CU : Computational Assessment of Short Free Text Answers - A Tool for Evaluating Students' Understanding

Ifeyinwa Okoye
Institute of Cognitive Science
Dept. of Computer Science
University of Colorado
Boulder, CO 80309, USA
okoye@colorado.edu

Steven Bethard
Institute of Cognitive Science
Dept. of Computer Science
University of Colorado
Boulder, CO 80309, USA
bethard@colorado.edu

Tamara Sumner
Institute of Cognitive Science
Dept. of Computer Science
University of Colorado
Boulder, CO 80309, USA
sumner@colorado.edu

Abstract

Assessing student understanding by evaluating their free text answers to posed questions is a very important task. However, manually, it is time-consuming and computationally, it is difficult. This paper details our shallow NLP approach to computationally assessing student free text answers when a reference answer is provided. For four out of the five test sets, our system achieved an overall accuracy above the median and mean.

1 Introduction

Assessing student understanding is one of the holy grails of education (Redecker et al., 2012). If we (teachers, tutors, intelligent tutors, potential employers, parents and school administrators) know what and how much a student knows, then we know what the student still needs to learn. And then, can efficiently and effectively educate the student. However, the task of assessing what exactly a student understands about a particular topic can be expensive, difficult and subjective.

Using multiple choice questionnaires is one of the most prevalent forms of assessing student understanding because it is easy and fast, both manually and computationally. However there has been a lot of pushback from educators about the validity of results gotten from multiple choice questionnaires.

Assessing student understanding by evaluating student free text answers either written or spoken is one of the preferred alternatives to multiple choice questionnaires. As an assessment tool, free text answers can illuminate what and how much a student knows since the student is forced to recall terms and make connections between those terms rather than just picking one out of several options. However, assessing free text answers manually is tedious, expensive and time-consuming, hence the search for a computational option.

There are three main issues that can limit the computational approach and corresponding performance when assessing free text answers: (1) the unit of assessment, (2) the reference and (3) the level of assessment. The unit of assessment can be words, facets, phrases, sentences, short answers or essays. The reference is the correct answer and what is being compared to the student answer. Most researchers generate the reference manually (Noorbehbahani and Kardan, 2011; Graesser et al., 2004) but some have focused on automatically generating the reference (Ahmad, 2009). The level of assessment can be coarse with 2 categories such as correct and incorrect or more finer-grained with up to 19 categories as in (Ahmad, 2009). In general, the finer-grained assessments are more difficult to assess.

2 The Student Response Analysis Task

The student response analysis task was posed as follows: Given a question, a known correct/reference answer and a 1 or 2 sentence student answer, classify the student answer into two, three or five categories. The two categories were *correct* and *incorrect*; the three categories were *correct*, *contradictory* and *incorrect*; while the five categories were *correct*, *partially correct but incomplete*, *contradictory*, *irrelevant* and *not in the domain* (Dzikovska et al., 2013).

We chose to work on the 2-way response task only

*Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 603–607, Atlanta, Georgia, June 14-15, 2013.

because for our application, we need to simply know if a student answer is correct or incorrect. Our application is an interactive essay-based personalized learning environment (Bethard et al., 2012).

The overarching goal of our application is to create a scalable online service that recommends resources to users based on the their conceptual understanding expressed in an essay or short answer form. Our application automatically constructs a domain knowledge base from digital library resources and identifies the core concepts in the domain knowledge base.It detects flaws and gaps in users' science knowledge and recommends digital library resources to address users' misconceptions and knowledge gaps. The gaps are detected by identifying the core concepts which the user has not discussed. The flaws (incorrect understanding/misconceptions) are currently being identified by a process of (1) segmenting a student essay into sentences, (2) aligning the student sentence to a sentence in the domain knowledge base and (3) using the system we developed for the student response analysis task to determine if the student sentence is correct or incorrect.

The development of our misconception detection algorithm has been limited by the alignment task. However, with the data set from the student response analysis task containing correct alignments, we hope to be able to use it to make improvements to our misconception detection algorithm. We discuss our current misconception detection system below.

3 System Description

Our system mainly exploits shallow NLP techniques, in particular text overlap, to see how much we can gain from using a simple system and how much more some more semantic features could add to the simple system. Although we have access to the question which a 1-2 sentence student answer corresponds to, we chose not to use that in our system because in our application we do not have access to that information. We were trying to build a system that would work in our current essay-based application.

Some of the student answers in the dataset have a particular reference answer which they match. However, we do not make use of this information in our system either. We assume that for a particular question, all the corresponding reference answers can be used to determine the correctness of any of the student answers.

3.1 Features

The features we use are:

1. **CosineSimilarity** : This is the average cosine similarity (Jurafsky and James, 2000) between a student answer vector and all the corresponding reference answer vectors. The vectors are based on word counts. The words were lowercased and included stopwords and punctuations.

2. **CosineSimilarityNormalized** : This is the average cosine similarity between a student answer vector and all the corresponding reference answer vectors, with the word counts within the vectors divided by the word counts in Gigaword, a background corpus. We divided the raw counts by the counts in Gigaword to ensure that punctuations, stopwords and other non-discriminatory words do not artificially increase the cosine similarity.

3. **UnigramRefStudent** : This is the average unigram coverage of the reference answers by a student answer. To calculate this, the student answer and all the corresponding reference answers are tokenized into unigrams. Next, for each reference answer, we count the number of unigrams in the reference answer that are contained in the student answer and divide it by the number of unigrams in the reference answer. The value we get for this feature, is the average over all the reference answers.

4. **UnigramStudentRef** : This is the average unigram coverage of the student answer by the reference answers. To calculate this, the student answer and all the corresponding reference answers are tokenized into unigrams. Next, for each reference answer, we count the number of unigrams in the student answer that are contained in the reference answer and divide it by the number of unigrams in the student answer. The value we get for this feature, is the average over all the reference answers.

5. **BigramRefStudent** : This is similar to the UnigramRefStudent feature, but using bigrams.

6. **BigramStudentRef** : This is similar to the UnigramStudentRef feature, but using bigrams.

7. **LemmaRefStudent** : This is similar to the UnigramRefStudent feature, but in this case, the lemmas are used in place of words.

8. **LemmaStudentRef** : This is similar to the UnigramStudentRef feature, but in this case, the lemmas are used in place of words.

9. **UnigramPosRefStudent** : This is similar to the UnigramRefStudent feature, but we use part-of-speech unigrams for this feature in place of word unigrams.

10. **UnigramPosStudentRef** : This is similar to the UnigramStudentRef feature, but we use part-of-speech unigrams for this feature in place of word unigrams.

11. **BigramPosRefStudent** : This is similar to the BigramRefStudent feature, but we use part-of-speech bigrams for this feature in place of word unigrams.

12. **BigramPosStudentRef** : This is similar to the BigramStudentRef feature, but we use part-of-speech bigrams for this feature in place of word unigrams.

3.2 Implementation

We used the ClearTK (Ogren et al., 2008) toolkit within Eclipse to extract features from the student and reference sentences. We trained a LibSVM (Chang and Lin, 2011) binary classifier to classify a feature vector into two classes, correct or incorrect. We used the default parameters for LibSVM except for the cost parameter, for which we tried different values. However, the default value of 1 gave us the best result on the training set. Our two runs/systems are essentially the same system but with a cost parameter of 1 and 10.

4 Results

The Student Response Analysis Task overall result can be found in the Task description paper (Dzikovska et al., 2013). The CU system achieved a ranking of above the mean and median for four of the five different test sets. We perfomed below the mean and median on the sciEntsBank unseen answers. The accuracy result for the test data is shown in Table 4. The results on our training data and a breakdown of the contribution of each feature is shown in Table 5. In Table 5 ALL refers to all the features while ALL-CosineSimilarity is all the features excluding the CosineSimilarity feature.

System	**beetle unseen answers**	**beetle unseen questions**	**sciEntsBank unseen answers**	**sciEntsBank unseen questions**	**sciEntsBank unseen domains**
CU run 1	0.786	0.718	0.656	0.674	0.693
CU run 2	0.784	0.717	0.654	0.671	0.691

Table 1: Overall Accuracy results for CU system on the test Data

5 Discussion

As can be seen from Table 4 and further elaborated on in (Dzikovska et al., 2013), there were two main datasets, Beetle and SciEntsBank. The Beetle data set has multiple reference answer per question while the SciEntsBank has one reference answer per question. Our system did better on the beetle data set than the SciEntsBank data set, both during development and on the final test sets. This leads us to believe that our system will do well when there are multiple reference answers rather than just one.

We analyzed the training data to understand where our system was failing and what we could do to make it better. We tried removing stopwords before constructing the feature vectors but that made the results worse. Here are two examples where removing the stopwords will make it impossible to ascertain the validity of the student answer:

- *It was connected.* becomes *connected*

- *It will work because that is closing the switch.* becomes *work closing switch*

Because the student answers are free text and use pronouns in place of the nouns that were in the question, the stop words are important to provide context.

	Feature Type	**Beetle & sciEnts Bank**
1	ALL	0.703
2	ALL - CosineSimilarity	0.702
3	ALL - CosineSimilarityNormalized	0.700
4	ALL - UnigramRefStudent	0.702
5	ALL - UnigramStudentRef	0.701
6	ALL - BigramRefStudent	0.702
7	ALL - BigramStudentRef	0.699
8	ALL - LemmaRefStudent	0.701
9	ALL - LemmaStudentRef	0.700
10	ALL - UnigramPosRefStudent	0.703
11	ALL - UnigramPosStudentRef	0.703
12	ALL - BigramPosRefStudent	0.702
13	ALL - BigramPosStudentRef	0.702

Table 2: Accuracy results for 5X cross validation on the training data

Currently, we are working on extracting and adding several features that we did not use for the task due to time constraints, to see if they improve our result. Some of the things we are working on are:

1. **Resolving Coreference**
   We will use the current state-of-art coreference system and assume that the question precedes the student answer in a paragraph when resolving coreference.

2. **Compare main predicates**
   The question is how to assign a value to the semantic similarity between the main predicates. If the predicates are *separate* and *connect*, then there should be a way to indicate that the mention of one of them in the reference, precludes the validity of the student answer being correct if it mentions the other. However, we also have to take negation into account here. *not separated* and *connected* should be marked as very similar if not equal. We plan to include the algorithm from the best system in the semantic similarity task to our current system.

3. **Compare main subject and object from a syntactic parse or the numbered arguments in semantic role label arguments**
   We have to resolve coreference for this to work well. And again, we run into the problem of how to assign a semantic similarity value to two words that might not share the same synset in ontologies such as Wordnet.

4. **Optimize parameters and explore other classifiers** Throughout developing and testing our system, we used only the LibSVM classifier and only optimized the cost parameter. However, there might be a different classifier or set of options that can model the data better. We hope to run through most of the classifiers available and see if using a different one, with different options improves our accuracy.

6 Conclusion

We have shown that there is value in using shallow NLP features to judge the validity of free answer text when the reference answers are given. However, looking at the sentences that our system labeled as correct and the gold standard incorrect or vice versa, it is clear that we have to delve into more semantic features if we want our system to be more accurate. We hope to keep working on this task in subsequent years to ensure continuous improvements in systems that can assess student knowledge by evaluating free answer texts. Such systems will be able to give students the formative feedback they need to help them learn better. In addition, such systems will provide teachers, intelligent tutors and administrators with feedback about student knowledge, so as to help them adapt their curriculum, teaching and tutoring methods to better serve students' knowledge needs.

References

Faisal Ahmad. 2009. *Generating conceptually personalized interactions for educational digital libraries using concept maps*. Ph.D. thesis, University of Colorado at Boulder.

Steven Bethard, Haojie Hang, Ifeyinwa Okoye, James H Martin, Md Arafat Sultan, and Tamara Sumner. 2012. Identifying science concepts and student misconceptions in an interactive essay writing tutor. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 12–21. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Myroslava O. Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. 2013. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In **SEM 2013: The First Joint Conference on Lexical and Computational Semantics*, Atlanta, Georgia, USA, 13-14 June. Association for Computational Linguistics.

Arthur Graesser, Shulan Lu, George Jackson, Heather Mitchell, Mathew Ventura, Andrew Olney, and Max Louwerse. 2004. AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods*, 36:180–192.

Daniel Jurafsky and H James. 2000. Speech and language processing an introduction to natural language processing, computational linguistics, and speech.

F Noorbehbahani and AA Kardan. 2011. The automatic assessment of free text answers using a modified bleu algorithm. *Computers & Education*, 56(2):337–345.

Philip V Ogren, Philipp G Wetzler, and Steven J Bethard. 2008. Cleartk: A uima toolkit for statistical natural language processing. *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP*, page 32.

Christine Redecker, Yves Punie, and Anusca Ferrari. 2012. eassessment for 21st century learning and skills. In *21st Century Learning for 21st Century Skills*, pages 292–305. Springer.