Rodrigues and Oliveira (2014) matched student answers with referred correct answers by cosine similarity after preprocessing. However, the performance of these answer-based approaches depends upon the availability of a large corpus, the

Table 1 Examples of two tutor–student dialogues: t_{11} and t_{12} form Dialogue 1 and t_{21} and t_{22} form Dialogue 2. Each dialogue consists of a tutor question, a student response and the correct answer

Dialogue 1	
t_{11}	*T* Why are there no potential energies involved in this problem?
t_{12}	*S* **There is no second object that is massive and can have gravitational energy. (Correct)**
	Referred Correct Because the rock is the only object in the system, there are no potential energies involved
…	
Dialogue 2	
t_{21}	*T* Explain why the Earth should be included in the system.
t_{22}	*S* **BrickRed The gravitational force acting on the truck puts it into motion. Therefore we should include the Earth in the system. (Incorrect)**
	Referred Correct The gravitational force on the truck is not perpendicular to the trucks motion. We should include the Earth in the system

incorporation of specific knowledge to port across different domains and/or the manually crafted patterns (Pulman and Sukkarieh 2005), all of which may be unavailable or often labor-intensive. Peer assessment (Luaces et al. 2015, 2017; Raman and Joachims 2014, 2015) whereby students grade the answers of other students is a potential effective solution, but bias and inexperience of students as graders are currently the major concerns for peer assessment.

On the other hand, in addition to student answers, other information such as *question* characteristics and *students'* knowledge levels can potentially be used to improve the performance of ASAG. Table 1 shows two sub-dialogue examples extracted from a student–system interaction log in our training corpus. In Dialogue 1 (timestamps: t_{11} and t_{12}), the student answer is *correct* but looks very different from the referred correct answer generated by human experts. Thus, directly applying simple yet popular NLP approaches such as a bag-of-words (BoW) model or keyword matching may fail to classify it as correct. In Dialogue 2 (timestamps: t_{21} and t_{22}), by contrast, word-to-word mapping between the student's answer and the referred correct answer is pretty high, much higher than that in Dialogue 1, but the student answer is *incorrect*. Thus, the simple BoW may misclassify the answer as correct. In both cases, however, we can improve the models' classification performance by augmenting them with the question and student models. More specifically, the answer in Dialogue 1 would be more likely to be identified as correct if (1) the question model suggests that the question involves certain domain principles, (2) the student model shows that the student's knowledge of those principles is high, and (3) the answer model comprehensively considers interchangeable terms from multiple sources. In the case of Dialogue 2, the similar arguments would hold.

There is a large body of prior research on methods to extract content information from questions and techniques to model students' knowledge levels. For example, Q-matrix (Barnes 2005) can be used to represent the relationship between *domain concepts* and *questions*, while Bayesian knowledge tracking (BKT) (Corbett and Anderson

1994; Lin and Chi 2016; Lin et al. 2016) can be used to model students' knowledge levels. This work investigates the impacts of combining information from questions and from students' knowledge levels with conventional answer-based approaches. We first define three primary models: the answer model (*Ans*), the question model (*Ques*), and the student model (*Stu*). *Ans* extracts various textual features from student answers and the expert's referred correct answer(s). *Ques* extracts information from the current question based on the domain model and from related questions that share important domain concepts and characteristics. *Stu* dynamically updates a student's temporal knowledge levels over the course of his/her training. In order to find an optimal feature space for ASAG, we constructed nine feature spaces based on the three primary models used individually or in composite form.

Machine learning (ML) approaches in much of the prior ASAG work, the researchers began by applying standard computational and linguistic tools to extract features such as word vectors from the training corpus (student answers) and then used them directly as features for various ML and data mining algorithms (Basu et al. 2013; Madnani 2013). Unlike the deep NLP approaches mentioned above (Pulman and Sukkarieh 2005; Dzikovska et al. 2013), these extracted features are often referred to as 'shallow NLP features'. That is, instead of directly applying deep NLP techniques to extract sophisticated text or conceptual similarity metrics which are then used to estimate answer correctness, the ML-based approaches (e.g., Basu et al. 2013; Madnani 2013) extract shallow language features which are then fed into learning models such as naïve Bayes (NB), decision tree (DT) (Pulman and Sukkarieh 2005), logistic regression (LR) (Madnani 2013) and support vector machines (SVM) (Hou et al. 2010) to estimate predictive models.

In recent years, various deep learning models have gained great popularity in a wide range of domains including education and health care (Lin et al. 2018; Kim and Chi 2018; Lin and Chi 2017; Zhang et al. 2017, 2019). Deep belief network (DBN) (Bengio et al. 2007), one of the most effective deep learning models, has been successfully implemented in a wide range of real-world tasks (Mohamed et al. 2009; Huang et al. 2012). DBN supports the automatic extraction of representative features via unsupervised pre-training, and it can be used to learn latent complex relationships among features. Therefore, it has the ability to exploit a more nuanced and discriminate feature space. In this work, we investigate our feature spaces through a series of comprehensive comparisons between DBN and five classic classifiers: NB, DT, LR, SVM and artificial neural networks (ANNs). Given the complex latent connections among the features in the *Ans*, *Ques* and *Stu* models, we hypothesize that DBN will further facilitate the task of ASAG.

In brief, in order to improve upon existing answer-based models, we investigate the impact of various state feature spaces based on the *Ans*, *Ques* and *Stu* models and various ML approaches including DBN on the task of ASAG using a real-world dataset. Our major contributions to the task of ASAG can be summarized as follows:

– So far as we know, this is the first study expanding the feature space of the conventional *Ans* model with *Ques* and *Stu* models and exploring feature space in both combined and composite manner. Our results showed that composing the three models significantly outperformed the conventional Ans-based approaches.

– So far as we know, this is the first study exploring the applicability and effectiveness of DBN for the task of ASAG. Our results showed that DBN indeed outperformed a series of classic ML approaches.

This paper is organized as follows: Section 2 presents an overview of related work in the field of ASAG. Section 3 presents our state feature spaces and briefly describes the applied ML approaches. Section 5 describes the training data and our experimental setup. Finally, Sects. 5 and 6 present our results and conclusions.

2 Related work

While there are numerous educational benefits for students to generate natural language responses, it is often challenging to grade them (Anderson and Biddle 1975; Karpicke and Roediger 2008). A number of approaches have been proposed for automatically grading such responses. For example, based on whether student-authored natural language answers are short answers or essays, ETS developed C-rater (Leacock and Chodorow 2003) for short-answer grading and the E-rater system (Attali and Burstein 2006) for essay grading, respectively. While they are related, short-answer grading differs from essay grading in at least three aspects: (1) the length of short answers varies from one phrase to one paragraph, while the length of essays is much longer, from two paragraphs to several pages, (2) area of interest for short-answer grading is more on the content, while for essays grading, it is more on writing style, grammar etc., and (3) evaluating short answers is often objective, correct or incorrect, while evaluating essays is often more subjective. Our work here focused on short answers.

Prior research in ASAG can be roughly divided into four phases based on the dominant techniques. In the order of appearance, they are: (1) the deep NLP phase; (2) the information extraction phase; (3) the shallow NLP with statistical analysis phase; and finally (4) the machine learning (ML) phase. These phases may overlap in time and techniques.

The first phase involves the applications of deep and complex NLP techniques to extract syntactic and semantic representations of student answers. For example, Manning and Schütze (1999) studied constituents and syntactic dependencies among texts using syntactic analyzers. Burstein et al. (2001) investigated on analyzing the discourse structures of texts through rhetorical parsers. C-rater Leacock and Chodorow (2003) considered that student answers consist of multiple concepts and thus it matched the concepts of student answers with those of teacher's answers through syntactic variation, morphological variation, spelling correction and so on. Overall, the methods in this phase are often limited by the underlying challenges of natural language understanding (Pulman and Sukkarieh 2005; Pérez-Marín et al. 2009).

In the second information extraction phase, pattern matching techniques, such as regular expressions or parse trees, are widely used. Usually, the system would use both student and correct answers as input. For example, AutoMark (Mitchell et al. 2002) utilized parse trees to obtain patterns of student and correct answers; they showed that the overall performance of the system was improved by incorporating a 'moderated' mode for human experts to revise the automatic grading. eMax (Sima et al. 2009) first

generated hand-crafted patterns for each question and then used them to grade student answers through pattern matching; they showed that their approach outperformed the baseline K-nearest neighbor using bag-of-words features. Pulman and Sukkarieh (2005) combined manually crafted patterns with the hidden Markov model part of speech (HMM POS) to tag student answers, and the resulted patterns were compared against the patterns of the correct answer. Given that it is often labor-intensive to manually create patterns for every question, they further explored using naïve Bayes and decision tree to automatically extract patterns; however, they showed that using machine-induced patterns was not as effective as using expert hand-crafted patterns. Hasanah et al. (2016) presented a detailed review of ASAG research using information extraction techniques. In general, research in this phase heavily depends on human expert knowledge and thus it may not scale well. Moreover, it may fail in domains that there are many correct patterns or it is hard to define or to represent them.

The third phase comes into the realm of statistical analysis leveraging a large corpus. Magnini et al. (2005) improved the effectiveness of the BiLingual Evaluation Understudy (BLEU) by integrating the initial system with latent semantic analysis (LSA). AutoTutor Graesser et al. (2007) adopted LSA as its primary representation of world knowledge. Mohler and Mihalcea (2009) investigated two major categories of text similarity measures, the knowledge-based measures using a language knowledge base such as WordNet, vs. corpus-based measures using a large domain-sensitive corpus; results showed that the best knowledge-based measures can perform as effective as the best corpus-based measures. Finally, in Klein et al.'s approach, they first manually graded a subset of student answers and from which it successfully built a LSA-based system to automatically grade the remaining ones (Klein et al. 2011). In short, the success of approaches in this phase often heavily depends on the availability of a large training corpus.

Finally, researchers have exploited various ML approaches to the field of ASAG. For example, based on the observation that similar answers often get similar grades, Microsoft's Power Grading Basu et al. (2013) proposed a semi-automated approach. In their approach, they used some classic NLP features and statistical corpus-based features including length, tf-idf and Wikipedia-based LSA similarity and so on. Instead of directly grading student answers, the center of the Power Grading approach is to build a hierarchy of short-answer clusters. More specifically, the clusters are generated using 'Similarity Metric' among student answers or/and correct answers combined with logistic regression classifiers. Once the hierarchy of short-answer clusters is built, human grader can manipulate the clusters as needed and then assign a single score to all the answers belong to the same cluster. Additionally, Hou et al. (2010) combined four classes of features, POS tags, term frequency, tf-idf (short for short for term frequency–inverse document frequency) and entropy, with support vector machine to the task of ASAG. Madnani et al. applied logistic regression and used eight answer-based features such as BLEU, number of sentences, counts of discourse connector words (Madnani 2013) to the task of ASAG. In short, ML methods have become more and more prevalent on the task of ASAG (Burrows et al. 2015).

To summarize, prior research on ML-based approaches shows great promises to handle the poor regularization in short answers, such as spelling errors, unstructured sentences and limited lexical features. In this work, we incorporate the additional

feature resources, the *Ques* and *Stu* models, to the conventional *Ans* model and subsequently find a optimal feature space for ASAG by feature engineering. That is, rather than building *question-specific* ASAG models (Pulman and Sukkarieh 2005), we take advantage of the shared mutual information or features existed in the *Ques* representations to obtain a generalized classifier for all questions; rather than treating all answers from a student independently, our *Stu* would take the temporal trajectories of the student's answers into account by evaluating the student knowledge levels over time. Note that incorporating *Ques* and *Stu* models has been widely investigated and applied in other related fields such as psychometrics. Item response theory (IRT) (Baker and Kim 2004; An and Yung 2014), for example, calculates the probability that a user will generate a correct response based on the item's difficulty and the user's (i.e., student's) ability. In the work by Lalor et al. (2016), IRT is applied to design effective, adaptive test items appropriate for measuring users' ability. However, as far as we know, this is the first work to incorporate these two models to the task of ASAG. Moreover, while previous research mainly evaluated two or three ML methods, we evaluated six classifiers including one state-of-the-art deep learning model, DBN, together with 5 classic classifiers on a real-world dataset.

3 Method

In this following, Sect. 3.1 thoroughly describes the three primary models together with how the composite feature spaces are constructed; Sect. 3.2 presents the characteristics of the 5 classic ML methods with a focus on deep belief networks (DBN).

3.1 State features

To investigate the impact of state features on the task of ASAG, we extracted various features for the three primary models: the *Ans*, the *Ques* and the *Stu* models in manners of *individually* and *combined*. We also *compose* new features generated within or across the three primary models.

3.1.1 Answer (Ans) model: 6 features

Pervasively in prior answer-based ASAG approaches, systems are provided with so-called *referred correct answers*: expert-designed correct answers and the grading is generated by comparing between student answers and the referred correct answers for each question (Leacock and Chodorow 2003). Our approach is to use a *correct answer space*. The use of the correct answer space is directly motivated by Power Grading (Basu et al. 2013) in which similar answers were assigned by the similar grades. The constructing of the correct answer space is essentially a bootstrapping process: initially the space only contains the expert-designed referred correct answer, and it expands as subsequent student answer is identified as *correct* over the training process. Note that, both the referred correct answers and the correct answer space are question-specific. Particularly, a word-answer (like term-document) matrix is built for each question

using bag-of-words (BoW) where rows represent words, columns represent answers, and the cell in the matrix represents the number of times that a corresponding word (row) shows in an answer (column). The word-answer matrix updates dynamically as new correct answers are identified.

Based on the referred correct answers and the correct answer space, our *Ans* model essentially includes the following *six* features:

1. *Length difference* is measured as the absolute difference in the number of words between a student answer and the *referred correct answer*.
2. *Cosine similarity* is calculated using the tf-idf vectors of a student answer and that of the *referred correct answer*.
3. *Max-matched idf* measures the amount of information overlapped between a student answer and the correct answer space generated by then. Idf is commonly used to measure the amount of information a word provides. In our case, we first calculate the idf of each word in the word-answer matrix of the correct answer space and then it is compared against the word-answer vector generated from the student answer. To get the max-matched idf, the maximum idf from matching words is selected. Its default value is 0 if there is no matching word.
4. *Latent semantic analysis (LSA)* is a statistical technique that can measure the conceptual similarity between the two text sources. In our work, LSA is used to assess the quality of a student answer by comparing it with all the answers in the correct answer space. Suppose A is the $m \times n$ word-answer matrix, where m and n correspond to the number of words and answers. A can be decomposed as follows according to the theory of linear algebra:

$$A = U \Sigma V^T, \tag{1}$$

where U, $m \times m$, and V, $n \times n$, are orthogonal matrices and Σ, $m \times n$, is a diagonal matrix of the singular values. In specific, U is the matrix of the eigenvectors of $A^T A$ and V is the matrix of the eigenvectors of AA^T. In practice, keeping the top k singular values from Σ, and their corresponding singular vectors from U and V can approximate to A with small error:

$$A_k = U_k \Sigma_k V_k^T, \tag{2}$$

where U_k is $m \times k$, Σ_k is $k \times k$, and V_k is $n \times k$. Intuitively, the k ingredients correspond to k hidden concepts where the words and answers are involved. The decomposition enables the transfer of the word-answer matrix to a lower-dimensional semantic space. Hence we can compare two answers using their vectors $\Sigma_k \hat{\mathbf{a}}_i$ and $\Sigma_k \hat{\mathbf{a}}_j$ in this space.

When we get a new student answer, we firstly transform it into the low-dimensional space as follows:

$$\hat{\mathbf{a}}_n = \Sigma_k^{-1} U_k^T \mathbf{a}_n. \tag{3}$$

Then the cosine similarity will be calculated between the vector of the new answer $\Sigma_k \hat{\mathbf{a}}_n$ and all the vectors of answers in the correct answer space $\{\Sigma_k \hat{\mathbf{a}}_i\}$. Finally the averaged similarity will feed into the feature space.

Note that originally there is only one expert-designed correct answer in the correct answer space ($n = 1$), then cosine similarity is directly calculated and LSA will not be performed. Later LSA will be re-decomposed when a new correct answer joins.

5. *Domain-specific text similarity* measures the text similarity between a student answer and the correct answer space at the *sentence* level with a focus on a domain-specific vocabulary. To do so, our experts manually construct a domain-specific word list dl consisted of terminologies and abbreviations in a specific domain. For example, in the domain of introductory physics, kinetic energy is included in the dl. The *domain-specific text similarity* $sim_d(s, c)$ is calculated between the student answer **s** and a correct answer **c** in the correct answer space by then. Through the BoW approach, both **s** and **c** are decomposed to word vectors denoted as **sv** and **cv**, respectively. $sim_d(s, c)$ is calculated using Eq. 4, where **1** is an indicator function: $\mathbf{1}_{dl}(w_1) = 1$ if the word w_1 is in the list of dl otherwise 0.
$$\text{sim}_d(s, c) = \sum_{\substack{w_1 \in \\ sv}} \sum_{\substack{w_2 \in \\ cv}} \mathbf{1}_{dl}(w_1) \cdot \mathbf{1}_{dl}(w_2) \tag{4}$$
6. *General text similarity* measures the general text similarity between **s** and **c**, denoted as $sim_g(s, c)$. To calculate it, we first define the *word*-level similarity denoted as sim_w defined by Wu and Palmer shown in Eq. 5:
$$\text{sim}_w(C_1, C_2) = \frac{2 * \text{depth(LCS)}}{\text{depth}(C_1) + \text{depth}(C_2)}. \tag{5}$$

In Eq. 5, C_1 and C_2 refer to two concepts and depth$(\cdot)$ represents the number of edges of a concept along a concept tree, which is generally built based on a knowledge-based dictionary. Here we used WordNet. WordNet is essentially a graph with a hierarchical structure, and concepts, or words, are the nodes in this tree structure. 'depth' represents the number of edges of a concept along the concept tree.

The least common subsumer (LCS) in Eq. 5 is defined as 'the most specific concept which is a *common* ancestor' of two concepts: C_1 and C_2 in the concept tree. When calculating sim_w between two words, we map each word to its synonym set which is used as the concept in Eq. 5. For example, 'car' is mapped to the concept set of {car, auto, motorcar}.

Once the word similarities are determined, the sentence similarity $sim_g(s, c)$ is calculated by summing over the word similarities of each pair of words from **sv** and **cv** shown in Eq. 6. In Eq. 6, dl^c is the *complement* of the dl, that is, dl^c includes all the words that are *not* in the domain-specific word list dl.

$$\text{sim}_g(s, c) = \sum_{\substack{w_1 \in \\ sv}} \sum_{\substack{w_2 \in \\ cv}} \mathbf{1}_{dl^c}(w_1) \cdot \mathbf{1}_{dl^c}(w_2) \cdot \text{sim}_w(w_1, w_2) \tag{6}$$

Equation 7 shows that the overall similarity between **s** and **c**, denoted as $\text{sim}(s, c)$, can be calculated by combining the domain-specific similarity $\text{sim}_d(s, c)$ and domain general similarity: $\text{sim}_g(s, c)$. In Eq. 7, $\text{sim}(s, c)$ is defined as a *weighted* combination of $\text{sim}_d(s, c)$ and $\text{sim}_g(s, c)$. Grid search results showed that $\{0.6, 0.4\}$ is the

optimal weight settings for $\text{sim}_d(s, c)$ and $\text{sim}_g(s, c)$, respectively. Thus, for our data the domain-specific similarity is weighted more important than the domain general similarity. Additionally, Z is the normalization factor.

$$\begin{aligned} \text{sim}(s, c) =& \frac{0.6 \cdot \text{sim}_d(s, c) + 0.4 \cdot \text{sim}_g(s, c)}{Z}, \\ Z =& \sum_{\substack{w_1 \in \\ sv}} \sum_{\substack{w_2 \in \\ cv}} \{0.4 \cdot \mathbf{1}_{dl^c}(w_1) \cdot \mathbf{1}_{dl^c}(w_2) \\ & + 0.6 \cdot \mathbf{1}_{dl}(w_1) \cdot \mathbf{1}_{dl}(w_2)\}. \end{aligned} \tag{7}$$

Finally, given that the correct answer space often has multiple correct answers, the final $\text{sim}(s, c)$ is defined as the *average* similarity of the student answer with each answer in the correct answer space generated by then.

3.1.2 Question (Ques) model: 9 features

While most existing ASAG systems are question-specific, our ASAG system is domain general. That is, instead of building one classifier for each question, we build one classifier for all questions. To do so, the *Ques* model decomposes questions into a general feature space so that our ASAG can learn grading experience from questions that share common or similar features. More specifically, our *Ques* model includes the following two major types of features:

Knowledge Components (KCs) Based formally, VanLehn defined KCs as 'a generalization of everyday terms like concept, principle, fact, or skill, and cognitive science terms like schema, production rule, misconception, or facet' (VanLehn et al. 2007). For the purpose of intelligent tutoring systems (ITSs), KCs are atomic units of knowledge. STEM domains such as physics or computer science are often seen as structured as a set of KCs and in complex domains, solving one question often involves in applying multiple co-occurring KCs.

The central idea of including KCs in our *Ques* model is to build a *Q-matrix* to represent the relationship between individual question and KCs. Q-matrix is typically encoded as a binary 2-dimensional matrix $q \times k$ with rows q representing questions and columns k representing KCs. For example, if a given cell $Q_{jk} = 1$, then question j is an application of KC k. In previous work, researchers have focused on the task of generating or tuning Q-matrix based upon a dataset (Barnes 2005; Tatsuoka 1983). For our work we employ a Q-matrix, where the KCs for each question are manually specified by experts. In the learning process, if a student is answering a question j, then the row corresponding to this question is fetched from Q-matrix, as shown in Eq. 8, and added to the feature vector of the current answer. For instance, if question j is associated with KC_1 and KC_2, then we have $Q_{j1} = 1$, $Q_{j2} = 1$ and $Q_{jk} = 0$ for the remaining KCs. The resulted representation is: [1, 1, 0, ..., 0].

$$Q_j = [Q_{j1}, Q_{j2}, \ldots, Q_{jK}] \tag{8}$$

By decomposing questions to a set of KCs, we can build connections among them. For example, two questions may appear to be very different but actually share the same set of KCs. In this case, grading experience from one question can be handily used for the other. Thus, using Q-matrix can potentially solve the data sparsity and cold-start problems. The former arises from the fact that sometimes little training data are available for a question and the latter arises from the fact that new question has no user data history and thus we have no training data to train our ASAG system. In this work, we focused on 8 primary KCs, which resulted in 8 KC-related *Ques* features.

The 9th *Ques* feature is *questionDifficulty*, indicating the difficulty level of a question. In prior work on modeling learning process, *questionDifficulty* has consistently shown to be one of the most effective features (Chi et al. 2011). In this work, *questionDifficulty* is estimated based on the percentage of answers for a question that is correct so far. It is initialized to a default value by experts and updates dynamically as new student answers are graded.

3.1.3 Student (Stu) model: 16 features

The impetus for the development of many interactive e-learning environments including ITSs is the desire to capture the effective learning experience provided by human one-on-one instruction. Ideally, an effective learning environment should craft and adapt its decisions to an individual's motivation, aptitude, attitude, engagement and incoming competence, etc. (Vanlehn 2006). However, creating an adaptive, personalized learning system that meets students' requirements can be challenging and a solution to this challenge is using student modeling. Student modeling can be defined as the process of gathering relevant information in order to infer the current cognitive state of the student and to represent it so as to be accessible and useful to the tutoring system for offering adaptation (Thomson and Mitrovic 2009). While student modeling has been widely applied in the context of ITSs, as far as we know, this is the first attempt to use it for the task of ASAG.

Bayesian Knowledge Tracing (BKT) is one of the most widely adopted student modeling methods in ITS (Corbett and Anderson 1994). BKT leverages sequences of observations (e.g., *correct*, *incorrect*) from student–system interaction log files to continually update the estimate of student latent knowledge levels: *unlearned* vs. *learned*. Figure 1 shows a graphical representation of the BKT model where the nodes S represent hidden knowledge states and the nodes O represent observation of students' behaviors. The edges between the nodes represent their conditional dependence.

Fundamentally, the BKT model is a two-state hidden Markov model (HMM) (Eddy 1996) characterized by five elements:

1. **N**, the number of different types of hidden state;
2. **M**, the number of different types of observation;
3. **Π**, the initial state distribution $P(S_0)$;
4. **T**, the state transition probability $P(S_{t+1}|S_t)$ and
5. **E**, the emission probability $P(O_t|S_t)$.

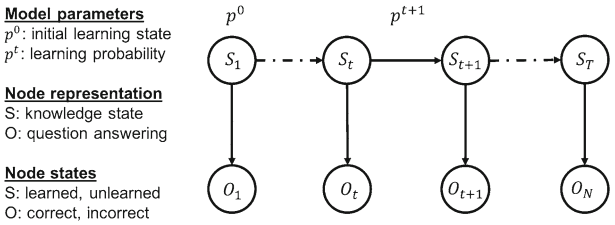

Fig. 1 The standard BKT model

Conventional BKT assumes $\mathbf{N} = 2$ hidden knowledge states: *unlearned* or *learned* and $\mathbf{M} = 2$ observations: *incorrect* or *correct*. That is whether a student's input answer (performance) is correct or not.

Once elements **N** and **M** are determined, $\mathbf{\Pi}$, **T** and **E** can be derived from student observation sequences. BKT makes two assumptions about its conditional dependence as reflected in the edges in Fig. 1. One is that a student's state at time $t + 1$ is only contingent on his/her state at time t. The other is a student's performance at time t is only dependent on his/her knowledge state at time t. These two assumptions are captured by the state transition probability **T** and the emission probability **E**. To derive $\mathbf{\Pi}$, **T** and **E**, Baum-Welch algorithm (or EM method) is used iteratively until a maximized probability of observing the student training sequences is achieved.

To fit in the context of student learning, BKT further defines five parameters:

1. **PriorKnowledge** = P(S_0 = learned);
2. **Learningrate** = P(learned|unlearned);
3. **Forget** = P(unlearned|learned);
4. **Guess** = P(correct|unlearned);
5. **Slip** = P(incorrect|learned).

The original BKT is KC-specific. That is, it assumes that for different KCs students would have different prior knowledge, learning rate, forget, guess and slip rates. Therefore, when applying BKT to our *Stu* model, we used a learning state vector L at time t, which is constructed by measuring student knowledge states on each specific KC at time t shown in Eq. 9:

$$L_t = \{l_t^1, l_t^2, \ldots, l_t^k, \ldots, l_t^K\}, \tag{9}$$

where K is the total number of KC in the domain and, in this work, we have $K = 8$. For any given time t, we have $l_t^k = p_t^k$, the probability the student is in learned state on KC k at time t. Since BKT assumes that student learning process is a Markov chain, p_t^k can be calculated using Eq. 10.

$$\begin{aligned} p_t^k &= P(S_t^k = \text{learned}|S_{t-1}^k, S_{t-2}^k, \ldots, S_1^k) \\ &= P(S_t^k = \text{learned}|S_{t-1}^k). \end{aligned} \tag{10}$$

Besides using the eight KC-specific knowledge tracing parameters to track the student's knowledge by then, we also integrate student pretest scores (how well the

student starts with) to our *Stu* model. We have: $E = \{e^1, e^2, \ldots, e^K\}$, where e^k represents the student pretest score on KC k. In short, our *Stu* model has a total of 16 features: L_t, the learning probability calculated from BKT on the $K = 8$ KCs so far (by time t), and E, the student pretest scores on all $K = 8$ KCs. So for a student i at time t, we have the following representation:

$$\begin{aligned} S(i) &= \{L_t(i), E(i)\} \\ &= \{l_t^1(i), \ldots, l_t^K(i), e^1(i), \ldots, e^K(i)\}, \end{aligned} \tag{11}$$

where $L_t(i)$ is updated over the student i's training trajectory using the trained BKT model and $E(i)$ is static vector determined by the student i's pretest.

To summarize, among the three primary models, the *Ans* model uses six features similar to those explored in most prior ASAG research, and the *Ques* model includes a total of nine features: the Q-matrix and the problem difficulty, to grasp the characteristics of questions which would allow us to generate a question-general ASAG system, and the *Stu* model includes 16 features: for each of the eight KCs, it uses the BKT model to estimate the student's knowledge on the KC at any given time and the student's KC-specific pretest score. In the following, we explored the three primary models individually and our results showed that the *Ans* model is indeed the best among them. Then we further combined the *Ans* model with the *Ques* model, the *Stu* model, or both. Our goal is to explore whether the other two models can further contribute to the task of ASAG beyond the *Ans* model alone. Additionally, we investigated on constructing composite feature space using Gaussian mixture model (GMM) and Cartesian product (CP).

3.1.4 Composite feature space

Student clustering (SC) is explored by grouping students using Gaussian mixture model (GMM). Since both knowledge states and pretest scores in the *Stu* model are KC-specific, we also include the Q-matrix in the *Ques* model. That is, our clustering dataset consists of each student's answer at each time during his/her training. More specifically for a student i at time t who is answering the question j, we form a new vector: the student i's pretest scores $E(i)$, his/her knowledge states at time t $L_t(i)$, and the corresponding target question j's Q-matrix: Q_j, shown in Eq. 12. Note that here we assume all student answers are independent of each other.

$$SC = \{E(i), L_t(i), Q_j\} \tag{12}$$

GMM clustering results in a fuzzy clustering in that instead of assigning the data to a specific cluster, we report a vector which provides the probabilities of the data belongs to each cluster. This fuzzy clustering is able to keep more complete information than the hard-assigned clustering. Compared with the original *Stu* model, the clustering can be seen as a more compact representation.

Cartesian product—CP(Q, S) is short for the Cartesian product (CP) of the *Ques* and the *Stu* models referred as *CP(Q, S)*. The goal is to explore the connections across

different KCs since a student's performance on one KC can imply his/her performance on other KCs. For example, in the domain of work and energy, the student's knowledge of kinetic energy directly implies his/her knowledge of total mechanical energy since the former is a part of the latter. For a student i's answer to a target question j at time t, we use the Q-matrix for the corresponding target question j in the *Que* model, the student i's knowledge states at time t: $L_t(i)$ and the student i's incoming competence: $E(i)$ in the *Stu* model to calculate CP(Q, S). More specifically, we have:

$$CP(Q, S) = \{Q_j \times L_t(i), Q_j \times E(i)\}, \tag{13}$$

where '$\times$' represents the Cartesian product and we have:

$$\begin{aligned} Q_j \times L_t(i) &= \{Q_{j1} \cdot l_t^1(i), \ldots, Q_{jk} \cdot l_t^k(i), ..., Q_{jK} \cdot l_t^K(i)\}, \\ Q_j \times E(i) &= \{Q_{j1} \cdot e^1(i), \ldots, Q_{jk} \cdot e^k(i), ..., Q_{jK} \cdot e^K(i)\}. \end{aligned} \tag{14}$$

Cartesian product—CP(Q, SC) we also apply Cartesian product (CP) using the Q-matrix in the *Ques* model and student clustering (SC), the fuzzy clustering derived above, referred as *CP(Q, SC)* in the following. The motivation is that SC can be more reliable representation of the original student model. To calculate CP(Q, SC), we have:

$$CP(Q, SC) = \{Q_j \times SC\}. \tag{15}$$

3.2 Six classifiers

Several classic ML approaches have been successfully applied to the task of ASAG. For example, Sima et al. (2009), Pulman and Sukkarieh (2005) employed naïve Bayes (NB) and decision tree (DT) to extract patterns from student answers; (Madnani 2013) comprised eight state features and applied logistic regression (LR); (Hou et al. 2010) employed SVM. In recent years, deep learning has shown tremendous success in various applications (Mohamed et al. 2012; Li et al. 2015; Jia et al. 2017) and one of the most widely implemented deep learning models is DBN (Jia et al. 2014, 2015; Mohamed et al. 2009, 2012). Given the space limitation, we briefly describe DBN in the following.

Generally speaking, DBN has two stages: an unsupervised pre-training stage and a supervised fine-tuning stage. In the pre-training stage, DBN utilizes the stacked restricted Boltzmann machine (RBM) layers to extract high-level latent and often more representative features from the original input features. In the second stage, the extracted latent features are used to train a deep neural network following a standard fine-tuning supervised training process. It is widely believed that the power of DBN mainly comes from the unsupervised pre-training stage; thus, we briefly describe the mechanism of RBM in the following.

Figure 2 displays the structure of RBM, which can be seen as a restricted Markov random field. It consists of two layers of variables, visible units V and hidden units H: V denotes the original input feature representation, and H denotes the extracted hidden feature representation. RBM is a degenerate version of Boltzmann machine

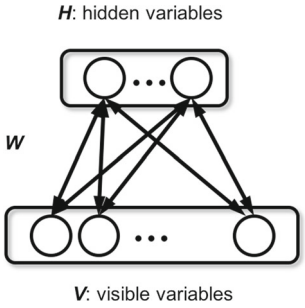

Fig. 2 Structure of RBM

since it has *no interconnections* between units in the same layer but *across layers*. The joint distribution of V and H is defined by an energy-based probabilistic model as:

$$P(V, H) = \frac{\exp(-E(V, H))}{Z},$$
$$Z = \sum_{V,H} \exp(-E(V, H)), \tag{16}$$

where denominator Z serves as the normalizer for the joint probability, and the energy function $E(V, H)$ is defined based on the bipartite structure of RBM as:

$$E(V, H) = -V^T W H - B^T V - C^T H, \tag{17}$$

where W denotes the weights between V and H. More specifically, $W_{i,j}$ is the weight between an original input feature V_i and an extracted hidden feature H_j; B and C denote the biases for visible units and hidden units, respectively.

Each unit of V and H is independent of other units in the same layer; thus, we can derive the conditional distribution $P(v|h)$ and $P(h|v)$ from the joint distribution shown in Eq. 16. Moreover, $P(v|h)$ and $P(h|v)$ are fully factorial and can be calculated as:

$$P(V_j|H) = \sigma(W_{.j}^T H + B_j),$$
$$P(H_i|V) = \sigma(W_{i.} V + C_i), \tag{18}$$

where $\sigma(\cdot)$ denotes the logistic sigmoid function.

The model parameters $\theta = \{W, B, C\}$ are updated and derived from Eq. 16 through gradient descent as shown in Eq. 19:

$$\frac{\partial P(V, H)}{\partial \theta} = -\mathbb{E}_{H|V}\left[\frac{\partial}{\partial \theta} E(V, H)\right] + \mathbb{E}_{V,H}\left[\frac{\partial}{\partial \theta} E(V, H)\right]. \tag{19}$$

Due to the intractability of gradient computation brought by the factor Z, the training of RBM (i.e., pre-training phase) follows the contrastive divergence algorithm (Hinton

2002), which executes K steps of alternating Gibbs sampling following Eq. 18 to approximate the gradient. For instance, the weight matrix W can be updated as follows:

$$\frac{\partial P(V,H)}{\partial W} = \sigma(W_{i.}V + C_i)V' - \sigma(W_{i.}\tilde{V} + C_i)\tilde{V}', \tag{20}$$

where $\tilde{V}$ denotes the reconstructed data obtained from K steps of Gibbs sampling. The parameters B and C can be updated in a similar pattern.

In this work, we compare DBN against the five classic classifiers: the aforementioned four methods used in prior ASAG research: naïve Bayes (NB), logistic regression (LR), decision tree (DT) and support vector machine (SVM) and one additional method: artificial neutral network (ANN) due to its popularity and efficiency.

Note that, there exists nonlinear relationship among the features across the three models and within each individual model. In particular, both *Ques* and *Stu* are constructed using the same set of KCs, and thus they are correlated. The KCs are also correlated in each individual model, e.g., in *Stu*, a student's pretest score on one KC might imply his/her performance in the following learning process on the same KC. Among the six ML classifiers, NB assumes that each state feature is independent given its output class and LR can only linearly combine state features. Therefore, we expect that NB and LR would not perform well given their limited capability of exploring feature connections. For the remaining four, DT can synthesize the features at different levels to make classifications; the hidden layers in ANN and the kernel functions in SVM allow them to effectively derive the nonlinear feature mappings; finally DBN can extract even more representative features via a separate unsupervised pre-training procedure.

4 Experiments

4.1 Data description

Our training corpus is collected from Cordillera (VanLehn et al. 2007; Carolyn 2007), a natural language tutoring system that teaches students introductory college physics. It consists of a subset of the physics work-energy domain, which is characterized by *eight* primary KCs including kinetic energy, gravitational potential energy and so on.

In total, 158 students participated in the data collection following five steps: (1) taking a background survey; (2) studying the textbook and the prerequisite materials; (3) taking a pretest; (4) training on Cordillera, and (5) taking a post-test. In total, there are 482 different short questions involved in the training session and it took students roughly 4–9 h to complete the training.

In Cordillera, students interact with tutors by means of natural language entries and thus our training data are composed of tutorial dialogues between students and Cordillera. Each dialogue between a student and Cordillera in the training corpus can be seen as a sequence of tutor-question and student-answer episodes/interactions. The average number of Cordillera-student episodes in a sequence is about 280. Each episode consists of a tutor question and a student answer to the question. One such

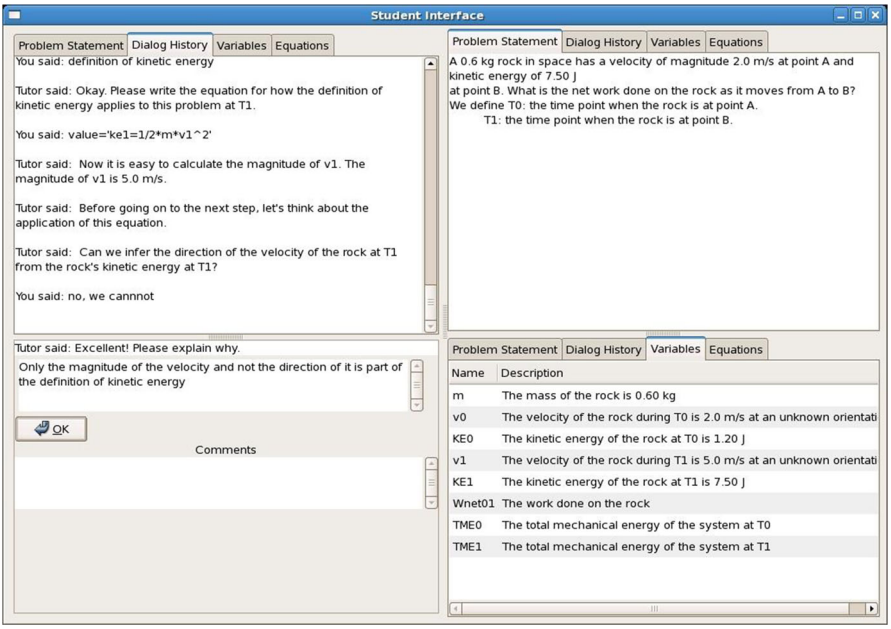

Fig. 3 Student interface in Cordillera

episode example is shown in Fig. 3. When collecting the training data, Cordillera used human wizards as the natural language understanding module (Bernsen et al. 2012). The *only* task for the human wizards was to match student answers to the closest response from a list of potential responses. Then Cordillera automatically labeled student answers correct or incorrect based on the response selected by human wizards. Thus, in our training data, each student answer has two types of output labels: one is the human wizard's mapped answer from the list of potential responses for the question, and the other is whether the student answer is ***correct*** or ***incorrect***. Since our task here is to automatically predict whether the student answer is correct or incorrect, the latter is the ground truth label in this work.

In Cordillera, questions are categorized into qualitative vs. quantitative ones. Quantitative questions require students to input mathematical equations, which is not the focus of ASAG. Therefore, they are excluded from our training corpus and we mainly concentrate on qualitative answers with natural language as input. As a result, there are 16228 out of episodes/interactions that are included in our training corpus. The average length of student answers in these dialogues is 7.6 words, which are qualified as short answers. Note that the majority of student answers is correct and the majority class of our training data is 61.66%. Additionally, a series of natural language pre-processings have been conducted on our training corpus:

– Standard natural language pre-processings: stop word removal, tokenization, punctuation removal and word correction.
– Domain-specific pre-processings: several acronyms written by students are transformed into unified forms. For instance, TME, KE are expanded to their full forms,

namely total mechanical energy, kinetic energy, respectively. This transformation is helpful for getting the true text similarity between student and the correct answer space.

Each episode is formalized as a feature vector extracted from the three primary models. Specifically, for the *Ans* model, we extract length difference, cosine similarity, max-matched idf, LSA and weighted text similarity and so on; for the *Ques* model, we fetch the corresponding KC information (for this domain, we have eight primary KCs) from Q-matrix and the *questionDifficulty* for the tutor question; finally, for the *Stu* model, we include 16 features (8 KC-based learning probabilities and 8 KC-based pretest scores) representing the student's learning states by then (before answering the tutor question) and incoming competence.

4.2 Exploration of feature space

The exploration of feature space can be divided into three stages in a progressive manner as summarized in Table 2.

In Stage 1, we evaluated the performance of *Ans*, *Ques* and *Stu* individually. Since most existing ASAG systems are mainly answer-based, our goal in Stage 1 was to investigate whether *Ques* or *Stu* can be used as alternative models. Our results showed that while the *Ans* model indeed performs the best, the *Ques* and *Stu* models can also contribute to the task of ASAG.

In Stage 2, we explored different ways of combining either *Ques*, or *Stu*, or both to the *Ans* model. Thus, we constructed three combined models: AQ for *Ans* + *Ques*; AS for *Ans* + *Stu*; and AQS for using all primary models. Our results showed that the best model is AQS.

Finally, in Stage 3, we further explored if the composite features described in Sect. 3.1.4 could facilitate a better grading performance beyond AQS, the best model from previous two stages. More specifically, we explored the following different ways: AQS + student clustering (SC), AQS + SC + Cartesian product (*Ques*, *Stu*) and finally AQS + SC + Cartesian product (*Ques*, SC), referred as CF1, CF2 and CF3, respectively, hereinafter.

4.3 Exploration of six classifiers

We evaluated the six ML classifiers, NB, LR, DT, ANN, SVM and DBN using 10-fold cross-validation (CV) on our training corpus. We randomly split the data (16,228 pieces of training data) into 10-fold. In one iteration of cross-validation, we use eightfold for training with randomly initializing the model, onefold for validation and the remaining fold for testing. We have run 10 iterations in total to get every fold serve as the test set. The average performance of each evaluation metric is reported. For DT, we adopted the standard CART algorithm. For ANN, we used one hidden layer and ran grid search to determine the optimal number of nodes to be 70. For SVM, we explored linear-SVM and SVM with different kernels and our results showed that RBF kernel achieved the best performance; thus, SVM with RBF kernel was reported in the following. For

Table 2 Nine feature spaces in three stages

Stage	Feature Abbr.	Construction
Stage 1 Primary	*A(ns)*	Ans model
	Q(ues)	Ques model
	S(tu)	Stu model
Stage 2 Combined	*AQ*	A + Q
	AS	A + S
	AQS	A + Q + S
Stage 3 Composite	*CF1*	*AQS* + SC (student clustering)
	CF2	*AQS* + SC + CP(Q,S)
	CF3	*AQS* + SC + CP(Q,SC)

CP Cartesian product

DBN, we used three hidden layers and similar to ANN, grid search results determined the number of nodes for each hidden layer to be 74, 34, 10, respectively.

4.4 Evaluation metrics

For the task of ASAG, we treat 'incorrect' as the positive class because it is more important for the system to know whether the student answer is incorrect. We use the following five evaluation metrics:

1. *Accuracy* is most widely used and indicates how many answers are correctly graded;
2. *Area under the curve (AUC)* is the area under the receiver operating characteristic (ROC) curve created by plotting the true-positive rate (TPR) against the false-positive rate (FPR) at various threshold settings. The AUC for a random predictor would be 0.5. AUC is a preferable evaluation metric than accuracy, especially when the ratio of classes in the data is not perfectly balanced. For example, if only 10% of the data is positive, the accuracy would still be 0.9 even we learn a bad predictor which classifies all the data to be positive. In contrast, AUC measures the overall performance of TPR and FPR rate given different thresholds and testify the performance of the model particularly on positive data.
3. *Precision* informs in all the predicted incorrect answers (positive class), how many are actually incorrect;
4. *Recall* informs in all the originally incorrect answers, how many are predicted as incorrect;
5. *F-measure* also known as F1-score provides an effective assessment by taking both precision and recall into account:

$$\mathrm{F1} = \frac{2 \cdot \mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{21}$$

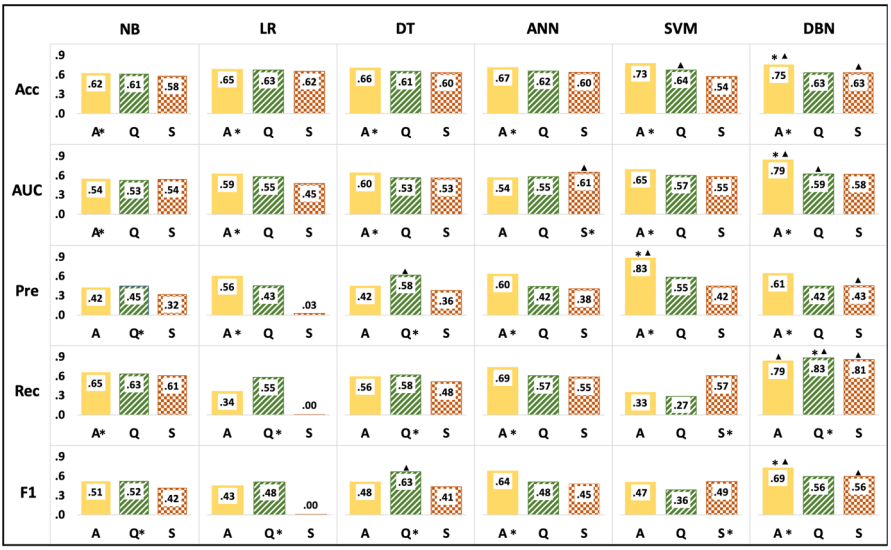

Fig. 4 Stage 1: performance of the three primary models

We report experimental results on all five measures with emphasis on AUC and F-measure since they are associated with both the true-positive rate and the false-positive rate.

5 Results

In the following, we report our results from each of the three stages as described above. Given that *Ans* is the fundamental model and widely studied by previous research, different models in each stage are compared against the *Ans* model. For each stage, we elaborate on the experiments on two viewpoints: comparing among the various feature spaces in the current stage and analyzing the effectiveness of different ML approaches for that stage.

5.1 Stage 1: exploring the three primary models

Figure 4 shows the 10-fold CV results on comparing the *Ans*, *Ques* and *Stu* models individually. In Fig. 4 and the figures, we have:

'$*$' represents the best feature space model for a specific ML classifier. For example, in the right-uppermost cell in Fig. 4, it compares the accuracy of the three primary models using DBN. Among the three models, *Ans* reaches the highest accuracy and thus A is labeled with '$*$'.

'▲' represents the best ML classifier when using a specific feature space. For example, in the first row in Fig. 4, for the *Ques* model, SVM reaches the highest accuracy among the six ML classifiers and thus the Q column in the SVM cell is labeled with '▲' on the top.

'∗▲' represents the best result across all feature spaces and classifiers. For example, in the first row in Fig. 4, the best accuracy is achieved when using *Ans* and DBN and thus A column in the DBN cell is labeled with '∗▲.'

The standard deviation of 10-fold cross-validation for each model and each classifier is within 5%.

5.1.1 *Ans* vs. *Ques* vs. *Stu*

For accuracy, Fig. 4 shows that the *Ans* model is the winner across all six ML classifiers. More specifically, 1) when using *Ans* alone, all of the six ML classifiers beat the simple majority baseline (0.617) ; 2) when using *Ques* alone, four out of six classifiers beat the majority baseline and the two exceptions are NB and DT with an accuracy of 0.608 and 0.614, respectively; 3) when using *Stu* alone, only DBN beats the baseline. The best accuracy comes from using *Ans* with DBN.

For AUC, *Ans* leads the other two primary models using five out of six classifiers; the exception is for ANN, *Stu* is the best model. Despite this, the best performance on AUC comes from using *Ans* with DBN. For precision, *Ans* leads in four of six ML classifiers: LR, ANN, SVM and DBN, while *Ques* takes the highest precision for NB and DT. Note that using the *Stu* model would generate pretty poor precision across the six classifiers, especially for LR. The best precision comes from using the *Ans* model with SVM. For recall, much to our surprise, *Ans* only leads in two out of six classifiers: NB and ANN; *Ques* leads in three out of six classifiers: LR, DT and DBN; and *Stu* leads in SVM. The best recall comes from using the *Ques* model with DBN. Finally, for F-measure, *Ans* leads in two out of six classifiers: ANN and DBN; *Ques* leads in three out of six classifiers: NB, LR and DT; and *Stu* leads in SVM. The best F-measure comes from using the *Ans* model with DBN.

To summarize, the *Ans* model outperforms the *Ques* and *Stu* models on accuracy, AUC, precision and F-measure. However, on recall, the *Ques* performs the best.

5.1.2 Comparing the six ML classifiers in Stage 1

Across the six classifiers, DBN outperforms the five classic classifiers on four out of five measures: accuracy, AUC and F-measure using the *Ans* model and recall using the *Ques* model. The only exception is on precision: the best precision comes from using SVM with the *Ans* model.

Next, we sort the six classifiers in the order of their performance, which is determined by the best performance among the three *Ans*, *Ques* and *Stu* models (highest value in the cells).

On both accuracy and AUC, we have:

$NB < LR < DT < ANN < SVM < DBN$;

on precision:

$NB < LR < DT < ANN < DBN < SVM$;

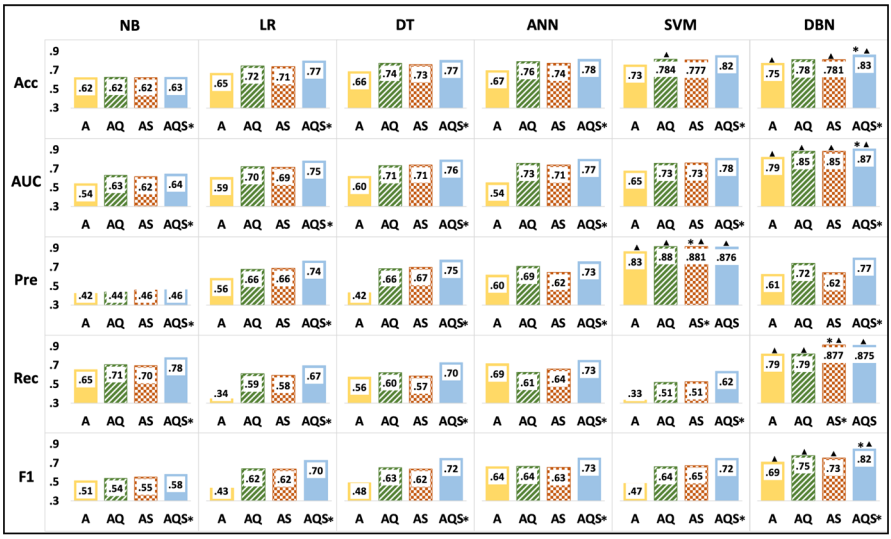

Fig. 5 Stage 1: performance of the three primary models

on recall:
$LR < SVM < DT < NB < ANN < DBN$;
and on F-measure:
$LR < SVM < NB < DT < ANN < DBN$.

To summarize, DBN generally outperforms the five classic classifiers except that on precision, SVM is the best. Across the three primary feature spaces and six classifiers, *Ans* with DBN gains the best performance across three out of five measures except that the *Ques* model with DBN achieves the best recall and the *Ans* model with SVM achieves the best precision. It is noteworthy that the best recall and F-measure under five classic classifiers are generated using either the *Ques* model for LR or the *Stu* model for SVM. Therefore, we hypothesize that adding the *Ques* model and the *Stu* model to the *Ans* model can further improve the task of the ASAG.

5.2 Stage 2: exploring the three combined models

Figure 5 shows the performance of AQ, AS and AQS across the six ML classifiers compared to that of *Ans*.

5.2.1 *Ans* vs. *AQ* vs. *AS* vs. *AQS*

For accuracy, AUC and F-measure, *AQS* outperforms the other three models across all six classifiers as *AQS* is labeled with '*' across all the cells in rows 1, 2 and 5 in Fig. 5.

For precision, the *AQS* model outperforms the other three models across five out of six classifiers; the exception is on SVM, *AS* has the best precision, but *AQ* and *AQS* are closely competitive. Also note that on NB, all models get poor performance. For recall, the *AQS* model outperforms the other three models across five classifiers and the exception is on DBN. While the best recall comes from using *AS* with DBN 0.877, the performance of *ASQ* with DBN is very close: 0.875.

A* vs. *AQ: by adding the *Ques* model into the *Ans* model as *AQ*, the effectiveness of the ASAG is improved on almost all of five evaluation metrics across all classifiers. The only exception is the recall under ANN decreased from 0.691 for *A* to 0.606 for *AQ*. On all other measures, the improvements are big: for example, the accuracy increases from 0.646 for *Ans* to 0.719 for AQ under LR and from 0.728 to 0.784 under SVM. The AUC is improved from 0.589 to 0.696 under LR and from 0.601 to 0.708 under DT. The precision goes from 0.564 up to 0.656 under LR and from 0.830 up to 0.880 under SVM.

A* vs. *AS: by adding the *Stu* model into the *Ans* model as the *AS*, we can observe the similar patterns. Most of the measurements of all classifiers are improved except the recall and F-measure are decreased using ANN.

Finally, Fig. 5 shows that without any exception, *AQS* outperforms *Ans* across all five evaluation metrics and across all six ML classifiers. Sometimes the improvements are large. *AQS* also outperforms both *AQ* and *AS* across classifiers and across the evaluation measures; the exceptions are on precision using SVM and recall using DBN where *AS* has slightly higher values that the *AQS*. These results suggest that while adding either *Ques* or *Stu* to *Ans* seems equally effective, the *Stu* and *Ques* models contribute different information for the task of ASAG; thus, the highest effectiveness is achieved by combining all three primary models together.

5.2.2 Comparing the six ML classifiers in Stage 2

Generally speaking, it seems that DBN is the best classifier again in this stage in that it has the best performance across all evaluation metrics except precision, where the best result is generated by SVM. Similar as in Stage 1, we sort the six classifiers in the order of their performance, which is determined by the best performance among the feature spaces in this stage. Like in Stage 1, the same order is kept for accuracy and AUC:
$NB < LR < DT < ANN < SVM < DBN$;
On precision:
$NB < ANN < LR < DT < DBN < SVM$;
on recall:
$SVM < LR < DT < ANN < NB < DBN$;
and on F-measure:
$NB < LR < DT < SVM < ANN < DBN$.

The best accuracy, AUC and F-measure come from using *AQS* model with DBN. Overall, *AQS* implemented on DBN is the optimal model. Meanwhile, it is noted that the improvement of the five classic classifiers from Stage 1 to Stage 2 is significantly larger than that of DBN from Stage 1 to 2. For example, from the best model in Stage 1: *Ans* to the best model in Stage 2: *AQS*, LR improved 27% on the AUC and SVM

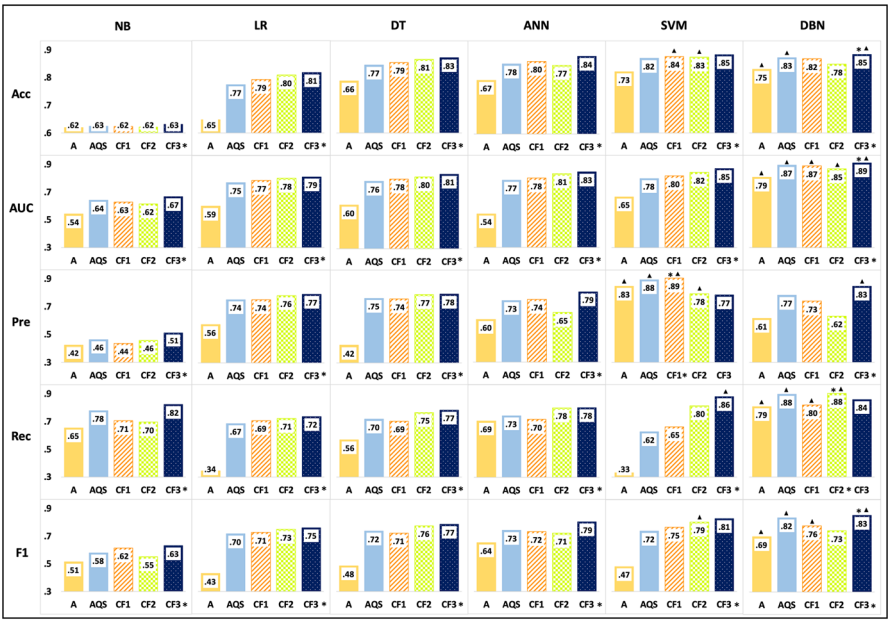

Fig. 6 Stage 3: performance of the composite models

improved 20% on it, while DBN improved 10%. It suggests that the integration of *Stu* and *Ques* benefits the classic ML classifiers more than DBN probably because the pre-training process in DBN already captures the latent relationships even using simple feature space.

5.3 Stage 3: three composite models

Given that *AQS* is the best model in Stage 2, we treat both *Ans* and *AQS* as baselines and explore whether the effectiveness of six classifiers can be further improved by adding composite features. Figure 6 shows the performance of CF1, CF2 and CF3, whose feature constructions are described in Table 2 and also in the footnote of Fig. 6.

5.3.1 *Ans* vs. *AQS* vs. *CF1* vs. *CF2* vs. *CF3*

Figure 6 shows that the CF3 outperforms all other four models: *Ans*, *AQS*, CF1 and CF2 on each cell (labeled with '$*$') with only two exceptions: the precision of SVM under CF1 and the recall of DBN under CF2. The advantage of CF3 over *AQS* demonstrates the effectiveness of using composite features; the advantage of CF3 over CF1 shows the effectiveness of using Cartesian product. The difference between CF2 and CF3 lies in the different choices of features used for Cartesian product. The result shows that the profound latent association between the *student clustering (SC)* and the *Ques* model would contribute more to the task of ASAG than that between the *Stu* model

and the *Ques* model. That is, in contrast with the *Stu* model, the SC seems to be more compact and representative.

Additionally, Fig. 6 shows that compared with the improvement from *Ans* to *AQS*, the improvement from the *AQS* model to the three composite models is relatively smaller. While across all five evaluation metrics the best models (labeled with '∗▲') are all from the three composite models, the difference of these best models with the corresponding *AQS* model is relatively small: on accuracy, we have 0.85 for the best model—CF3 and 0.83 for *AQS*; on AUC, we have 0.89 for the best model of CF3 and 0.87 for *AQS*; on precision, CF1 is the best model with a value of 0.89 and that is 0.88 for *AQS*; on recall, we have 0.88 for CF2 and 0.87 for *AQS*; finally on F-measure, the best model is CF3 with a value of 0.83 and the value is 0.82 for *AQS*. Additionally, Fig. 6 shows that for several cases the composite models can perform even worse than the *AQS*. For example, when using DBN the F-measure of CF1 and CF2 is worse than that of *AQS*: 0.76, 0.73 and 0.82, respectively; when using SVM, the precision of CF3 is worse than *AQS*: 0.88 vs. 0.77, respectively.

To summarize, while our composite feature spaces especially the CF3 indeed further improve the effectiveness of ASAG, the improvement from *AQS* is relatively smaller than the improvement of *AQS* from *Ans* alone.

5.3.2 Comparing the six ML classifiers in Stage 3

Figure 6 shows that NB falls behind other classifiers with a large margin of 18% on every evaluation metrics except on recall. This suggests that our dataset does not meet the assumption of NB that all the state features are conditionally independent given the output label. Thus, while NB is commonly used in many NLP-related task, it may not be a proper classifier for the task of ASAG. Note that further research is needed to verify this conclusion.

Overall, Fig. 6 shows that DBN generates the best four of five measures (those marked '∗▲') and SVM again has the best precision.

Similar as in previous two stages, we sort the six classifiers in the order of their performance, which is determined by the best performance among the feature spaces in Stage 3. Like in Stage 1 and Stage 2, the same order is kept for accuracy and AUC:
$NB < LR < DT < ANN < SVM < DBN$;
On precision (the same as in Stage 1):
$NB < LR < DT < ANN < DBN < SVM$;
on recall:
$LR < DT < ANN < NB < SVM < DBN$;
and on F-measure:
$NB < LR < DT < ANN < SVM < DBN$.

To summarize, across three stages, DBN outperforms the other five classic ML classifiers on four measures: accuracy, AUC, recall and F-measure, and on precision, SVM performs the best, while DBN is the second. In order to compare the performance among six ML classifiers, Wilcoxon signed rank tests were conducted on resulted values of accuracy, AUC, precision, recall and F-measure across the nine feature space: A, Q, S, AQ, AS, AQS, CF1, CF2 and CF3. Results showed that both SVM and DBN significantly outperformed the four classic ML classifiers: NB, LR, DT and

ANN on all evaluation metrics; between SVM and DBN, while SVM achieves the better precision than DBN, $Z = 2.33$, $p = 0.020$, DBN is significantly better than SVM on AUC $Z = 2.65$, $p = 0.008$, recall $Z = 2.54$, $p = 0.011$, and F-measure $Z = 2.31$, $p = 0.021$, respectively. Thus we can conclude that applying DBN to the task of ASAG is productive. By making use of pre-training stage, DBN is more effective to learn the latent connections among different feature resources. Additionally, among the five classic ML classifiers, SVM performs the best. In particular, in Stage 3 when the feature space is more informative, the performance of SVM is even very close to the performance of DBN. When comparing across the feature spaces, throughout the three stages, the five measures have been greatly improved. Overall, from the conventional *Ans* model in Stage 1 to the CF3 in Stage 3, it increases from 75% to 85% on accuracy, from 79% to 89% on AUC and from 60% to 83% on F-measure, respectively.

6 Conclusion

ASAG is essential to various learning environments. In a typical classroom scenario, students normally get feedback directly from instructors. However, this is labor-intensive and does not scale well to large classes (Burrows and D'Souza 2005), which is particularly true for massive open online courses (MOOCs) where the growing demand far outstrippes available opportunities. In 2015, for example, more than 35 million learners were enrolled in some 4,200 MOOCs offered worldwide [1]. By 2016 this number had exploded to 58 million over 6,850 offerings[2]. This dramatic increase in demand has in turn fueled prominent interest in developing robust automatic grading systems.

Conventional ASAG methods are answer-based since they primarily build models based on student answers using natural language processing techniques. However, student short answers are often too short to provide sufficient lexical features for fully taking advantage of the power of natural language analysis (Pulman and Sukkarieh 2005; Pérez-Marín et al. 2009). On the other hand, even though the ML approaches have been more and more popular in this field, their effectiveness has never been thoroughly investigated and compared for the task of ASAG.

In this work we aim to improve the performance of ASAG by exploring both effective feature spaces and efficient ML approaches. First we incorporate the information about questions and student knowledge levels into the conventional answer-based model. To do so, we construct the answer (*Ans*) model, question model (*Ques*) and student model (*Stu*). We explore the three primary models individually and in combined and composite fashions. Specifically we apply Gaussian mixture model (GMM) and Cartesian product (CP) to compose features. Second, we evaluate different feature spaces on six ML classifiers including five classical ones and a popular deep learning model, deep belief networks (DBN).

[1] https://www.class-central.com/report/moocs-2015-stats/.

[2] https://www.class-central.com/report/moocs-2016-stats/.

The proposed grading framework is evaluated on a training corpus from a natural language physics ITS, Cordillera. Experimental studies show that the introduction of *Ques* and *Stu* models is capable of enhancing the grading performance when they are incorporated into the *Ans* model. In particular the most representative feature space is reached when combining the three primary models together with composite features, including clustering on the *Stu* model (SC) and CP between the *Ques* model and SC. Comparative analysis demonstrates the superiority of DBN on grading short answers across all three stages. DBN outperforms all five classic ML classifiers on accuracy, AUC, precision and F-measure. Only on recall, DBN performs slightly worse than SVM.

Despite its great promises, several aspects of this work can be further improved. First, the robustness of the proposed framework is only evaluated on one dataset, which might make the conclusion biased toward the specific domain and the specific student group. Evaluations on additional large-scale datasets are needed to further validate the applicability and scalability of the proposed framework. When this work was conducted, it was very hard to get access to such type of datasets because of privacy concerns. Currently, we are investigating on publicly available datasets, such as Texas (Mohler et al. 2011) or CoMiC (Meurers et al. 2011) and to evaluate our proposed framework on their data. Second, we did not explore full combinations among different resources when constructing feature spaces. We only add features to the winner model in the previous stage, leading to the overlapping information in the same feature space. Third, we employed a Q-matrix where the KCs for each question are manually specified by experts. Though accurate, it is still labor extensive. We are working on generating and tuning Q-matrix automatically based upon a dataset, as Barnes (2005) and Tatsuoka (1983) have investigated. In the future, instead of only predicting correct or incorrect, we will evaluate whether our ASAG framework can identify different categories of incorrect answers. For example, we can explore whether an incorrect answer is a misconception, a neglectful error or an incomplete answer. An effective personalized learning environment should provide different remediation for different types of errors.

Overall, our results suggest the great potentials for including additional information resources, such as *Ques* and *Stu* into the feature space for the task of ASAG. They can greatly facilitate a personalized and effective grading in a large scale of education system. Also, with the help of more advanced ML approaches, we may expect the grading model to understand student answers more accurately.

Acknowledgements This research was supported by the NSF Grants #1432156: 'Educational Data Mining for Individualized Instruction in STEM Learning Environments', #1651909: 'CAREER: Improving Adaptive Decision Making in Interactive Learning Environment', #1660878 'MetaDash: A Teacher Dashboard Informed by Real-Time Multichannel Self-Regulated Learning Data' and #1726550: 'Integrated Data-driven Technologies for Individualized Instruction in STEM Learning Environments'.

References

An, X., Yung, Y.-F.: Item response theory: what it is and how you can use the IRT procedure to apply it. SAS Institute Inc. SAS364-2014, (2014)

Anderson, R.C., Biddle, W.B.: On asking people questions about what they are reading. Psychol. Learn. Motiv. **9**, 89–132 (1975)
Attali, Y., Burstein, J.: Automated essay scoring with e-rater® v.2. J. Technol. Learn. Assess. **4**(3) (2006)
Baker, F.B., Kim, S.-H.: Item Response Theory: Parameter Estimation Techniques. CRC Press, Boca Raton (2004)
Barnes, T.: The q-matrix method: mining student response data for knowledge. In: American Association for Artificial Intelligence 2005 Educational Data Mining Workshop (2005)
Basu, S., Jacobs, C., Vanderwende, L.: Powergrading: a clustering approach to amplify human effort for short answer grading. Trans. Assoc. Comput. Linguist. **1**, 391–402 (2013)
Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al.: Greedy layer-wise training of deep networks. Adv. Neural Inf. Process. Syst. **153–160**, (2007)
Bernsen, N.O., Dybkjær, H., Dybkjær, L.: Designing Interactive Speech Systems: From First Ideas to User Testing. Springer, Berlin (2012)
Burrows, S., D'Souza, D.: Management of teaching in a complex setting. In: Proceedings of the 2nd Melbourne computing education conventicle, pp. 1–8 (2005)
Burrows, S., Gurevych, I., Stein, B.: The eras and trends of automatic short answer grading. Int. J. Artif. Intell. Educ. **25**(1), 60–117 (2015)
Burstein, J., Leacock, C., Swartz, R.: Automated evaluation of essays and short answers (2001)
Carolyn, R.O.S.E.: Tools for authoring a dialogue agent that participates in learning studies. Artif. Intell. Educ. Building Technol. Rich Learn. Contexts Work **158**, 43 (2007)
Carterette, B., Bennett, P.N., Chickering, D.M., Dumais, S.T.: Here or there. In: European Conference on Information Retrieval, pp. 16–27. Springer (2008)
Chi, M., VanLehn, K., Litman, D., Jordan, P.: Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. User Model. User Adapt. Interact. **21**, 137–180 (2011)
Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Model. User Adapt. Interact. **4**(4), 253–278 (1994)
Dzikovska, M.O., Moore, J.D., Steinhauser, N., Campbell, G., Farrow, E., Callaway, C.B.: Beetle ii: a system for tutoring and computational linguistics experimentation. In: Proceedings of the ACL 2010 System Demonstrations, pp 13–18. Association for Computational Linguistics (2010)
Dzikovska, M.O., Farrow, E., Moore, J.D.: Improving interpretation robustness in a tutorial dialogue system. In: Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications, pp. 293–299 (2013)
Eddy, S.R.: Hidden markov models. Curr. Opin. Struct. Biol. **6**(3), 361–365 (1996)
Graesser, A.C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R.: Tutoring research group, et al. autotutor: a simulation of a human tutor. Cogn. Syst. Res. **1**(1), 35–51 (1999)
Graesser, A. C., Penumatsa, P., Ventura, M., Cai, Z., Hu, X.: Using lsa in autotutor: learning through mixed initiative dialogue in natural language. Handbook of latent semantic analysis, pp. 243–262 (2007)
Hasanah, U., Permanasari, A.E., Kusumawardani, S.S., Pribadi, F.S.: A review of an information extraction technique approach for automatic short answer grading. In: International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE), pp. 192–196. IEEE (2016)
Higgins, D., Burstein, J., Marcu, D., Gentile, C.: Evaluating multiple aspects of coherence in student essays. In: HLT-NAACL (2004)
Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Comput. **14**, 1771–1800 (2002)
Hou, W.-J., Tsao, J.-H., Li, S.-Y., Chen, L.: Automatic assessment of students' free-text answers with support vector machines. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 235–243. Springer (2010)
Huang, G.B., Lee, H., Learned-Miller, E.: Learning hierarchical representations for face verification with convolutional deep belief networks. In: CVPR (2012)
Jia, X., Li, K., Li, X., Zhang, A.: A novel semi-supervised deep learning framework for affective state recognition on eeg signals. In: 2014 IEEE International Conference on Bioinformatics and Bioengineering (BIBE), pp. 30–37. IEEE (2014)
Jia, X., Wang, A., Li, X., Xun, G., Xu, W., Zhang, A.: Multi-modal learning for video recommendation based on mobile application usage. In: 2015 IEEE International Conference on Big Data (Big Data), pp. 837–842. IEEE (2015)

Jia, X., Khandelwal, A., Nayak, G., Gerber, J., Carlson, K., West, P., Kumar, V.: Incremental dual-memory lSTM in land cover prediction. In: Proceedings of the 23rd KDD, pp 867–876. ACM (2017)
Jordan, P.W., Makatchev, M., Pappuswamy, U., VanLehn, K., Albacete, P.L.: A natural language tutorial dialogue system for physics. In: FLAIRS Conference, pp 521–526 (2006)
Karpicke, J.D., Roediger, H.L.: The critical importance of retrieval for learning. Science **319**(5865), 966–968 (2008)
Kim, Y.-J., Chi, M.: Temporal belief memory: Imputing missing data during RNN training. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden, pp. 2326–2332 (2018)
Klein, R., Kyrilov, A., Tokman, M.: Automated assessment of short free-text responses in computer science using latent semantic analysis. In: Proceedings of the 16th ITiCSE, pp. 158–162. ACM (2011)
Lalor, J.P., Wu, H., Yu, H.: Building an evaluation scale using item response theory. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing, vol. 2016, pp. 648. NIH Public Access (2016)
Leacock, C., Chodorow, M.: C-rater: automated scoring of short-answer questions. Comput. Humanit. **37**(4), 389–405 (2003)
Li, Xiaoyi, Jia, Xiaowei, Xun, Guangxu, Zhang, Aidong: Improving eeg feature learning via synchronized facial video. In: 2015 IEEE International Conference on Big Data (Big Data), pages 843–848. IEEE, (2015)
Lin, C., Chi, M.: Intervention-BKT: incorporating instructional interventions into bayesian knowledge tracing. In: Intelligent Tutoring Systems—13th International Conference, ITS 2016, Zagreb, Croatia, June 7–10, 2016. Proceedings, pp. 208–218 (2016)
Lin, C., Chi, M.: A comparisons of bkt, RNN and LSTM for learning gain prediction. In: Artificial intelligence in education—18th International Conference, AIED 2017, Wuhan, China, June 28–July 1, 2017, Proceedings, pp. 536–539 (2017)
Lin, C., Shen, S., Chi, M.: Incorporating student response time and tutor instructional interventions into student modeling. In: Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP 2016, Halifax, NS, Canada, July 13–17, 2016, pp 157–161 (2016)
Lin, C., Zhang, Y., Ivy, J.S., Capan, M., Arnold, R., Huddleston, J.M., Chi, M.: Early diagnosis and prediction of sepsis shock by combining static and dynamic information using convolutional-lSTM. In: IEEE International Conference on Healthcare Informatics, ICHI 2018, New York City, NY, USA, June 4–7, 2018, pp. 219–228 (2018)
Litman, D.J., Silliman, S.: Itspoke: an intelligent tutoring spoken dialogue system. In: Demonstration papers at HLT-NAACL 2004, pp 5–8. Association for Computational Linguistics (2004)
Luaces, O., Díez, J., Alonso-Betanzos, A., Troncoso, A., Bahamonde, A.: A factorization approach to evaluate open-response assignments in moocs using preference learning on peer assessments. Knowl. Based Syst. **85**, 322–328 (2015)
Luaces, O., Díez, J., Alonso-Betanzos, A., Troncoso, A., Bahamonde, A.: Content-based methods in peer assessment of open-response questions to grade students as authors and as graders. Knowl. Based Syst. **117**, 79–87 (2017)
Madnani, N., Burstein, J., Sabatini, J., O'Reilly, T.: Automated scoring of a summary writing task designed to measure reading comprehension, vol. 163. In: NAACL/HLT 2013 (2013)
Magnini, B., Rodríguez, M.P., Strapparava, C., Gliozzo, A., Cubero, E.A., Pérez, D.: About the effects of combining latent semantic analysis with natural language processing techniques for free-text assessment. Rev. Signos Estud. Lingüíst. **59**, 325–343 (2005)
Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing, vol. 999. MIT Press, Cambridge (1999)
Marcu, D.: The Theory and Practice of Discourse Parsing and Summarization. MIT press, Cambridge (2000)
Mason, O., Grove-Stephensen, I. : Automated free text marking with paperless school (2002)
Meurers, D., Ziai, R., Ott, N., Kopp, J.: Evaluating answers to reading comprehension questions in context: results for german and the role of information structure. In: Proceedings of the TextInfer 2011 Workshop on Textual Entailment, pp. 1–9 (2011)
Mitchell, T., Russell, T., Broomhead, P., Aldridge, N.: Towards robust computerised marking of free-text responses (2002)
Mohamed, A.-R., Dahl, G., Hinton, G.: Deep belief networks for phone recognition. In: NIPs (2009)
Mohamed, A.-R., Dahl, G.E., Hinton, G.: Acoustic modeling using deep belief networks. IEEE Trans. Audio Speech Lang. Process. **20**(1), 14–22 (2012)

Mohler, M., Mihalcea, R.: Text-to-text semantic similarity for automatic short answer grading. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics (2009)
Mohler, M., Bunescu, R., Mihalcea, R.: Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 752–762. Association for Computational Linguistics (2011)
Pérez, D.: Automatic evaluation of user's short essays by using statistical and shallow natural language processing techniques. Advanced Studies Diploma (Escuela Politécnica Superior, Universidad Autónoma de Madrid) (2004)
Pérez-Marín, D., Pascual-Nieto, I., Rodríguez, P.: Computer-assisted assessment of free-text answers. Knowl. Eng. Rev. **24**(04), 353–374 (2009)
Pulman, S.G., Sukkarieh, J.Z.: Automatic short answer marking. In: Proceedings of the 2nd Workshop on Building Educational Applications Using NLP
Raman, K., Joachims, T.: Methods for ordinal peer grading. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1037–1046. ACM (2014)
Raman, K., Joachims, T.: Bayesian ordinal peer grading. In: Proceedings of the 2nd (2015) ACM Conference on Learning@ Scale, pp. 149–156. ACM (2015)
Rodrigues, F., Oliveira, P.: A system for formative assessment and monitoring of students' progress. Comput. Educ. **76**, 30–41 (2014)
Sima, D., Schmuck, B., Szöllősi, S., Miklós, Á.: Intelligent short text assessment in emax. In: Towards Intelligent Engineering and Information Technology, pp. 435–445. Springer (2009)
Tatsuoka, K.: Rule space: an approach for dealing with misconceptions based on item response theory. J. Educ. Meas. **20**(4), 345–354 (1983)
Thomson, D., Mitrovic, A.: Towards a negotiable student model for constraint-based ITSS (2009)
VanLehn, K., Jordan, P.W., Litman, D.J.: Developing pedagogically effective tutorial dialogue tactics: experiments and a testbed. In: SLaTE. Citeseer (2007)
Vanlehn, K.: The behavior of tutoring systems. Int. J. Artif. Intell. Educ. **16**(3), 227–265 (2006)
Zhang, Y., Lin, C., Chi, M., Ivy, J., Capan, M., Huddleston, J.M.: LSTM for septic shock: adding unreliable labels to reliable predictions. In: 2017 IEEE International Conference on Big Data (Big Data), pp 1233–1242. IEEE (2017)
Zhang, Y., Yang, X., Ivy, J.S., Chi, M.: ATTAIN: attention-based time-aware LSTM networks for disease progression modeling. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10–16, 2019, pp. 4369–4375 (2019)

Yuan Zhang is currently working at Apple as a data scientist specialized in natural language processing. She obtained her PhD degree in Computer Science from North Carolina State University. Her speciality is applying deep learning techniques to the fields of education and health care. She had extensive industrial experience at Apple, Mayo Clinic, eBay and IBM.

Chen Lin is a research staff member in IBM Research. She received her PhD in Computer Science at North Carolina State University. Her research has focused on applying data mining, machine learning and deep learning to solve problems in various domains such as education, health care, IT services and cloud management.

Min Chi is an Associate Professor of Computer Science at North Carolina State University. Her research focuses on human-centered machine learning and data mining. Dr. Chi received her MS and PhD degrees in the Intelligent Systems Program at the University of Pittsburgh in 2006 and 2009, respectively. She was a Postdoctoral Fellow in the Machine Learning Department at Carnegie Mellon University and at the Human Sciences and Technologies Advanced Research Institute at Stanford University.