Fast and Easy Short Answer Grading with High Accuracy

Md Arafat Sultan **Cristobal Salazar** **Tamara Sumner**
Institute of Cognitive Science
Department of Computer Science
University of Colorado, Boulder, CO
{arafat.sultan,crsa7687,sumner}@colorado.edu

Abstract

We present a fast, simple, and high-accuracy short answer grading system. Given a short-answer question and its correct answer, key measures of the correctness of a student response can be derived from its semantic similarity with the correct answer. Our supervised model (1) utilizes recent advances in the identification of short-text similarity, and (2) augments text similarity features with key grading-specific constructs. We present experimental results where our model demonstrates top performance on multiple benchmarks.

1 Introduction

Short-answer questions are a useful device for eliciting student understanding of specific concepts in a subject domain. Numerous automated graders have been proposed for short answers based on their semantic similarity with one or more expert-provided correct answers (Mohler et al., 2011; Heilman and Madnani, 2013; Ramachandran et al., 2015). From an application perspective, these systems vary considerably along a set of key dimensions: amount of human effort involved, accuracy, speed, and ease of implementation. We explore a design that seeks to optimize performance along all these dimensions.

Systems developed for the more general task of short-text semantic similarity provide a good starting point for such a design. Major progress has been made in this task in recent years, due primarily to the SemEval Semantic Textual Similarity (STS) task (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015). However, the utility of top STS systems has remained largely unexplored in the context of short answer grading. We seek to bridge this gap by adopting the feature set of the best performing STS system at SemEval-2015 (Sultan et al., 2015). Besides high accuracy, this system also has a simple design and fast runtime.

Textual similarity alone, however, is inadequate as a measure of answer correctness. For example, while the Sultan et al. (2015) system makes the general assumption that all content words[1] contribute equally to the meaning of a sentence, domain keywords (e.g., "mutation" for biological evolution) are clearly more significant than arbitrary content words (e.g., "consideration") for academic text. As another example, *question demoting* (Mohler et al., 2011) proposes discarding words that are present in the question text as a preprocessing step for grading. We augment our generic text similarity features with such grading-specific measures.

We train supervised models with our final feature set; in two different grading tasks, these models demonstrate significant performance improvement over the state of the art. In summary, our contribution is a fast, simple, and high-performance short answer grading system which we also release as open-source software at: https://github.com/ma-sultan/short-answer-grader.

2 Related Work

A comprehensive review of automatic short answer grading can be found in (Burrows et al., 2015). Here

[1] meaning-bearing words (e.g., nouns and main verbs), as opposed to function words that play predominantly syntactic roles in a sentence (e.g., auxiliary verbs and prepositions).

Proceedings of NAACL-HLT 2016, pages 1070–1075,
San Diego, California, June 12-17, 2016.

we briefly discuss closely related work.

Early short answer grading work relied on patterns (e.g., regular expressions) manually extracted from expert-provided reference answers (Mitchell et al., 2002; Sukkarieh et al., 2004; Nielsen et al., 2009). Such patterns encode key concepts representative of good answers. Use of manually designed patterns continues to this day, e.g., in (Tandalla, 2012), the winning system at the ASAP answer scoring contest.[2] This is a step requiring human intervention that natural language processing can help to eliminate. Ramachandran et al. (2015) propose a mechanism to automate the extraction of patterns from the reference answer as well as high-scoring student answers. We adopt the simpler notion of semantic alignment to avoid explicitly generating complicated patterns altogether.

Direct semantic matching (as opposed to pattern generation) has been explored in early work like (Leacock and Chodorow, 2003). With advances in NLP techniques, this approach has gained popularity over time (Mohler et al., 2009; Mohler et al., 2011; Heilman and Madnani, 2013; Jimenez et al., 2013). Such systems typically use a large set of similarity measures as features for a supervised learning model. Features range from string similarity measures like word and character n-gram overlap to deeper semantic similarity measures based on resources like WordNet and distributional methods like latent semantic analysis (LSA). However, a large feature set contributes to higher system runtime and implementation difficulty. While following this generic framework, we seek to improve on these criteria by employing a minimal set of core similarity features adopted from (Sultan et al., 2015). Our features also yield higher accuracy by utilizing more recent measures of lexical similarity (Ganitkevitch et al., 2013; Baroni et al., 2014), which have been shown to outperform traditional resources and methods like WordNet and LSA.

Short-text semantic similarity has seen major progress in recent times, due largely to the SemEval Semantic Textual Similarity (STS) task (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015). STS systems can serve as a source of important new features and design elements for automatic short answer graders (Bär et al., 2012; Han et al., 2013; Lynum et al., 2014; Hänig et al., 2015).

Surprisingly, few existing grading systems utilize simple and computationally inexpensive grading-specific techniques like question demoting (Mohler et al., 2011) and term weighting. Our model augments the similarity features using these techniques.

3 Method

Following feature extraction, our system trains a supervised model for grading. As we discuss in Section 4, this can be a regressor or a classifier depending on the task. This section describes our features; specifics of the models are given in Section 4.

3.1 Features

3.1.1 Text Similarity

Given reference answer $R = (r_1, ..., r_n)$ and student response $S = (s_1, ..., s_m)$ (where each r and s is a word token), we compute three generic text similarity features.

Alignment. This feature measures the proportion of content words in R and S that have a semantically similar word in the other sentence. Such pairs are identified using a word aligner (Sultan et al., 2014). The semantic similarity of a word pair (r_i, s_j) is a weighted sum of their lexical and contextual similarities. A paraphrase database (PPDB, Ganitkevitch et al. (2013)) identifies lexically similar word pairs; contextual similarity is computed as average lexical similarity in (1) dependencies of r_i in R and s_j in S, and (2) content words in [-3, 3] windows around r_i in R and s_j in S. Lexical similarity scores of pairs in PPDB as well as weights of word and contextual similarities are optimized on an alignment dataset (Brockett, 2007).

To avoid penalizing long student responses that still contain the correct answer, we also employ a second version of this feature: the proportion of aligned content words only in R. We will refer to this feature as *coverage* of the reference answer's content by the student response.

Semantic Vector Similarity. This feature employs off-the-shelf word embeddings.[3] A sentence-level semantic vector is computed for each input

[2] https://www.kaggle.com/c/asap-sas/

[3] 400-dimensional word embeddings reported by Baroni et al. (2014).

sentence as the sum of its content word embeddings (lemmatized). The cosine similarity between the R and S vectors is then used as a feature. While the alignment features distinguish only between paraphrases and non-paraphrases, this feature enables integration of finer-grained lexical similarity measures between related concepts (e.g., *cell* and *organism*).

3.1.2 Question Demoting

We recompute each of the above similarity features after removing words that appear in the question text from both the reference answer and the student response. The objective is to avoid rewarding a student response for repeating question words.

3.1.3 Term Weighting

To be able to distinguish between domain keywords and arbitrary content words, in our next set of features we assign a weight to every content word in the reference and the student answer based on a variant of *tf-idf*. While general short-text similarity models typically use only *idf* (inverse document frequency) to penalize general words, the domain-specific nature of answer grading also enables the application of a *tf* (term frequency) measure.

To fully automate the process for a question and reference answer pair, we identify all content words in the pair. The top ten Wikipedia pages related to these words are retrieved using the Google API. Each page is read along with all linked pages crawled using Scrapy (Myers and McGuffee, 2015). The in-domain term frequency (tf_d) of a word in the answer is then computed by extracting its raw count in this collection of pages. We use the same set of tools to automatically extract Wikipedia pages in 25 different domains such as Art, Mathematics, Religion, and Sport. A total of 14,125 pages are retrieved, occurrences in which are used to compute the *idf* of each word.

We augment our alignment features—both original and question-demoted—with term weights to generate new features. Each word is assigned a weight equal to its $tf_d \times idf$ score. The sum of weights is computed for (1) aligned, and (2) all content words in the reference answer (after question demoting, if applicable). The ratio of these two numbers is then used as a feature. We compute only coverage features (Section 3.1.1) to avoid computing term weights for each student response. Thus the process of crawling and reading the documents is performed once per question; all the student responses can subsequently be graded quickly.

3.1.4 Length Ratio

We use the ratio of the number of words in the student response to that in the reference answer as our final feature. The aim is to roughly capture whether or not the student response contains enough detail.

4 Experiments

We evaluate our features on two grading tasks. The first task, proposed by Mohler et al. (2011), asks to compute a real-valued score for a student response on a scale of 0 to 5. The second task, proposed at SemEval-2013 (Dzikovska et al., 2013), asks to assign a label (e.g., *correct* or *irrelevant*) to a student response that shows how appropriate it is as an answer to the question. Thus from a machine learning perspective, the first is a regression task and the second is a classification task. We use the NLTK stopwords corpus (Bird et al., 2009) to identify function words. Results are discussed below.

4.1 The Mohler et al. (2011) Task

The dataset for this task consists of 80 undergraduate Data Structures questions and 2,273 student responses graded by two human judges. These questions are spread across ten different assignments and two tests, each on a related set of topics (e.g., programming basics, sorting algorithms). A reference answer is provided for each question. Inter-annotator agreement was 58.6% (Pearson's ρ) and .659 (RMSE on a 5-point scale). Average of the two human scores is used as the final gold score for each student answer.

We train a ridge regression model (Scikit-learn (Pedregosa et al., 2011)) for each assignment and test using annotations from the rest as training examples. A *dev* assignment or test is randomly held out for model selection. Out-of-range output scores, if any, are rounded to the nearest in-range integer. Following Mohler et al. (2011), we compute a single Pearson correlation and RMSE score over all student responses from all datasets. Average results across 1000 runs of the system are shown in Table 1. Our

System	Pearson's r	RMSE
tf-idf	.327	1.022
Lesk	.450	1.050
Mohler et al. (2011)	.518	.978
Our Model	**.592**	**.887**

Table 1: Performance on the Mohler et al. (2011) dataset with out-of-domain training. Performances of simpler bag-of-words models are reported by those authors.

System	r	RMSE
Ramachandran et al. (2015)	.61	.86
Our Model	**.63**	**.85**

Table 2: Performance on the Mohler et al. (2011) dataset with in-domain training.

model shows a large and significant performance improvement over the state-of-the-art model of Mohler et al. (two-tailed t-test, $p < .001$). Their system employs a support vector machine that predicts scores using a set of dependency graph alignment and lexical similarity measures. Our features are similar in intent, but are based on latest advances in identification of lexical similarity and monolingual alignment.

Ramachandran et al. (2015) adopt a different setup to evaluate their model on the same dataset. For each assignment/test, they use 80% of the data for training and the rest as test. This setup thus enables in-domain model training. Their system automatically generates regexp patterns intended to capture semantic variations and syntactic structures of good answers. Features derived from match with such patterns as well as term frequencies in the student response are used to train a set of random forest regressors, whose predictions are then combined to output a single score. Results in this setup are shown in Table 2. Again, averaged over 1000 runs, our model performs better on both evaluation metrics. The differences are smaller than before but still statistically significant (two-tailed t-test, $p < .001$).

4.2 The SemEval-2013 Task

Instead of a real-valued score, this task asks to assign one of five labels to a student response: *correct*, *partially correct/incomplete*, *contradictory*, *irrelevant*, and *non-domain* (an answer that contains no domain content). We use the SCIENTSBANK corpus, containing 9,804 answers to 197 questions in 15 science domains. Of these, 3,969 are used for model training and the remaining 5,835 for evaluation. A reference answer is provided for each question.

System	UA	UQ	UD	Wt. Mean
Lexical Overlap	.435	.402	.396	.400
Majority	.260	.239	.249	.249
ETS$_1$	.535	.487	.447	.460
SoftCardinality$_1$	.537	.492	.471	.480
Our Model	**.582**	**.554**	**.545**	**.550**

Table 3: F_1 scores on the SemEval-2013 datasets.

The test set is divided into three subsets with varying degrees of similarity with the training examples. The *Unseen Answers* (UA) dataset consists of responses to questions that are present in the training set. *Unseen Questions* (UQ) contains responses to in-domain but previously unseen questions. Three of the fifteen domains were held out for a final *Unseen Domains* (UD) test set, containing completely out-of-domain question-response pairs. For this task, we train a random forest classifier with 500 trees in Scikit-learn using our feature set.

Table 3 shows the performance of our model (averaged over 100 runs) along with that of top systems[4] at SemEval-2013 (and of simpler baselines). ETS (Heilman and Madnani, 2013) employs a logistic classifier combining lexical and text similarity features. SoftCardinality (Jimenez et al., 2013) employs decision tree bagging with similarity features derived from a set cardinality measure—soft cardinality—of the question, the reference answer, and the student response. These features effectively compute text similarity from commonalities and differences in character n-grams.

Each cell on columns 2–4 of Table 3 shows a weighted F_1-score on a test set computed over the five classes, where the weight of a class is proportional to the number of question-response pairs in that class. The final column shows a similarly weighted mean of scores computed over the three test sets. On each test set, our model outperforms the top-performing models from SemEval (significant at $p < .001$). Its performance also suffers less on out-of-domain test data compared to those models.

[4]Systems with best overall performance on SCIENTSBANK.

Features	Pearson's r	RMSE
All	.592	.887
w/o alignment	.519	.938
w/o embedding	.586	.892
w/o question demoting	.571	.903
w/o term weighting	.590	.889
w/o length ratio	.591	.888

Table 4: Ablation results on the Mohler et al. (2011) dataset.

4.3 Runtime Test

Given parsed input and having stop words removed, the most computationally expensive step in our system is the extraction of alignment features. Each content word pair across the two input sentences is assessed in constant time, giving the feature extraction process (and the whole system) a runtime complexity of $O(n_c \cdot m_c)$, where n_c and m_c are the number of content words in the two sentences. Note that all alignment features can be extracted from a single alignment of the input sentences.

Run on the Mohler et al. dataset (unparsed; about 18 words per sentence on average), our system grades over 33 questions/min on a 2.25GHz core.

4.4 Ablation Study

Table 4 shows the performance of our regression model on the Mohler et al. dataset without different feature subsets. Performance falls with each exclusion, but by far the most without alignment-based features. Features implementing question demoting are the second most useful. Length ratio improves model performance the least.

Surprisingly, term weighting also has a rather small effect on model performance. Further inspection reveals two possible reasons for this. First, many reference answers are very short, only containing words or small phrases that are necessary to answer the question (e.g., "push", "enqueue and dequeue", "by rows"). In such cases, term weighting has little or no effect. Second, we observe that in many cases the key words in a correct answer are either not domain keywords or are unidentifiable using *tf-idf*. Consider the following:

- Question: What is a stack?
- Answer: A data structure that can store elements, which has the property that the last item added will be the first item to be removed (or last-in-first-out).

Important answer words like "last", "added", "first", and "removed" in this example are not domain keywords and/or are too common (across different domains) for a measure like *tf-idf* to work.

5 Conclusions and Future Work

We present a fast, simple, and high-performance short answer grading system. State-of-the-art measures of text similarity are combined with grading-specific constructs to produce top results on multiple benchmarks. There is, however, immense scope for improvement. Subtle factors like differences in modality or polarity might go undetected with coarse text similarity measures. Inclusion of text-level paraphrase and entailment features can help in such cases. Additional term weighting mechanisms are needed to identify important answer words in many cases. Our system provides a simple base model that can be easily extended with new features for more accurate answer grading.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grants EHR/0835393 and EHR/0835381. We thank Lakshmi Ramachandran for clarification of her work.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A Pilot on Semantic Textual Similarity. In *SemEval*.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 Shared Task: Semantic Textual Similarity. In **SEM*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *SemEval*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *SemEval*.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures. In *SemEval*.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't Count, Predict! A Systematic Comparison of Context-Counting vs. Context-Predicting Semantic Vectors. In *ACL*.

Steven Bird, Edward Loper, and Ewan Klein. 2009. Natural Language Processing with Python. O'Reilly Media Inc.

Chris Brockett. 2007. Aligning the RTE 2006 Corpus. Tech Report MSR-TR-2007-77, Microsoft Research.

Steven Burrows, Iryna Gurevych, and Benno Stein. 2015. The Eras and Trends of Automatic Short Answer Grading. *International Journal of Artificial Intelligence in Education*, 25.1.

Myroslava O. Dzikovska, Rodney Nielsen, Chris Brew,Claudia Leacock, Danilo Giampiccolo, Luisa Ben-tivogli, 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 *SemEval*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *NAACL*.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC EBIQUITY-CORE: Semantic Textual Similarity Systems. In **SEM*.

Christian Hänig, Robert Remus, and Xose de la Puente. 2015. ExB Themis: Extensive Feature Extraction from Word Alignments for Semantic Textual Similarity. In *SemEval*.

Michael Heilman and Nitin Madnani. 2013. ETS: Domain Adaptation and Stacking for Short Answer Scoring. In *SemEval*.

Sergio Jimenez, Claudia Becerra, and Alexander Gelbukh. 2013. SOFTCARDINALITY: Hierarchical Text Overlap for Student Response Analysis. In *SemEval*.

Claudia Leacock and Martin Chodorow. 2003. C-rater: Automated Scoring of Short-Answer Questions. *Computers and the Humanities*, 37(04).

André Lynum, Partha Pakray, Björn Gambäck, and Sergio Jimenez. 2014. NTNU: Measuring Semantic Similarity with Sublexical Feature Representations and Soft Cardinality. In *SemEval*.

Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. 2002. Towards Robust Computerised Marking of Free-Text Responses. In *Proceedings of the 6th International Computer Assisted Assessment (CAA) Conference*.

Michael Mohler and Rada Mihalcea. 2009. Text-to-text Semantic Similarity for Automatic Short Answer Grading. In *EACL*.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to Grade Short Answer Questions Using Semantic Similarity Measures and Dependency Graph Alignments. In *ACL*.

Daniel Myers and James W. McGuffee. 2015. Choosing Scrapy. *Computing Sciences in Colleges*, 31(1).

Rodney D. Nielsen, Wayne Ward, and James H. Martin. 2009. Recognizing Entailment in Intelligent Tutoring Systems. *Natural Language Engineering*, 15(04).

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Machine Learning Research*, vol. 12.

Lakshmi Ramachandran, Jian Cheng, and Peter Foltz. 2015. Identifying Patterns For Short Answer Scoring using Graph-based Lexico-Semantic Text Matching. In *SemEval*.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. TakeLab: Systems for Measuring Semantic Text Similarity. In *SemEval*.

Jana Z. Sukkarieh, Stephen G. Pulman and Nicholas Raikes. 2004. Auto-Marking 2: An Update on the UCLES-Oxford University research into using Computational Linguistics to Score Short, Free Text Responses. In *Proceedings of the 30th Annual Conference of the International Association for Educational Assessment*.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association for Computational Linguistics*, 2 (May).

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In *SemEval*.

Louis Tandalla. 2012. Scoring Short Answer Essays. `https://kaggle2.blob.core.windows.net/competitions/kaggle/2959/media/TechnicalMethodsPaper.pdf`.