Grading Essays with Supervised Learning

Bob Effinger
bdeffinger@wisc.edu

David Merrell
dmerrell@cs.wisc.edu

Jack Ordman
jordman@wisc.edu

Jack Truskowski
jtruskowski@wisc.edu

Samuel Vinitsky
vinitskys@cs.wisc.edu

Abstract

In this paper, we apply various supervised learning techniques to the domain of automated essay grading. This task was originally examined in a 2012 Kaggle competition sponsored by The Hewlett Foundation (Hamner 2012). Our goal was to match or improve upon the results of the models from the competition, which we did: our best model would have ranked 32nd of the 154 competitors.

Introduction

For our project, we explored the task of automatically grading essays using supervised learning. The Hewlett Foundation identifies exam grading as a primary expense in course administration, as essays are particularly labor-intensive to grade. In 2012, Hewlett sponsored a Kaggle competition to address this issue (Hamner 2012). The competition called for machine learning models that could emulate human graders.

Our project revisits the Kaggle competition with the goal of improving upon the results of past competitors. We explored several different machine learning approaches to this task such as simple regressors, ensemble methods, and probabilistic graphical models.

The Learning Task

As described above, our goal is to emulate human grading of essays. In this section, we describe the data available to us and formulate the problem as a machine learning task.

The Dataset We used the dataset provided in the Hewlett Foundation's Kaggle competition: a corpus of essays written by public school students enrolled in grades 7–10. The dataset is composed of responses to eight distinct essay prompts, with between 1,000 and 3,000 essays for each prompt (for a total of around 12,000 essays). Most essays are between 100 and 500 words in length.

Each essay in the corpus was assigned a score by human graders. However, the essays for different essay prompts were scored on different scales (e.g., 0–60 for one prompt or 1–4 for another prompt). We address this issue by transforming all scores to the interval [0,1]. Furthermore, each essay set was scored according to different criteria. None of the methods in this report explicitly account for the heterogeneous grading criteria, as we hypothesize that the general attributes of good writing are universal.

The original dataset was not suitable for our purposes, as it contained misspelled words and inconsistent capitalization. We preprocessed the corpus using Python's Natural Language Toolkit (NLTK), which allowed us to tokenize, correct spelling errors, and make all words lowercase. This served a double purpose, as we used some of the results of this cleaning as features (such as number of misspellings).

Regression Task Since our goal was to assign numeric scores, it seemed natural to approach this as a regression task. As described above, we trained our models to predict a score in the range $[0, 1]$. We then rescaled the predictions to the correct range of scores, as determined by the essay prompt. This rescaling is necessary for evaluating our models in a manner consistent with the original Kaggle competition, which employs a custom performance metric (described in the "Empirical Evaluation" section below).

Our Approach

We took several different approaches to this task including simple regressors, ensemble methods, and probabilistic graphical models. For each model, we used the same set of features, which we chose by doing a lesion study (described in "Results").

Feature Selection

The first set of features we tried were related to the structure of the essay itself: average word length, average number of words per sentence, and total number of words. The idea behind this is that the structure of the essay may be just as important as the content, and that writers of similar skill tend to use similar structure.

The second set of features we introduced were related to the quality of the writing: proportion of words that were misspelled and Flesch–Kincaid readability score. The idea here is that good writers tend to have fewer misspellings and better overall readability. Unfortunately, we were unable to successfully implement the readability score.

The last set of features we introduced were related to the grammar used in the essay: for each part of speech X, we

made a feature that was "frequency of words that are X". The idea here is that similar writers will tend to use similar distributions of parts of speech (a good writer may use a lot of adverbs and very few conjunctions, for example). The part of speech frequencies were calculated by parsing each sentence using NLTK's built-in parser, with a tag-set of 12 parts of speech (Bird, Klein, and Loper 2014). We chose this smaller tag-set to ensure each tag had enough representation.

Since we chose a wide range of feature types, they had widely different ranges. For example, the part of speech frequencies were in the range $[0, 1]$, but essay length was in the range $[0, 550]$. In light of this, we normalized each feature as discussed in class.

Regression using Vanilla Models

We tested several different simple regressors on this feature set, including decision trees, k-nearest neighbors, support vector machines, and neural nets with one hidden layer. These models performed well, as described in the "Results" section below.

Regression using Ensemble Methods

Since our vanilla models performed well, we explored the effect of boosting on their performance. Specifically, we used gradient boosting and ADABOOST on all of our vanilla models. The intuition behind using ADABOOST as one of our ensembles is that it is well suited for improving the accuracy of a learner that performs only slightly better than random guessing (Freund and Schapire 1996). Our preliminary results from testing suggested that our models only moderately reduce error compared to random guessing, thus an ADABOOST regressor was a good fit for our domain. We also used a gradient boosting regressor. Similarly to ADABOOST, gradient boosting attempts to overcome the limitations of a weak learner by sequentially adding new models to the ensemble, with a new base learner trained every iteration using the "error-so-far" (Natekin and Knoll 2013). This provides dimensionality-reduction and reduces over-fitting. While gradient boosting often uses decision trees as the base model, we used all of the models in order to compare how each vanilla learning model responds to gradient boosting.

Classification using Graphical Models

Another approach used simple graphical models to classify the essays, with classes corresponding to discretized scores.

A simple hypothesis is that the "goodness" of an essay is reflected in the sequences of words that appear in it. In other words, some sequences of words are more probable than others, given the essay's score. This hypothesis can be represented by a directed graphical model, as illustrated in Figure 1. Note the similarity of this hypothesis to n-gram document models. It only differs by additional dependencies on the score variable y.

Mathematically, this hypothesis assumes the following distribution over words w_i that appear in a document, and the document's score y:

$$p(w_1, w_2, \dots w_L, y) = p(y) \prod_i p(x_i | y, x_{i-1}, \dots x_{i-n+1})$$

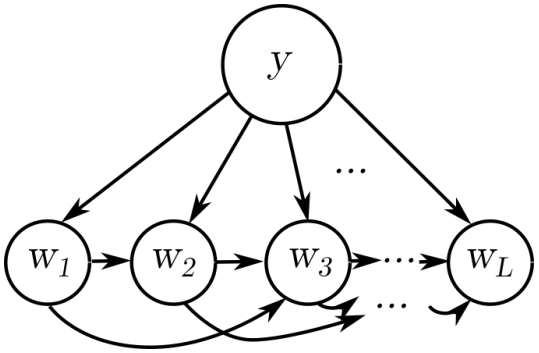

Figure 1: An n-gram document classifier model. Note that when $n = 1$, this is equivalent to Naïve Bayes over the document's tokens.

The corresponding rule for classification is

$$\begin{aligned}\hat{y} &= \arg\max_{y \in \mathcal{Y}} p(y | w_1, w_2, \dots, w_L) \\ &= \arg\max_{y \in \mathcal{Y}} p(w_1, w_2, \dots, w_L | y) p(y) \\ &= \arg\max_{y \in \mathcal{Y}} p(y) \prod_i p(w_i | w_{i-1}, w_{i-2}, \dots, w_{i-n+1}, y)\end{aligned}$$

Algorithmically, this amounts to training a distinct n-gram model for each $y \in \mathcal{Y}$; i.e., each possible value of the score variable. Documents are classified by computing the probability of the document with each of the $|\mathcal{Y}|$ trained n-grams. The score y with greatest n-gram probability is then assigned to the document.

The simple n-gram model can be augmented to include additional features in a Naïve Bayes fashion. This augmented hypothesis posits that additional features x_j depend directly on the document's score—see Figure 2 for illustration. This enables the model to benefit from feature engineering.

With this augmentation, the classification rule becomes

$$\begin{aligned}\hat{y} = \arg\max_{y \in \mathcal{Y}} p(y) \prod_i (w_i | w_{i-1}, w_{i-2}, \dots, w_{i-n+1}, y) \\ \cdot \prod_j p(x_j | y)\end{aligned}$$

For convenience, the conditional probability distributions $p(x_j|y)$ are discretized into conditional probability *tables* by some binning rule. The statistical soundness of discretization is questionable, but we find that extra features do improve the classifier's performance, despite using naïvely discretized CPT's.

Our augmented n-gram hypothesis family has parameters that need to be tuned. These include:

- n—the length of the n-grams.
- $\mathcal{Y}$—the discretization of the score variable. Currently, we restrict ourselves to uniform discretizations and only adjust the *number* of discrete values.
- The binning rule for conditional probability tables $p(x_j|y)$. Choices include a simple $\sqrt{N}$ rule, or Sturges' $\log(N) + 1$ rule (Sturges 1926).

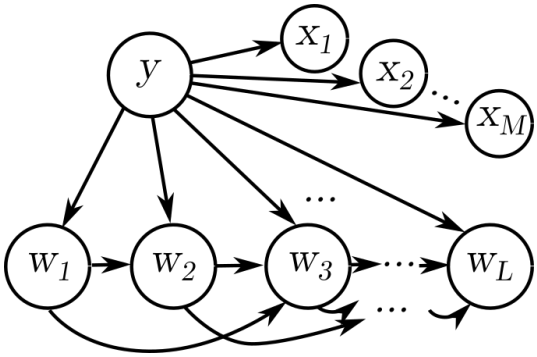

Figure 2: An n-gram document classifier model, augmented to account for additional document features.

The results of tuning and evaluation are detailed later in this report.

Empirical Evaluation

In this section, we describe our methods for evaluating our models, and present our results.

Testing Methodology

While we originally intended to use the training and testing sets Kaggle provided to the competition participants (Hamner 2012), the testing set from the competition was not publicly available. Instead, we built our own test set by separating out 10% of the training data. This produced an adequately-sized training set of around 11,000 essays and a testing set of around 1,200 essays. The split was stratified on score and essay set.

Evaluation Metrics

We used two different evaluation metrics: root-mean-square error (for internal evaluation) and the quadratic weighted kappa metric (for comparison with models from the competition).

Root-Mean Squared Error As discussed in class, root-mean-square error is a metric that measures a model's average performance on a regression task. The root-mean-square error is computed as

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}},$$

where $\hat{y}_i$ denotes our predicted score for the i'th essay, and y_i denotes the actual score.

We chose to use RMSE error as the metric for internal evaluation, since it is more intuitive than the Hewlett Foundation's metric.

Quadratic-Weighted Kappa Metric The Hewlett Foundation used a *quadratic weighted kappa metric* to evaluate the performance of models. Roughly speaking, their metric describes the level of agreement between two raters, A and B. In our setting, one rater is a human and the other is a learned model.

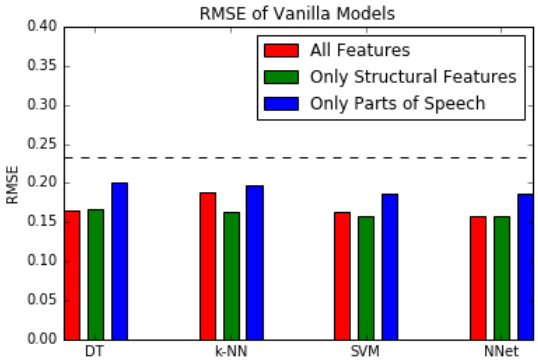

Figure 3: Comparative performance of vanilla models using root mean squared error. The dotted line represents the RMSE for guessing the mean of the test set.

Model	Parameter	Values
Decision tree	max depth	$[1, 100]$
k-NN	k	$[1, 100]$
	weighting	{uniform, distance}
SVM	kernel	{RBF, sigmoid, poly of deg. 1,2,3}
Neural Networks	num hidden units	$[1, 100]$
	activation function	{identity, RELU, sigmoid, $\tanh$}

Figure 4: Parameters we tested for each vanilla model.

The quantity κ is computed as follows:

$$\kappa = 1 - \frac{\sum_{i,j}\mathbb{P}[A = i, B = j](i-j)^2}{\sum_{i,j}\mathbb{P}[A = i]\cdot\mathbb{P}[B = j](i-j)^2}$$

Notice that $\kappa = 1$ when the two raters always assign the same score. On the other hand, $\kappa = 0$ when the raters' scores are stochastically independent of each other. For more details see the Kaggle site (Hamner 2012).

We used the κ metric to compare our own models with those submitted to the Kaggle competition.

Results

In this section, we present the results from the testing as described above. We used RMSE for internal evaluation, and κ for comparing our performance with that of the Kaggle competition. Recall that we want a low RMSE and a high κ.

Vanilla Models Each of the models we tested has several tunable parameters. We chose to vary a select few parameters that seemed the most important based on our discussion of these models in class and our preliminary testing. See Figure 4 for a complete list of these parameters.

All of the vanilla models performed better than guessing the mean of the test set (see Figure 3). Across all models, the inclusion of part of speech frequencies as features hurt performance. However, even the parts of speech features on their own were able to outperform mean guessing.

The different models all seemed to perform relatively similarly under their best parameters. Varying the feature vectors had a much stronger effect on RMSE than varying the model. Interestingly enough, most of the models had RMSE

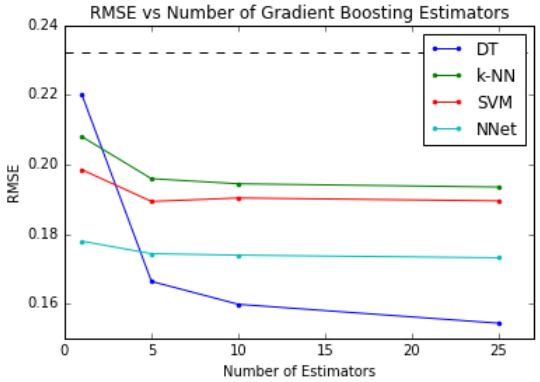

Figure 5: Relationship between the number of ensemble estimators used in boosting and RMSE. The dotted line represents the RMSE for guessing the mean of the test set.

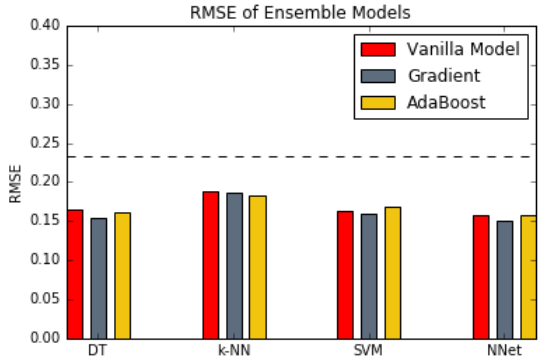

Figure 6: Comparative performance of ensemble models using root mean squared error. The dotted line represents the RMSE for guessing the mean of the test set.

less than 0.23 (the RMSE of mean guessing) under almost all parameters. The only exception to this was support vector machines, which did horribly for certain kernels (the sigmoid kernel had a RMSE of 53).

The parameters that produced the best results for each model are as follows:

- **Decision tree:** max depth $= 5$
- **k-nearest neighbors:** $k = 20$, weighting $=$ distance
- **Support vector machine:** kernel $=$ RBF
- **Neural network:** hidden units $= 100$, activation $=$ RELU

Ensemble Methods Overall, both gradient boosting and ADABOOST marginally reduced mean-squared error across all essay sets and learning models with a few exceptions.

For decision trees, k-nearest neighbors, and neural networks, both gradient boosting and ADABOOST outperformed the best vanilla model. With the exception of k-nearest neighbors, gradient boosting produced the greatest reduction in RMSE. However, gradient boosting provided only a marginal improvement — with the greatest reduction of 0.011 and an average reduction of 0.0054 compared to the best vanilla model. ADABOOST had an even smaller improvement, averaging 0.00396 and being out-performed by both ADABOOST and the best vanilla model with an SVM base estimator.

Varying the number of estimators seemed to effect the accuracy of ensemble models. As expected, using a single estimator did not improve the accuracy over the best vanilla model. However, increasing the number of estimators reduced the RMSE for all base models. In the case of k-nearest neighbors, support vector machines, and neural networks, increasing the number of estimators beyond five did not result in a meaningful reduction in error (See Figure 5). While decision trees had the worst RMSE with a single estimator, it was the only model that saw consistent improvement with increasing estimators (up to 25).

Graphical Models We tuned the parameters of the augmented n-gram model using 5-fold cross-validation on the

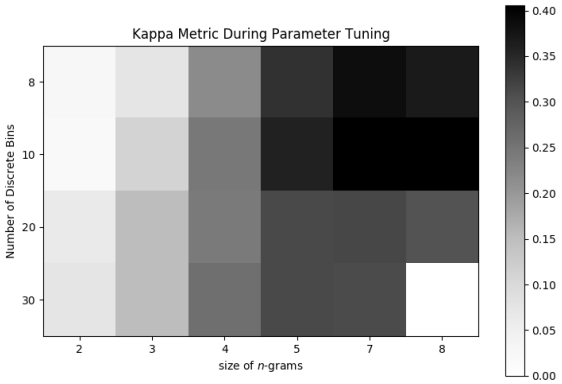

Figure 7: Parameter tuning for the n-gram model. This figure shows the κ metric, averaged over folds, for different parameter settings. As suggested in the figure, setting $n = 8$ and $|\mathcal{Y}| = 10$ resulted in the best validation performance. Note that the binning rule is set to Sturges in this cross-section.

training set. When subsampling for cross-validation, we stratified over the essay set and (normalized) essay scores. Our search explored a rectangular region of parameter space:

- Choosing $n \in [1, 10]$
- Choosing $|\mathcal{Y}| \in \{8, 10, 12, 20, 30\}$
- Using bins generated by either Sturges' binning rule or $\sqrt{N}$ binning.

During validation and testing, we always augmented the n-gram model with 5 additional features: *(i)* fraction of misspelled words; *(ii)* average word-length; *(iii)* essay length; *(iv)* the occurrences of numbers in an essay; *(v)* the occurrences of capitalized words in an essay.

A cross-section of the parameter search is illustrated in Figure 7.

We found that setting $n = 8$, $|\mathcal{Y}| = 10$, and using Sturges' binning rule for feature discretization yielded the highest kappa metric, averaged over folds. Broadly speaking, κ metric was most strongly determined by n—performance was

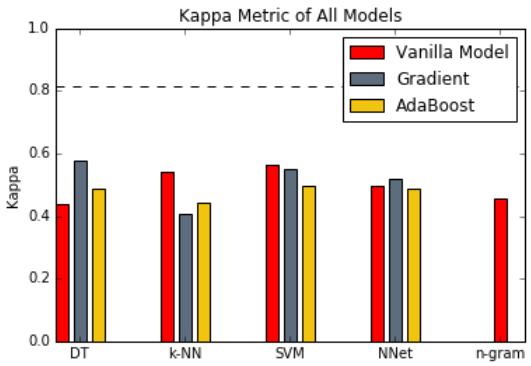

Figure 8: Comparative performance of ensemble models using the κ metric. The dotted line represents the κ metric for the best model from the Kaggle competition.

very poor for small n, tended to improve with increasing n and peaked at $n = 8$. The score discretization $|\mathcal{Y}|$ also had significant influence on performance. Interestingly, performance peaked consistently at $|\mathcal{Y}| = 10$, across values of n.

Using these parameters, the augmented n-gram model achieved a κ metric of 0.4577 on the held-out test set. This was surprising, given that the model's highest observed κ during validation was near 0.41 We discuss possible causes for this in the Discussion section of this document.

Comparison to Kaggle Competition

In order to compare our models with the best models from the Kaggle competition, we used the κ metric (described above), which was the metric used by the competition. See Figure 8 for our results. Interestingly, the more sophisticated n-gram model performed no better than the simpler models. In fact, the best model was a simple boosted decision tree. This seems to imply that the more sophisticated models were held back by sub-par features.

As stated above, the best performing model was decision trees with gradient boosting, which had a κ of 0.57876. If we had entered the Kaggle competition, we would have ranked 32^{nd} out of 154 participants. However, our best κ score was still much lower than the best performing models in the competition, which boasted a κ of 0.81407.

Discussion

Our results looked good overall, but closer analysis provides interesting observations.

Regression using Vanilla Models As seen in Figure 3, our performance was affected much more by our choice of features than by our choice of model. In particular, the addition of the part of speech frequency features did not give much improvement over structural features in any model (though they did well enough on their own). This implies that the part of speech features may have been "explained away" by the structural features.

The fact that our performance across all regression models was similar implies that our performance was constrained by the quality of available features, not the quality of the model. This suggests that better features may result in much better performance for these same models.

Ensemble Methods Unsurprisingly, decision trees responded the best to boosting. Boosting works by using information from multiple overlapping regions of the feature space (Gorman 2017), which is well-represented by a tree structure. Ensemble models proved to be very strong in conjunction with a decision tree base model. Using gradient boosting produced the highest κ score of all models (0.579, Figure 8).

While using boosting on the other models did marginally reduce the RMSE, the fact that they didn't respond as strongly as decision trees suggests that there may be better-suited ensembles for these regressors. Additionally, with the exception of decision trees and gradient boosting on neural nets, boosting performed worse than the vanilla models on the κ evaluation metric.

Overall, boosting is an excellent method of improving decision tree learners in this domain, but did not provide a significant improvement on any of the other vanilla models.

Graphical Models Evaluating the augmented n-gram model showed surprising results.

First, we observe that a fairly large value of $n = 8$ yielded the best performance in validation. It seems intuitive that smaller n would generalize better to unseen instances. Given our relatively small data set, it is hard to imagine many sequences of 8 words that appear frequently enough to inform an essay's score. It may be that long sequences of words *are* informative in this setting as a result of commonalities between the authors, who are all middle- or high-school students. They are likely to have similar mannerisms resulting from their demographic similarity.

Another surprise was that the augmented n-gram's test set κ metric was a fair amount higher than its average κ during cross-validation. Two factors may account for this:

1. The model's performance is data-constrained—the extra 20% of the training set available to the model during testing gave it greater predictive power.
2. Randomness in our train/test split. Since we made a 90-10 split, the test set is relatively small and could be subject to significant deviation. The correct response to this would be to cross-validate, rather than using a single 90-10 split as we have done.

It was also surprising that Sturges' binning rule yielded a significantly better performance than the $\sqrt{N}$ binning rule. During validation, Sturges' rule provided a κ metric as much as 80% greater than the $\sqrt{N}$ binning rule; a change from $\kappa = 0.24$ to $\kappa = 0.43$. Mathematically, Sturges' rule yields a much smaller number of bins—$\log(N) + 1$—than the $\sqrt{N}$ rule. Visually, it produces a much coarser discretization. One might expect the "nicer" looking discretization to yield better classification performance, but this is not the case. It is likely the case that coarsely discretized distributions generalize better to unseen instances, giving better predictive power.

Future Work

Although our models performed well, we still did not reach the level of success of the best models from the Kaggle competition. In this section, we describe future work that might improve the success of our models.

Feature Engineering For our vanilla regressors, we found that the choice of features was the biggest factor in success. As such, choosing better features would likely be the biggest factor in improving these models. One feature we were unable to successfully implement was the Flesch–Kincaid readability score. There are also many other features we could explore relating to grammar and sentence structure (such as the use of passive voice).

Additional Distance Metrics For k-nearest neighbors, there are several distance metrics that are built into scikit-learn. We tried to implement our own distance metric based on cosine similarity, but this proved to be much worse than the default metric. It would be interesting to explore the effect of distance metrics on the performance of k-nearest neighbors (both in this setting and in general).

Ensemble Methods Although our ensemble models performed very well, there is always room for improvement. While gradient boosting and ADABOOST are both theoretically well-suited to our domain and produced good results, it would be interesting to try other variants of these algorithms as well as other ensemble learners.

Graphical Models The augmented n-gram model did not compete well with the other models presented in this paper—not to mention the winners of the Kaggle competition. However, there are some interesting directions for developing this model further.

One possibility is to modify the model to perform regression, rather than classification. There are multiple ways to approach this. A naïve approach would use essentially the same model, but instead of returning the $\arg\max_{y \in \mathcal{Y}}$, would return the expected value of y. However, this lacks theoretical justification and we have no reason to think it would do better than the current model.

A more sophisticated approach would assume n-grams are distributed by a categorical random variable that depends on y. In particular, we might assume there is a one-dimensional curve parameterized by y on the probability simplex; i.e., each value of y is associated with a unique distribution over the n-grams. Learning this model would amount to fitting that curve to the frequencies observed in the data. There are still many details to work out here, and it is not clear whether such a model would work.

A more conservative improvement to the augmented n-gram model would use *continuous* conditional probability distributions for the augmenting variables, in place of the discrete CPTs described in this report. Generally speaking, discretizing continuous variables is a statistically unsound practice. This is evidenced by our observation that changing the feature binning rule produced a significant change in κ metric during parameter tuning.

LDA: Topics as Features For a set of documents with the same prompt, the semantic content of an essay may be correlated with the grade the essay received. By using latent Dirichlet allocation (LDA), we can infer the distribution of topics for each essay. We could then use this distribution of topics as a feature vector, and input it into any of our existing regression models. Although it may seem as though essays on a particular topic would all have similar topic distributions, it may be the case that the distribution of less frequent topics within a larger field still may be informative.

LDA: Topics as Grades Although usually used for topic modeling, LDA may work well for direct grade prediction as well. Instead of learning word distributions for each topic, we could learn word distributions for each *grade* – the idea here being that writers of similar levels use similar vocabulary. In order to classify new essays, we could just predict the "topic" as in regular LDA, but this "topic" would really correspond to a grade. This approach is an extension of the less sophisticated graphical models we used in this paper.

Deep Learning As discussed in class, LSTM's are well suited for natural language processing tasks. During our research, we did not have the computational resources necessary to implement an LSTM, but it would be very interesting to explore deep learning models for this domain.

Concluding Remarks

Our goal for this project was to create models for the task of essay grading. While our models did not perform as well as the winners of the Kaggle competition, we still did better than some of the entries, and much better than a baseline of random guessing. By analyzing the performance of these models we are able to discover what worked well for this regression task, and gain insight as to how to design better models in the future.

References

Bird, S.; Klein, E.; and Loper, E. 2014. Natural language processing with python. https://www.nltk.org/book/ch05.html.

Freund, Y., and Schapire, R. 1996. Experiments with a new boosting algorithm. https://cseweb.ucsd.edu/~yfreund/papers/boostingexperiments.pdf.

Gorman, B. 2017. A kaggle master explains gradient boosting. http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting.

Hamner, B. 2012. The hewlett foundation: Automated essay scoring. https://www.kaggle.com/c/asap-aes.

Natekin, A., and Knoll, A. 2013. Gradient boosting machines, a tutorial. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/.

Sturges, H. 1926. The choice of a class interval. *Journal of the American Statistical Association*.