A Large Scale Quantitative Exploration of Modeling Strategies for Content Scoring

Nitin Madnani **Anastassia Loukina** **Aoife Cahill**

Educational Testing Service
Princeton, NJ, 08541 USA
{nmadnani,aloukina,acahill}@ets.org

Abstract

We explore various supervised learning strategies for automated scoring of content knowledge for a large corpus of 130 different content-based questions spanning four subject areas (Science, Math, English Language Arts, and Social Studies) and containing over 230,000 responses scored by human raters. Based on our analyses, we provide specific recommendations for content scoring. These are based on patterns observed across multiple questions and assessments and are, therefore, likely to generalize to other scenarios and prove useful to the community as automated content scoring becomes more popular in schools and classrooms.

1 Introduction

Automatic scoring of free-text content-based questions is a challenging task. Although it may appear similar to the task of automatically scoring student responses for writing quality (Page, 1966; Landauer et al., 2003; Attali and Burstein, 2006), it has important differences. Scoring for content deals with responses to open-ended questions designed to test primarily what the student knows, has learned, or can do in a *specific* subject area such as Computer Science, Math, Biology, or Music with fluency being secondary. It is not important if the student makes some spelling mistakes or grammatical errors as long as the desired specific information (e.g., scientific principles, trends in a graph, or details from a reading passage) is included in the response.

Assessing the content of student responses requires a different set of features that pay attention to whether students are using the right concepts, the right relationships between those concepts, and whether they are providing the right amount of detail. In addition, scoring for content generally requires building separate scoring models for each question since each question usually focuses on a specific set of concepts within a specific subject area. However, automated scoring for writing quality can utilize "generic" scoring models that can be used to assess student responses independent of the question to which they were written since the aspects of writing being measured are topic-independent (Attali and Burstein, 2006).

In this paper, we focus on a content scoring approach that learns a scoring model by extracting a large number of sparse, binary features from human-scored responses to a given question. The model can then predict scores for previously unseen responses to the question. There are many decisions that one needs to make for such an approach: what machine learning algorithm is likely to give the best performance? Is it better to use regression or classification? Is it worth allocating additional data and time for tuning the hyper-parameters of the learning algorithm? For many practical applications, the amount of human-scored data available may not even be sufficient for model training and evaluation let alone for these types of meta-analyses.

We conduct analyses on a large corpus of *real* student responses to identify patterns that are consistent across multiple questions and contexts and are, therefore, likely to generalize to other scenarios. Our corpus contains 130 different questions spanning four different subject areas and more than 230,000 human-scored responses. To our knowledge, this is the largest collection of responses ever used in a study on automated content scoring. The large number of questions allows us to test many of our hypotheses in a rigorous manner and convert the answers into specific recommendations for the community that we hope will be useful in guiding further development of supervised content scoring models.

2 Related Work

Content scoring is sometimes also referred to in the literature as "short-answer" scoring. Although it is true that many content-based questions tend to be very specific and elicit responses that are relatively short, this is not always the case. Previously published studies have considered responses that span a range of lengths — from a few words (Basu et al., 2013) to a few dozen words (Madnani et al., 2013; Horbach et al., 2013) to a few hundred words (Madnani et al., 2016). Given that the primary facet of interest is the content of the response and not its length, we refer to the task as "content scoring" in the rest of the paper.

Content scoring approaches fall into two general categories: (a) **reference-based** where responses are scored on the basis of their similarity to reference answers provided by the authors of the question or selected from existing high-scoring responses (Alfonseca and Pérez, 2004; Nielsen et al., 2008; Meurers et al., 2011; Sukkarieh et al., 2011; Horbach et al., 2013; Pado and Kiefer, 2015). These studies generally use a small number of continuous-valued features, often with a single model trained for multiple questions. (b) **response-based** which use a large number of detailed features extracted from the student responses themselves (e.g., word n-grams, etc.) and human scores assigned to the responses to learn a supervised machine-learning model (Mohler et al., 2011; Dzikovska et al., 2013; Ramachandran et al., 2015; Zesch et al., 2015; Zhu et al., 2016). Response-based approaches generally require training a separate model for each question.

The choice of whether the reference-based approach is better than the response-based approach depends on the open-ended nature of the question and whether there is a sufficient number of human-scored responses available. Sakaguchi et al. (2015) — who explored the combination of the two approaches — observed that if sufficient human-scored data is available, response-based approaches often work better than reference-based approaches. Since several of the questions in our dataset are relatively open-ended and we have sufficient scored data available for all of them, we focus on the response-based approach in this paper.

Our study is different from the work we have discussed so far in that its goal is not simply to obtain the best performance for a given question or a set of questions. Instead, we focus on meta-analyses of scoring performance as a function of modeling strategies and data set characteristics. Some previous studies have considered the choice of learner in automated scoring for writing quality. Chen and He (2013) compared support vector classification, regression, and ranking for automatically scoring writing quality using a single dataset. Chen et al. (2016) reported that using support vector regression with a radial kernel produced better performance than a simple linear model. In addition, several studies (Feng et al., 2003; Haberman and Sinharay, 2010; Santos et al., 2012) have consistently reported that use of probabilistic classifiers such as cumulative logistic regression might be more appropriate for the task of automated scoring than linear regression since such models incorporate the assumption that the score is categorical in nature. All of these studies used a small number of continuous-valued features.

More generally in the machine learning literature, papers have analyzed and compared the performance of different learning algorithms on standard machine learning datasets from the UCI repository and/or synthetic datasets (Caruana and Niculescu-Mizil, 2006; Matykiewicz and Pestian, 2012; Doan and Kalita, 2015). These studies reported substantial variability in learner performance across problems which suggests that the learners that performed best for other applications may not necessarily do so for our task.

The work that might be closest to ours is that of Heilman and Madnani (2015) who explored the impact of the amount of training data available on content scoring performance across a range of questions. However, they used a much smaller set of 44 questions and did not investigate any questions about specific modeling strategies such as the choice of learning algorithm or the impact of hyper-parameter tuning.

2.1 Research Questions

We aim to answer the following specific questions about supervised learning specifically in the context of automated scoring of content:

1. What type of learner has the best performance for response-based automated content

Subject	**N**	**Number of Responses**	**Score Range**	**Grade Level**	**Task Type**	**Response Lengths**
Science	89	454–5824	0–4, 0–6, 1–5	6–10	Explanations and arguments embedded in inquiry science units that call for students to use evidence to link ideas.	47–320 chars
English Language Arts	23	737–2685	0–2, 0–3, 0–4	7–10	Summarization, argument development, and analysis of English reading passages.	105–506 chars
Math	15	669–3265	0–2, 0–3, 0–4	7–9	Explanation of how mathematical principles apply to given situations involving linear equations.	40–150 chars
Social Studies	3	3000–3100	0–3	9–12	Summarization of stories and passages focused on social issues.	150–180 chars

Table 1: A detailed breakdown of our corpus by subject. **N**: number of questions; **Number of Responses**: minimum and maximum number of human-scored responses available for questions on this subject; **Score Range**: ranges of possible scores that can be awarded to responses for questions on this subject according to the human-authored scoring rubrics; **Grade Level**: the grades of students whose responses were used for analysis; **Response Lengths**: minimum and maximum number of characters in responses to questions on a given subject.

scoring? Do non-linear learners offer a substantial advantage over linear models? Do margin-based methods such as support vector machines perform better than bagging ensembles like random forests?

2. How do probabilistic classifiers perform compared to regressors when predicting real-valued scores?

3. Do hyper-parameters matter? Is it worth spending extra time and effort to tune the hyper-parameters of any given learner over simply using the default values provided by the implementation being used?

3 Methodology

Before we provide more details of our corpus and the specific learning strategies, we would like to describe two factors that will be shared by all strategies in order to perform a controlled comparison: (a) the features and (b) the evaluation metric.

Features. Since the goal of this paper is to compare modeling strategies, we need to use the same fixed set of features for all strategies in order to obtain a meaningful comparison. We use a set of features that have been employed in many previously published response-based approaches to building content scoring models (Heilman and Madnani, 2015; Zesch et al., 2015; Sakaguchi et al., 2015; Madnani et al., 2016). We extract the following features for all of the responses in our corpus:

(a) character n-grams including whitespace and punctuation (n=2–5)

(b) word n-grams (n=1,2)

(c) triples extracted over dependency parses obtained from ZPar (Zhang and Clark, 2011), and

(d) length bins (specifically, whether the log of 1 plus the number of characters in the response, rounded down to the nearest integer, equals x, for all possible x from the training set). For example, consider a question for which the responses in the training data are between 50 and 200 characters long. For this question, we will have 3 length bins numbered from 5 ($\lfloor \log_2 51 \rfloor$) to 7 ($\lfloor \log_2 201 \rfloor$). For a new response of length 150 characters, length bin 7 ($\lfloor \log_2 151 \rfloor$) would be the binary feature that gets a value of 1 with the other two bins getting the value of 0.

All of the features are binary (indicating presence or absence) and can be thought to indirectly approximate the requirements of content scoring we described earlier: good responses generally contain (a) the right concepts (approximately captured by words and bigrams), (b) the right syntactic relationships between those concepts (approximately captured by dependency triples), and (c) the right amount of detail (coarsely captured by length bins).

The character n-grams serve to capture spelling and morphological variations such that responses are not excessively penalized for misspellings or for using the incorrect morphological variants. For example, if the correct response to a question must contain the phrase ”temperature increased”, a candidate response containing the phrase ”temprature increase” (with a misspelling and an incorrect verb form) can still get credit for that concept.

Metric. Human scores generally tend to be integers, while automated scores can be either integers or real values on a continuous scale. One advantage of the real-valued scores is that they allow for more fine-grained distinction than a small set of integers. In this paper, we predict real-valued scores on a continuous scale and evaluate the accuracy of the predicted scores by using mean squared error (MSE) as our default metric. Although some previous studies have used quadratically-weighted kappa (QWK) as another possible metric for evaluating content-scoring models, more recent work has shown that QWK may possess properties that render it less than suitable for automated scoring evaluation (Yannakoudakis and Cummins, 2015).

3.1 Data

Our corpus contains over 230,000 human-scored responses that were collected in response to 130 different questions. The questions spanned 4 subject areas: Science, English Language Arts, Math, and Social Studies and are administered as part of several different assessments. The 130 questions include the 10 content-based questions from the Automated Student Assessment Prize competition organized by the Hewlett Foundation that are publicly available.[1] The remaining questions are in active use in various classroom settings and are not publicly available. Table 1 shows a detailed breakdown of the corpus.

3.2 Learners

Table 2 summarizes the learners considered in this study. We choose learners that (a) have either been shown to perform well with feature sets comparable to ours in previously published work — Mohler et al. (2011), Sakaguchi et al. (2015), and Zesch et al. (2015) all used support vector machines; Ramachandran et al. (2015) use a random forest regressor — or (b) are generally known to perform well with a large number of sparse features (Hastie et al., 2001; Fan et al., 2008; Chang and Lin, 2011). We use the scikit-learn (Pedregosa et al., 2011) implementations for all learners.

All the implementations incorporate some means of reducing learner variance either by design — random forests average over a large number of decision trees trained using bootstrapped samples — or by explicitly incorporating some form of regularization, e.g., an ℓ_2-penalty over the feature weights for logistic regression and a misclassification error penalty for SVMs.

The first four learners are classifiers. Since we are interested in predicting continuous values evaluated by using mean squared error, we would expect that classifiers that simply produce the most likely (integer) score would generally do worse than regressors, which produce continuous values. Therefore, for all four classifiers, we use the `predict_proba()` method of their scikit-learn implementations to obtain probability distributions[2] over the possible score points and then compute the expected value using the distribution as the final classifier prediction.[3] For the “Rescaled Support Vector Regressor”, we simply rescale the predictions obtained from the regressor using the mean and standard deviation of the human scores for the training data. This form of post-processing has been shown to be particularly effective as evidenced by its use in many of the top submissions to the Kaggle automated scoring competitions.

3.3 Experiments

Although we have a fairly large number of responses for each question, we choose to use cross-validation instead of a single train-test split in or-

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

[2] scikit-learn has two implementations available for a support vector classifier with a linear kernel: one using LibSVM (Chang and Lin, 2011) and another using LibLinear (Fan et al., 2008). We use the former since the latter doesn't support probabilistic classification.

[3] The probabilities can vary in reliability depending on the calibration algorithm chosen to convert the predictions of the classifier into posterior probabilities (Platt, 1999; Zadrozny and Elkan, 2002). For this paper, we assume that scikit-learn provides reasonably well-calibrated implementations.

Learner	Type	Linear	Grid
Random Forest Classifier	Classifier	No	max_depth: $[1, 5, 10, \texttt{None}]$
Logistic Regression	Classifier	Yes	C: $[10^{-3}, 10^{-2}, \ldots, 10^{3}, 10^{4}]$
SVC (linear)	Classifier	Yes	C: $[10^{-3}, 10^{-2}, \ldots, 10^{3}, 10^{4}]$
SVC (RBF)	Classifier	No	C: $[10^{-3}, 10^{-2}, \ldots, 10^{3}, 10^{4}]$, γ: $[\frac{1}{\|F\|}, 10^{-7}, 10^{6}, \ldots, 10^{-3}]$
Random Forest Regressor	Regressor	No	max_depth: $[1, 5, 10, \texttt{None}]$
SVR (linear)	Regressor	Yes	C: $[10^{-3}, 10^{-2}, \ldots, 10^{3}, 10^{4}]$
SVR (RBF)	Regressor	No	C: $[10^{-3}, 10^{-2}, \ldots, 10^{3}, 10^{4}]$, γ: $[\frac{1}{\|F\|}, 10^{-7}, 10^{6}, \ldots, 10^{-3}]$
Rescaled SVR (RBF)	Regressor	No	C: $[10^{-3}, 10^{-2}, \ldots, 10^{3}, 10^{4}]$, γ: $[\frac{1}{\|F\|}, 10^{-7}, 10^{6}, \ldots, 10^{-3}]$

Table 2: Learners chosen for this study and their characteristics. "linear" and "RBF" refer to linear and radial basis function kernels. The **Grid** column shows the grid of possible values searched when tuning the hyper-parameters for each learner. C denotes the complexity parameter that controls the amount of regularization for the learner. γ denotes the kernel coefficient for the RBF kernel ($\frac{1}{\|F\|}$ — where $\|F\|$ refers to the number of features — is a commonly used value for γ and we include it in the grid). The max_depth parameter for random forests controls the maximum depth of the tree. Setting it to `None` tells scikit-learn to automatically compute that number based on internal calculations.

der to average over any possible biases that a single split might yield. We perform two sets of 5-fold cross-validation experiments for each of the eight learners. For both sets of experiments, the folds are stratified by the human scores, e.g., all train/test splits have similar distributions of human scores.

In the first set of cross-validation experiments, we train each learner using default hyper-parameters for each of the five folds. We compute the MSE for each of the five folds by comparing to the human scores and then use their average as the final MSE for the learner. For the second set of experiments, instead of using the default hyper-parameter values, we run scikit-learn's `GridSearchCV` over the (four-fold) "training" set *inside* each of the five top-level cross-validation runs in order to search a pre-specified parameter grid for values that yield the lowest estimated MSE. The learner with these tuned hyper-parameters is then used to make predictions on the fifth held-out fold as per usual. As before, we use the average MSE value across the five folds as the final MSE value for the learner.

Table 2 shows the hyper-parameter grids that we search for each of the eight learners in the second set of experiments. Although there are several hyper-parameters that can be tuned for each learner, we focus on parameters that are more likely to have a significant impact on performance, e.g., those that control regularization and, hence, over-fitting. Any parameters not included in the grids are assigned default values by scikit-learn.

Both sets of cross-validation experiments (with and without hyper-parameter tuning) were conducted using the SKLL toolkit which makes it easy to run scikit-learn experiments with multiple learners in batch mode.[4]

4 An Aside: Feature Characteristics

Before we delve into the analyses of modeling strategies, it might be instructive to explore some characteristics of our feature set. On average, we extract a total of ~46,000 binary features for each question. As expected, the largest set of features are the character n-grams (about 3x as large as the word n-grams and the dependency triples). The length bin features constitute the smallest set with at most a dozen or so features.

We also wanted to explore how the number of features extracted for a given question varies by (a) the average length of a response for that question and by (b) the number of responses available for that question. Figure 1 shows correlations between the average number of features and the average response length and number of responses for

[4]`http://github.com/EducationalTestingService/skll`

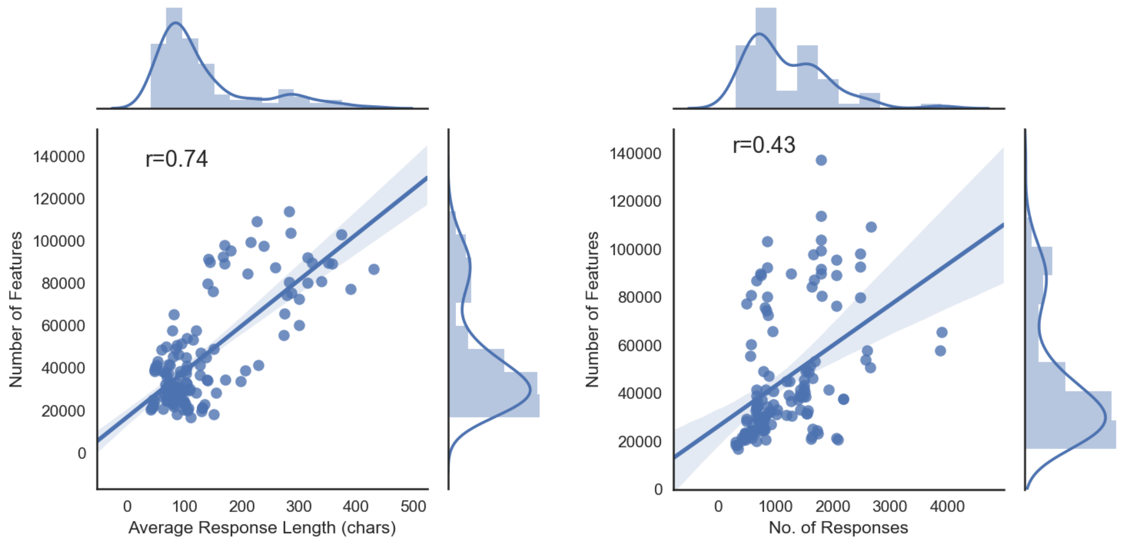

Figure 1: Joint distributions plots showing the impact of response length (in characters; left) and the number of available responses (right) on the number of features extracted for the questions in our corpus. Each plot is composed of a scatterplot with points denoting the 130 questions as well as histogram and density plots for each of the two variables being correlated. r = Pearson's correlation coefficient.

each question. As one might expect, the number of extracted features is larger for longer responses and for more numerous responses. We also observe that the average length of the response has a substantially larger impact on the number of extracted features than the number of responses available.[5] This intuitively makes sense since among the answers to the same question, there is a higher likelihood of newer words (and character sequences and dependency relations, etc.) being encountered *within* the same response as it grows longer, than across different responses.

5 Results

5.1 Effects of Modeling Strategies

Table 3 shows the mean squared error — averaged across all 130 questions in our corpus — for all eight learners both with and without tuned hyper-parameters. We observe that the best performance is achieved by probabilistic support vector classifiers with linear and RBF kernels.

5.1.1 General Learner Properties

We first explored whether general properties of the learners such as being linear vs. non-linear or being a probabilistic classifier vs. a regressor had a statistically significant impact on model performance across different learners and questions.

[5]We removed an extreme outlier from the left plot to minimize its impact on the correlation.

Learner	**Average MSE**	
	not tuned	**tuned**
Logistic Regression	.401	.391
Random Forest Classifier	.385	.368
Random Forest Regressor	.356	.356
SVC (linear)	.336	.326
SVR (linear)	.514	.388
SVC (RBF)	.434	.321
SVR (RBF)	.723	.342
Rescaled SVR	.546	.343

Table 3: Average MSE across 130 questions for each of the eight learners using hyper-parameter values that are either **not tuned** (default) values or **tuned** via grid search. Lower values are better.

We used a hierarchical mixed-effects linear model (Snijders and Bosker, 2011) implemented in R via the `lmerTest` package (Bates et al., 2015; Kuznetsova et al., 2016) to determine which modeling decisions have a statistically significant effect on model MSE over multiple questions. As our dependent variable, we used the standardized MSE value for each question and each learner. We also included the identity of each question and learner as a random factor to account for any random effects that might stem from characteristics unique to any particular question or learner. We used the following learner properties as independent variables, specified as discrete factors with a fixed set of possible values (shown in quotes):

- **Type**. Is the learner a (probabilistic) "classifier" or a "regressor"?
- **Linearity**. Is the learner "linear" or "non-linear"?
- **Family**. Is the learner logistic regression ("LR"), a random forest ("RF"), or a support vector machine ("SVM")?
- **Tuning**. Are the learner hyper-parameters "tuned" or "not tuned"?

We generally focused on the main effects of each factor but also included interactions between the **Tuning** factor and the other factors.[6] The equation describing the mixed-effects model is shown below:

$$\begin{aligned}\texttt{MSE} \sim\ & \texttt{Type} * \texttt{Tuning} \\ & + \texttt{Family} * \texttt{Tuning} \\ & + \texttt{Linearity} * \texttt{Tuning} \\ & + (1|\texttt{question}) + (1|\texttt{learner}) \quad (1)\end{aligned}$$

where MSE denotes the dependent variable; Type, Family, Linearity, and Tuning denote the learner factors we defined earlier, used here as fixed effects; the $*$ operator indicates an interaction between two factors; question and learner denote the question and learner identity respectively; and $(1|X)$ denotes the addition of a random intercept to the model for X. The reference values for the learner factors are: "classifier" (for Type), "SVM" (for Family), "non-linear" (for Linearity) and "tuned" (for Tuning).

The standardized coefficients for the model with all four fixed effects and the interactions are shown in Table 4. The coefficient in each row corresponds to the estimated difference in MSE (in number of standard deviations) relative to a learner with the chosen reference values for the learner factors. Note that since lower values are better for MSE, positive coefficients actually indicate worse performance and vice versa.

	Factor	**Coef.**	p-value
1	Intercept	$-.322$	.02
2	"linear"	.094	.47
3	"regressor"	.087	.43
4	"LR"	.205	.27
5	"RF"	.113	.38
6	"not tuned"	.596	$<.00001$
7	"LR": "not tuned"	$-.090$	.15
8	"RF": "not tuned"	$-.799$	$<.00001$
9	"linear": "not tuned"	$-.542$	$<.00001$
10	"regressor": "not tuned"	.395	$<.00001$

Table 4: Standardized coefficients and p-values for fixed factors and interactions included in the mixed-effects model (Equation 1). N=130 questions $\times$ 8 learners $\times$ 2 tuning conditions=2,080.

These results clearly show that not tuning the hyper-parameters leads to significantly worse performance (higher MSE) irrespective of learner (row 6). They also show that tuning interacts significantly with learner family, with linearity, and with learner type. For example, row 8 shows that the difference between "tuned" and " not tuned" versions of "RF" is significantly lower than the corresponding difference for the reference learner family ("SVM"). Therefore, we can infer that not tuning has a significantly larger detrimental effect on SVMs than on random forests. Similarly, we can infer that not tuning is significantly worse for the (reference) non-linear learners than for linear ones (row 9), and for regressors than for the (reference) classifiers (row 10).

The results also show that overall there are no significant differences in performance due to linearity (row 2), learner type (row 3), and learner family (rows 4 and 5). We want to be particularly clear about the interpretation of these rows. For example, row 2 states that just because you pick a non-linear model does not *automatically* mean that you will obtain better performance than a linear model, and so on. The specifics matter. That is, these results do *not* say anything about the differences between the specific learner instantiations used in our study.

[6]Any other higher-level interactions cannot be consistently evaluated using our set of learners. For example, to consider the interaction between learner Type and Linearity, we would want several learners representing each of the four possible combinations of linear and non-linear regressors and classifiers. In our chosen set of learners, linear regressors are represented *only* by SVR (linear) which makes it impossible to tell whether any patterns observed for linear regressors are actually meaningful or just quirks of SVR (linear).

5.1.2 Comparing Specific Learners

In the previous section, we considered whether there are any general learner properties that would lead to consistently significant differences in performance. However, one may also reasonably ask which of the particular learners in our study turned out to be most accurate. To answer this question, we fit another mixed-effects model, which also used MSE as a dependent variable and question as a random factor, but uses the learner as a fixed factor instead of a random factor. Unlike the model described by Equation 1 — where we treated the chosen learners as a sample from the population of all possible learners — in this model we are specifically interested in the differences *within* this set of learners. This model is described by the equation:

$$\texttt{MSE} \sim \texttt{Learner} * \texttt{Tuning} + (1|\texttt{question}) \quad (2)$$

We set the reference learner to be the best performing one from Table 3 – the SVC with RBF kernel and tuned hyper-parameters. The model coefficients (not shown here due to lack of space) confirmed the consistently useful effect of tuning we observed in the previous section as well as the different effect sizes of tuning for different learners. They also showed that, among the tuned learners, the reference learner significantly outperformed all other learners except for SVC (linear) with tuned hyper-parameters. We discuss the implication of these results in §6 and conclude with practical recommendations in §7.

6 Discussion

In this paper, we considered the effect that different modeling strategies have on the accuracy of automated content scoring. We analyzed the performance of 8 different learners on a very large corpus of real student responses to evaluate both the impact of general learner characteristics as well as differences between specific learner instantiations.

We found that no individual learner characteristic had a consistent effect on model performance across all learners and questions. For example, SVM-based probabilistic classifiers with tuned hyper-parameters outperformed all other learners, but this was no longer the case for SVM-based regressors or even SVM-based learners with default hyper-parameters. Similarly, (linear) logistic regression performed worse than random forests, but an SVC with a linear kernel performed comparably to SVC with an RBF kernel. In other words, our results indicate that no general family of learners is likely to be the most appropriate for this task out of the box: when choosing a learner, one should take into account *all* the factors considered in this study.

At the same time, we found that tuning hyper-parameters significantly improves model performance for all learners even when a moderate number of responses is available to train the model for each question (the models in our study were trained on an average on 1,400 responses). Finally, we identified probabilistic support vector classifiers with linear or RBF kernel and tuned hyper-parameters as the best performing learners across multiple questions in our corpus.

The sample of learners used in our paper is not exhaustive by any means. We focused on learners that have generally been shown to work well with similar features. There are other learners that we did not include in our study such as deep neural networks, nearest neighbor regressors etc. We leave the analysis of additional learner types for future work.

In addition to modeling choices, content scoring performance is also affected by data-related factors. In our analyses, we accounted for potential variation in baseline performance for each question by adding the question as a random factor to both mixed-effects models (Equations 1 and 2). We also conducted an additional exploratory study to evaluate whether the variation in performance of SVC (RBF), our best performing model, could be explained by specific question characteristics: the number of responses available, the number of different score levels, and the subject area. We found that the number of score levels had a moderate effect with model performance being higher for questions with a greater number of score levels, but there was no further effect of the number of responses available or of the subject area.

Of course, this does not mean that these parameters are not important for model performance. In fact, Heilman and Madnani (2015) reported a strong effect of the training size on the model performance for content scoring. A more likely explanation is that our data did not contain sufficient contrasts to establish such an effect: their study systematically considered training sets with N between 100 and 1,600, while in our study the N varied between 454 and 5,824 with the median at 1,350. It is also possible that the effects of the training size were confounded by other data-related factors. Furthermore, model performance may also be affected by the reliability of human scores, the context in which the assessment was delivered, and many other factors. We leave a more detailed exploration of such factors to future work.

7 Recommendations

Based on the observed results for general learner properties (§5.1.1) and for the specific learners (§5.1.2), we can provide the following recommendations to practitioners building automated content scoring models specifically when using features similar to ours:

1. It is generally beneficial to tune the hyper-parameters for every learner, if sufficient data and resources are available. [7]

2. If the goal is to pick a single learner that performs well for any question, the probabilistic SVC with an RBF kernel and with the C and γ parameters properly tuned via grid-search is likely to be a very good choice.

3. Not tuning the hyper-parameters has a substantial detrimental effect for non-linear SVMs, especially the regressors. If tuning is not possible due to lack of data or another reason, consider using random forests or probabilistic SVC with a linear kernel. If using scikit-learn/SKLL directly, these can be used "out of the box". However, if using another library, manually set the hyper-parameter values to be the same as the scikit-learn/SKLL defaults.

Acknowledgments

We would like to thank Beata Beigman Klebanov, Paul Deane, Guangming Ling, Brian Riordan, and the anonymous BEA reviewers for their useful comments and suggestions.

References

Enrique Alfonseca and Diana Pérez. 2004. Automatic Assessment of Open Ended Questions with a BLEU-inspired Algorithm and Shallow NLP. In *Advances in Natural Language Processing*, Springer, pages 25–35.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater V. 2. *The Journal of Technology, Learning and Assessment* 4(3):1–30.

Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. 2013. Powergrading: a Clustering Aapproach to Amplify Human Effort for Short Answer Grading. *Transactions of the Association for Computational Linguistics* 1:391–402.

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2015. Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software* 67(1):1–48.

Rich Caruana and Alexandru Niculescu-Mizil. 2006. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the International Conference on Machine Learning*. pages 161–168.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology* 2:1–27.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA, pages 1741–1752.

Jing Chen, James H. Fife, Isaac I. Bejar, and Andr A. Rupp. 2016. Building e-rater Scoring Models Using Machine Learning Methods. *ETS Research Report Series* 2016(1):1–12.

Tri Doan and Jugal Kalita. 2015. Selecting Machine Learning Algorithms Using Regression Models. In *Proceedings of the IEEE International Conference on Data Mining Workshops*. pages 1498–1505.

Myroslava Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. 2013. Task 7: The Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge. In *Proceedings of SemEval*. pages 263–274.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9:1871–1874.

Xin Feng, Neil Dorans, Liane N Patsula, and Bruce Kaplan. 2003. Improving the Statistical Aspects of E-rater ® : Exploring Alternative Feature Reduction and Combination Rules. *ETS Research Report Series* (June).

Shelby J. Haberman and Sandip Sinharay. 2010. The Application of the Cumulative Logistic Regression Model to Automated Essay Scoring. *Journal of Educational and Behavioral Statistics* 35(5):586–602.

[7] We would go so far as to recommend hyper-parameter tuning irrespective of task and the set of features used.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer New York Inc.

Michael Heilman and Nitin Madnani. 2015. The Impact of Training Data on Automated Short Answer Scoring Performance. In *Proceedings of the Workshop on Innovative Use of NLP for Building Educational Applications*. pages 81–85.

Andrea Horbach, Alexis Palmer, and Manfred Pinkal. 2013. Using the Text to Evaluate Short Answers for Reading Comprehension Exercises. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (* SEM)*. pages 286–295.

Alexandra Kuznetsova, Per Bruun Brockhoff, and Rune Haubo Bojesen Christensen. 2016. *lmerTest: Tests in Linear Mixed Effects Models*. R package version 2.0-33. https://CRAN.R-project.org/package=lmerTest.

Thomas K Landauer, Darrell Laham, and Peter W Foltz. 2003. Automated scoring and annotation of essays with the Intelligent Essay Assessor. *Automated Essay Scoring: A Cross-disciplinary Perspective* pages 87–112.

Nitin Madnani, Jill Burstein, John Sabatini, and Tenaha O'Reilly. 2013. Automated Scoring of a Summary-Writing Task Designed to Measure Reading Comprehension. In *Proceedings of the Workshop on Innovative Use of NLP for Building Educational Applications*. pages 163–168.

Nitin Madnani, Aoife Cahill, and Brian Riordan. 2016. Automatically scoring tests of proficiency in music instruction. In *Proceedings of the Workshop on Innovative Use of NLP for Building Educational Applications*. pages 217–222.

Pawel Matykiewicz and John Pestian. 2012. Effect of Small Sample Size on Text Categorization with Support Vector Machines. In *Proceedings of the Workshop on Biomedical Natural Language Processing*. pages 193–201.

Detmar Meurers, Ramon Ziai, Niels Ott, and Janina Kopp. 2011. 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*. pages 1–9.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments. In *Proceedings of ACL: HLT*. pages 752–762.

Rodney D Nielsen, Wayne H Ward, and James H Martin. 2008. Learning to Assess Low-Level Conceptual Understanding. In *Proceedings of the FLAIRS conference*. pages 427–432.

Ulrike Pado and Cornelia Kiefer. 2015. Short Answer Grading: When Sorting Helps and When It Doesnt. In *Proceedings of the Workshop on NLP for Computer Assisted Language Learning at NODALIDA*. 114, pages 42–50.

Ellis B Page. 1966. The imminence of... grading essays by computer. *The Phi Delta Kappan* 47(5):238–243.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

John Platt. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers* 10(3):61–74.

Lakshmi Ramachandran, Jian Cheng, and Peter Foltz. 2015. Identifying Patterns For Short Answer Scoring Using Graph-based Lexico-Semantic Text Matching. In *Proceedings of the Workshop on Innovative Use of NLP for Building Educational Applications*. Denver, Colorado, pages 97–106.

Keisuke Sakaguchi, Michael Heilman, and Nitin Madnani. 2015. Effective Feature Integration for Automated Short Answer Scoring. In *Proceedings of NAACL: HLT*. pages 1049–1054.

Vdo Santos, M Verspoor, and J Nerbonne. 2012. Identifying important factors in essay grading using machine learning. In *Language Testing and Evaluation Series (International Experiences in Language Testing and Assessment)*, volume 28, pages 295–309.

T.A.B. Snijders and R.J. Bosker. 2011. *Multilevel Analysis: An Introduction to Basic and Advanced Multilevel Modeling*. SAGE Publications.

Jana Z Sukkarieh, Ali Mohammad-Djafari, Jean-Francois Bercher, and Pierre Bessiére. 2011. Using a MaxEnt Classifier for the Automatic Content Scoring of Free-text Responses. In *Proceedings of the Conference on American Institute of Physics*. volume 1305, page 41.

Helen Yannakoudakis and Ronan Cummins. 2015. Evaluating the Performance of Automated Text Scoring systems. In *Proceedings of the Workshop on Innovative Use of NLP for Building Educational Applications*. pages 213–223.

Bianca Zadrozny and Charles Elkan. 2002. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 694–699.

Torsten Zesch, Michael Heilman, and Aoife Cahill. 2015. Reducing Annotation Efforts in Supervised

Short Answer Scoring. In *Proceedings of the Workshop on Innovative Use of NLP for Building Educational Applications*. pages 124–132.

Yue Zhang and Stephen Clark. 2011. Syntactic Processing Using the Generalized Perceptron and Beam Search. *Computational Linguistics* 37(1):105–151.

Mengxiao Zhu, Ou Lydia Liu, Liyang Mao, and Amy Pallant. 2016. Use of Automated Scoring and Feedback in Online Interactive Earth Science Tasks. In *Proceedings of the 2016 IEEE Integrated STEM Education Conference*.