E2VScorer: An Embedding-based Approach to Evaluate Argument Strength in Student Essays

Aaron Hammond

Massachusetts Institute of Technology
6.806 - Advanced Natural Language Processing
ahammond@mit.edu

Abstract

While automatic essay grading remains a topic of interest in NLP, methods which seek to evaluate argument strength, instead of more formal dimensions, remain rare; furthermore, those described in the literature primarily rely on feature-based classifiers, whose heuristic rules can lose predictive value when applied across different essay sets. We propose a supervised, neural network-based approach, based on word2vec and one of its extensions, doc2vec, to construct a classifier over trained essay and prompt embeddings; this classifier provides a means for scoring essays on the basis of the quality of their argument in answering the relevant prompt without a priori feature extraction or annotation. Our methods significantly outperform a frequency-based baseline, achieving upwards of 75-80% accuracy on unseen student essays drawn from two distinct prompt-response sets.[1]

I. Introduction

Many of the dominant trends in education reform today—standards-based educational evaluation (Common Core); individualized digital instruction (Khan Academy); and Massive Open Online Courses (edX), to name a few—require some degree of automated performance evaluation to be efficient or even feasible at scale. While computerized technologies to evaluate simple, factual responses to multiple choice, true-or-false, and computation-based questions have gained traction—either in the form of machine-gradable answer sheets (Scantron) or computer-adaptive testing (a majority of the GRE)—the same cannot be said for more open-ended questions which require a free-form response. For this class of questions, of which our domain of interest, prompted argumentative essays, is a subset, manual scoring by humans is most often used instead. However, human grading of essays in particular suffers from a number of structural pitfalls which weigh against the enormous value of these assessments in broad evaluation of educational outcomes.

For one, human graders, unlike machines, require wages; this represents a significant cost burden on educational institutions, who may need to hire several graders to achieve the level of accuracy required for even low-stakes evaluations. As a consequence, the number of open-ended questions that can reasonably be included in an assessment drops off quickly at scale. More of concern to those students and teachers and administrators whose educational and vocational outcomes may depend on the results of educational assessments, however, are the regrettably human traits of human graders. These workers may score on the order of hundreds of essays per day with little background in education or commensurate training. To quote a newspaper article on the subject,

> In a matter of minutes, a $10-an-hour temp assigns a score to your child's test, a grade that helps deter-

[1] The code used to construct the models and perform the analysis for this paper is available on github

> mine how money is spent in Washington schools, which courses students take and ... who is denied a high-school diploma.[2]

Especially worrisome is the level of inconsistency in assigned scores—between different graders, between different grading sessions, and even across a single grader's work in a single session – even when a well-defined rubric is provided; the level of disagreement between graders may reach as high as 40% in some cases. Thus, a method to perform automatically such essay scoring in a consistent and accurate manner would be a boon for education reformers.[3]

This lack motivates our investigation. While automated techniques for evaluation of text quality on lexical dimensions such as grammaticality and fluency have been frequently explored in the literature, approaches which probe instead more semantic dimensions of essay quality remain scant. In particular, evaluation of the argument strength of essays written to answer open-ended prompts is seldom explored. Moreover, the methods that are described in the literature oftentimes rely on heuristics which may be appropriate for some essay sets but not others: as a concrete example, function-based labeling of sentences on the basis of presence of string-prefixes like "suggest," "conflict," or "hypothes" may produce meaningful results over essays by college undergraduates [2, 1], but the same is certainly not true for the essays by 8th or 10th graders which compose our corpus.

In this paper, we develop a more general approach which relies on the semantic content of an essay, captured in a single embedding for each essay by the doc2vec model, within the context of the expected semantic content for essays written by students in response to a specific prompt, captured likewise in a single embedding, to assign a score for the strength of its argument over a discrete range. Significant variations in model performance dependent on the topology of our neural networks suggest that our obtained results are not optimal; instead, we seek merely to demonstrate the viability and fitness of our novel approach and to motivate further work along these lines.

[2] Jolayne Houtz writing for the Seattle Times

[3] ibid.

II. Corpus and Instrumentation

One possible reason for the relative dearth of work on evaluation of essay argument strength is the lack of a publicly accessible corpus with explicit score annotations along this dimension. That said, we found adequate analogues in the scoring rubrics for two of the essay sets used in the 2012 Hewlett Foundation Automated Essay Scoring competition on Kaggle. Specifically, we made use of Essay Sets #1 and #2. The (concatenated) prompts for each follow.

> More and more people use computers, but not everyone agrees that this benefits society... Write a letter to your local newspaper in which you state your opinion on the effects computers have on people. Persuade the readers to agree with you.

> Write a persuasive essay to a newspaper reflecting your views on censorship in libraries... Support your position with convincing arguments from your own experience, observations, and/or reading.

Essay Set #1 contained 1785 essay responses composed by 8th graders. The average length of each essay was 350 words, and essays were given a score from 1-6 by two separate graders; "a well-developed response that takes a clear and thoughtful position and provides persuasive support" merited a score of 6, while "an undeveloped response that may take a position but offers no more than very minimal support" earned a score of 1.

Essay Set #2 contained 1800 essay responses composed by 10th graders. The average length of each essay was also 350 words, and essays were scored across two domains: writing applications and language conventions. Only the first is relevant to our investigation. Essays were given a score for this domain from 1-6 by two separate graders; within this domain were included the dimensions of "Ideas and Content," "Organization," "Style," and "Voice."

It is important to note that in neither set was the selected scoring domain dependent on lexical measures; rather, in both cases, the scores awarded were evaluations of the quality of the essay specifically on the basis of the strength of its forwarded argument in answering the relevant prompt. In addition, for each essay, two scores produced by two different human graders were provided; as we will discuss later, we chose to target one of these scores, but use both in order to validate the accuracy of our predictions. This is consistent with the scenario in which our machine grader functions as a third set of "eyes" to evaluate the quality of the scores assigned by the two human graders.

Table 1: *Score Frequencies*

Set	1	2	3	4	5	6
#1	0.01	0.02	0.11	0.52	0.28	0.30
#2	0.01	0.09	0.42	0.43	0.04	0.01
#1 ∪ #2	0.01	0.05	0.27	0.47	0.16	0.05

Within Essay Set #1, the two human graders disagreed in their assigned scores 34.7% of the time; in Essay Set #2, the two disagreed 21.7% of the time. We must view our results in this context: an absolute accuracy (that is, where the score predicted is exactly the score given by the rater whose scores we target during training) above 65.3% on Essay Set #1 or above 78.3% on Essay Set #2 (or over the whole set, above 71.9%) indicates that our machine grader assigns scores more more consistently (with respect to the first grader) than the second human grader does.

The ordering of essays within our dataset was shuffled once to ensure consistency of results and to prevent time-sensitive overfitting of the neural networks. Training and testing were performed on (disjoint) random slices of 80% and 20% of the dataset respectively. We used the Keras wrapper around the TensorFlow platform for construction and training of our neural networks and the gensim implementation of doc2vec to extract essay embeddings.

III. Methods

To evaluate the performance of our two proposed methods, we first construct a baseline grading system. We use the same Most Frequent Baseline used elsewhere in the literature [1]. We also considered implementing either the Learning-based Ong et al. approach [2, 1] or the Persing and Ng feature-rich approach [1] as additional baselines; however, the advertised accuracy gain over the Most Frequent Baseline was only 1.6% for the first and 5.0% for the second [1]. Because our two developed methods immediately yielded more significant gains in accuracy and because implementation of either approach as a baseline would require a mapping between words identified in the heuristics used to their appropriate equivalents in our corpus [1], neither was an appropriate candidate.

For all three methods, training and testing were performed on (disjoint) random slices of 80% and 20% respectively of the entire set of essays; that is, we made no partition between essays from Essay Set #1 and Essay Set #2 in training or testing.

1. Most Frequent Baseline

During training, the frequency distribution of scores is computed over the training set. During testing, we assign a score for each essay equal to the most frequently occurring score per the computed distribution. Based on the frequencies provided in **Table 1**, we expect every essay to be assigned a score of 4 during testing.

Essay2Vec

Our two proposed methods both make use of a single embedding vector to capture the semantic content of each essay; we compute these essay embeddings prior to training. These embeddings are used as ordinary inputs to the neural networks constructed in each approach so as to prevent pollution of the semantic content captured in each essay embedding with information related to the essay's score. We use the doc2vec model developed by Le and Mikolov to learn such a vector representation for each essay.

First, the text of each essay is split according to its whitespace; then, we substitute a STOP token as appropriate to demarcate the end of each sentence. This list of appropriately post processed words is the *document* of the essay. A classifier is then trained on some number of sampled skip-grams within each fixed-length window, taking as inputs the embedding for the document of the essay and a skip gram and producing as outputs the softmax probability that a given word w_i is present in the window, and thus, the document [3]. After a sufficient number of these unsupervised iterations, the embedding for an essay's document becomes a dense vector representation of the semantic content contained therein [3]. Because this semantic content is what interests us, we use this embedding as a vector representation for the essay itself.

We used the gensim implementation of doc2vec with both hierarchical and negative sampling with a window size of 20 words and pruned words that occurred in the corpus fewer than five times; the size of the resulting vocabulary was 9904 words. We trained vector representations over the complete contents of Essay Sets #1-8 to ensure a broadly representational corpus. At the conclusion of this step, we therefore have a relation $essay2vec(essay_i)$ which maps each distinct $essay_i$ to a distinct dense vector representation of some fixed dimension.

Prompt2Vec

To actually predict the score of an essay, given its vector representation as provided by $essay2vec()$, we will construct an additional neural network. The inputs to this neural network will be the fixed vector representation of an essay and the embedding for its associated prompt. The precise topology of the output layer will differ between our two approaches, but in ether case will provide the means to construct a predicted score for the essay, given its associated prompt. Thus, training will consist of stepping over the essays in our training set; for each, we will use the $essay2vec()$ relation to set the value of some number of inputs and set the remaining inputs to the latest values of the embedding of its associated prompt. If our training example is $essay_i$, which was written in response to $prompt_j$, then the inputs to the network will be given by $essay2vec(essay_i)$ and $prompt2vec(prompt_j)$. After forward propagation, we compare the predicted score to the actual score awarded by the grader we target—in practice, we always select the first grader—and perform the appropriate back propagation through the network according to the topology of the output layer so as to minimize error. After back propagation, we update the network's weights and the prompt embedding, $prompt2vec(prompt_j)$. As discussed above, however, we do *not* make any updates to the embedding given by $essay2vec(essay_i)$ because we treat it here as a simple input vector.

We experimented with a range of topologies for the intermediate (hidden) layers in the network and methods of back propagation, but this characterization of the input layer and training procedure remained constant throughout. Visualizations of both network topologies are offered in the Appendix.

2. Categorical Prediction

In our first proposed method, the output layer of the neural network is a simple softmax probability estimation for each of the six *categories* of essay performance where each category represents one of the six possible discrete scores. After forward propagation, we can predict the score for an essay, given its associated prompt, by simply determining the score associated with the largest probability estimate at the output layer. To train the neural network, we set the output target to the appropriate one-hot vector given the real score assigned by the targeted grader and perform the associated back propagation.

3. Absolute Prediction

In our second proposed method, the output layer of the neural network is a single node. After forward propagation, we can predict the score for an essay given its associated prompt by determining which of the six interval-defined "buckets" contains the final output value of the network. These buckets are given by splitting the interval $[0, 1]$ six ways, in ascending order, where the width of a score's bucket is equal to its weight in the frequency distribution of scores in the training set. That is, if the score 1 had frequency 0.2, score 2 had 0.2, score 3 had 0.2, score 4 had 0.2, and scores 5 and 6 both had frequency 0.1, then values in the interval $(-\infty, 0.2)$ would be assigned scores of 1, values in the interval $[0.2, 0.4)$ would be assigned scores of 2, and so on, with values in the interval of $[0.9, \infty)$ receiving a score of 6.

To train with this output layer, we would set the target output value of the network to be in the center of the real score's associated bucket; continuing the prior example, a real score of 2 would translate to a target output value of 0.3.

IV. Evaluation

Given that our goal was to construct an essay scoring model which generates accurate and consistent score predictions and that, as established in the introduction, the scores awarded by the first human grader may be neither entirely consistent nor absolutely accurate, we selected three primary metrics to evaluate the performance of our models.

Absolute Accuracy the frequency with which the predicted score *exactly* matched the real score awarded by the first human grader

Acceptable Accuracy the frequency with which the predicted score matched *either* the real score awarded by the first human grader *or* the real score awarded by the second human grader

Diff>1 the frequency with which the predicted score differed by *more* than one full point from the real score awarded by the first human grader

V. Results

Table 2: *Performance under maximal observed network topology*

	Training Set			Testing Set		
Model	AbA	AcA	D>1	AbA	AcA	D>1
BL	0.483	0.594	0.096	0.441	0.578	0.095
CatP	0.635	0.761	0.014	0.653	0.801	0.020
AbsP	0.605	0.740	0.025	0.641	0.778	0.023

In the table above, we record the performance of each method under the maximal observed network topology. With categorical prediction, we achieved an absolute accuracy of 65.3% and an acceptable accuracy of 80.1%—a gain of slightly more than 20% across both metrics from the baseline. The performance gains achieved by the absolute prediction model were a bit more modest but still represent a significant improvement over the baseline, especially in comparison to the models presented by Persing and Ng [1].

Curiously, for both models under maximal network topologies, the absolute and acceptable accuracies were *higher* on the testing set than the training set. Moreover, this phenomenon occurs frequently across different network topologies, suggesting that something more systemic than a particularly favorable local maxima is to blame. We hypothesize that this effect could be due to the use of Dropout (with $p = 0.5$) at every hidden layer in the network [4]; we reason that, because we apply Dropout only during training, some neuronal circuits necessary for good fits over the training data are missing when we test on the training set after convergence. The neuronal circuits preserved in spite of Dropout during training, however, are sufficiently general to produce accurate results on the testing data [4]. In spite of the lower absolute and acceptable accuracy on the training set, that the D>1 measure is lower on the training set would tend to suggest that the network is still fit better—speaking broadly—on the training data.

A second curious result relates to the maximal topologies for each model: namely, they're quite different. The maximal topology for the Categorical Prediction approach included 128-node hidden layers immediately after both the prompt embedding input nodes and the essay embedding nodes, which both fed into a 256-node hidden layer node, which fed into the output layer. The maximal topology for the Absolute Prediction approach, however, required only a single 128-node hidden layer, fed by both embeddings, which fed into the single-node output layer—additional complexity in the network in the form of additional hidden layers actually *hurt* performance. This could perhaps be explained by the greater discrimination afforded by the absolute prediction scheme relative to the comparatively rigid categorical softmax output layer used in the Categorical Prediction approach.

VI. Conclusion

Given the extent to which topological variations in the hidden layers of the network between the input and output layers can positively or negatively impact model performance—sometimes, with little apparent rhyme, reason, or explanatory principle—we seriously doubt that we discovered the maximally performant topology for either model over the course of our investigations. Presuming that there is room for improvement in our models, there exists the distinct possibility that an embedding-based approach could surpass 71.9% absolute accuracy; recalling our discussion of the corpus, this would mean that such a machine grader awarded scores *more* consistently than a second human grader. That said, in our most-performant model, the consistency in scores awarded by our machine grader is only about 6% lower than in scores awarded by the second human grader—whether that 6% figure or something of a similar magnitude is acceptable relative to resources saved by using a machine instead is beyond the scope of this investigation. At any rate, the substantial gains in scoring accuracy—just north of 20% in the highest performing model—over the baseline with even a straightforward embedding-based approach relative to the disappointingly poor performance enabled by complex feature-rich approaches merits further exploration.

References

[1] Isaac Persing and Vincent Ng. *Modeling Argument Strength in Student Essays*. 2015.

[2] Nathan Ong, Diane Litman, and Alexandra Brusilovsky. *Ontology-based Argument Mining and Automatic Essay Scoring*. 2014.

[3] Quoc Le and Tomas Mikolov. *Distributed Representations of Sentences and Documents*. 2014.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, et al. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. 2014.

Appendix A

The following is the maximal discovered network topology for the Categorical Prediction method.

```
prompt_embedding = Sequential()
prompt_embedding.add(Embedding(config.prompt_count+1, config.prompt_embedding_size,
    input_length = 1))
prompt_embedding.add(Flatten())
prompt_embedding.add(Dense(config.prompt_hidden_layer_size))
prompt_embedding.add(Activation('sigmoid'))
prompt_embedding.add(Dropout(0.5))

d2v = Sequential()
d2v.add(Reshape((config.essay_embedding_size,),
    input_shape=(config.essay_embedding_size,)))
d2v.add(Dense(config.essay_hidden_layer_size))
d2v.add(Activation('sigmoid'))
d2v.add(Dropout(0.5))

out_score = Sequential()
out_score.add(Merge([prompt_embedding, d2v], mode='concat'))
out_score.add(Dense(config.hidden_layer_size))
out_score.add(Activation('sigmoid'))
out_score.add(Dropout(0.5))

out_score.add(Dense(config.hidden_layer_size))
out_score.add(Activation('sigmoid'))
out_score.add(Dropout(0.5))

out_score.add(Dense((config.score_range[1] - config.score_range[0])+1))
out_score.add(Activation('softmax'))
out_score.compile(loss='categorical_crossentropy', optimizer=config.optimizer)
```

with the configuration

```
essay_embedding_size = 150
prompt_embedding_size = 600
essay_hidden_layer_size = 128
prompt_hidden_layer_size = 128
hidden_layer_size = 256
targeted_prompts = [1, 2]
targeted_field = "rater_1_domain_1"
validation_fields = ["rater_1_domain_1", "rater_2_domain_1"]
score_range = (1, 6)
prompt_count = 8
p2v_training_iterations = 200
testing_slice_size = 0.2
batch_size = 8
optimizer = 'adam'
```

Appendix B

The following is the maximal discovered network topology for the Absolute Prediction method

```
prompt_embedding = Sequential()
prompt_embedding.add(Embedding(config.prompt_count+1, config.prompt_embedding_size,
    input_length = 1))
prompt_embedding.add(Flatten())

d2v = Sequential()
d2v.add(Reshape((config.essay_embedding_size,),
    input_shape=(config.essay_embedding_size,)))

out_score = Sequential()
out_score.add(Merge([prompt_embedding, d2v], mode='concat'))
out_score.add(Dense(config.hidden_layer_size))
out_score.add(Activation('sigmoid'))
out_score.add(Dropout(0.5))
out_score.add(Dense(1))
out_score.compile(loss='mse', optimizer=config.optimizer)
```

with the configuration

```
essay_embedding_size = 150
prompt_embedding_size = 600
hidden_layer_size = 128
targeted_prompts = [1, 2]
targeted_field = "rater_1_domain_1"
validation_fields = ["rater_1_domain_1", "rater_2_domain_1"]
score_range = (1, 6)
prompt_count = 8
p2v_training_iterations = 200
testing_slice_size = 0.2
batch_size = 8
optimizer = 'adam'
```