Benefits of Modularity in an Automated Essay Scoring System

Jill BURSTEIN
ETS Technologies
Rosedale Road
Princeton, NJ 08541
jburstein@ets.org,
jburstein@etstechnologies.com

Daniel MARCU
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292-6601
marcu@isi.edu

Abstract

E-rater is an operational automated essay scoring application. The system combines several NLP tools that identify linguistic features in essays for the purpose of evaluating the quality of essay text. The application currently identifies a variety of syntactic, discourse, and topical analysis features. We have maintained two clear visions of *e-rater's* development. First, new linguistically-based features would be added to strengthen connections between human scoring guide criteria and *e-rater* scores. Secondly, *e-rater* would be adapted to automatically provide explanatory feedback about writing quality. This paper provides two examples of the flexibility of *e-rater's* modular architecture for continued application development toward these goals. Specifically, we discuss a) how additional features from rhetorical parse trees were integrated into *e-rater*, and b) how the salience of automatically generated discourse-based essay summaries was evaluated for use as instructional feedback through the re-use of *e-rater*'s topical analysis module.

1 Introduction

E-rater is an operational automated essay scoring system that was designed to score essays based on holistic scoring guide criteria (Burstein, et al 1998), specifically for the Graduate Management Admissions Test (GMAT). Holistic scoring guides instruct the human reader to assign an essay score based on the quality of writing characteristics in an essay. For instance, the reader is to assess the overall quality of the writer's use of *syntactic variety*, the *organization of ideas*, and appropriate *vocabulary use*. *E-rater* combines several NLP tools to identify syntactic, discourse, and vocabulary-based features.

In developing this automated essay scoring application, we have two primary goals. We are continually experimenting with *e-rater* to enrich its current feature sets to represent additional scoring guide criteria. Furthermore, we are adapting the system to provide test-takers with feedback about the quality of their writing, so that they may use it to improve their overall writing competency.

In light of the application development goals, this paper discusses the *e-rater* application components and the benefits of its modular design. Using specific studies to exemplify, the paper points out the importance of the application's modularity with regard to: a) experiments that evaluate the integration of new features, and b) the re-use of modules for evaluations that contribute to the adaption of the system toward the generation of feedback.

2 *E-rater* System Modules & Design

The *e-rater* application currently has five main independent modules. The application is designed to identify features in the text that reflect writing qualities specified in human reader scoring criteria. The system has three independent modules for identifying scoring guide relevant features from the following categories: syntax, discourse, and topic. Each of the feature recognition modules described below identifies features that correspond to scoring guide criteria features which can be correlated to essay score, namely, *syntactic variety*, *organization of ideas*, and *vocabulary usage*. *E-rater* uses a fourth independent model building module to select and weight predictive features for essay scoring. The model building module reconfigures the feature selections and associated regression weightings given a sample of human reader scored essays for a particular test question. A fifth module is used for final score assignment.

All modules are called from a main driver program. Each independent module can be run as a stand-alone program. There are interactions between the modules, and these are described throughout the paper.

The modules and their subcomponents are written in either Perl or C programming languages. The model building module is implemented in SAS, a statistical programming language. *E-rater* can be run on both Unix or PC platforms.

2.1 Syntactic Module

E-rater's syntactic analyzer (parser) works in the following way to identify syntactic features constructions in essay text. *E-rater* tags each word for part-of-speech (Brill, 1997), uses a syntactic "chunker" (Abney, 1996) to find phrases, and assembles the phrases into trees based on subcategorization information for verbs (Grishman, et al, 1994). The parser identifies various clauses, including infinitive, complement, and subordinate clauses. The ability to identify such clause types allows *e-rater* to capture *syntactic variety* in an essay.

2.2 Discourse Module

E-rater identifies discourse cue words, terms, and syntactic structures, and these are used to annotate each essay according to a discourse classification schema (Quirk, et al, 1985). The syntactic structures, such as complement clauses, are outputs from the syntactic module described earlier. Such syntactic structures are used to identify, for example, the beginning of a new argument based on their position within a sentence and within a paragraph.

Generally, *e-rater's* discourse annotations denote the beginnings of arguments (the main points of discussion), or argument development within a text, as well as the classification of discourse relations associated with the argument type (e.g., *parallel relation*). Discourse features based on the annotations have been shown to predict the holistic scores that human readers assign to essays, and can be associated with *organization of ideas* in an essay.

E-rater uses the discourse annotations to partition essays into separate arguments. These argument partitioned versions of essays are used by the topical analysis module to evaluate the content individual arguments (Burstein, et al, 1998; Burstein & Chodorow, 1999). *E-rater's* discourse analysis produces a flat, linear sequence of units. For instance, in the essay text *e-rater*'s discourse annotation indicates that a contrast relationship exists, based on discourse cue words, such as *however*. Discourse-based relationships across sentences in text are not defined by this module.

2.3 Topical Analysis Module

Vocabulary usage is another criterion listed in human reader scoring guides. To capture use of vocabulary, or identification of topic *e-rater* includes a topical analysis module. The procedures in this module are based on the vector-space model, commonly found in information retrieval applications (Salton, 1989). These analyses are done at the level of the essay (big bag of words) or the argument.

For both levels of analysis, training essays are converted into vectors of word frequencies, and the frequencies are then transformed into word weights. These weight vectors populate the training space. To score a test essay, it is converted into a weight vector, and a search is conducted to find the training vectors most similar to it, as measured by the cosine between the test and training vectors. The closest matches among the training set are used to assign a score to the test essay.

As already mentioned, *e-rater* uses two different forms of the general procedure sketched above. For looking at **topical analysis at the essay level**, each of the training essays (also used for training *e-rater*) is represented by a separate vector in the training space. The score assigned to the test essay is a weighted mean of the scores for the 6 training essays whose vectors are closest to the vector of the test essay.

In the method used to analyze **topical analysis at the argument level**, all of the training essays are combined for each score category to populate the training space with just 6 "supervectors", one each for scores 1-6. The argument partitioned version of the essays generated from the discourse module are used in the set of test essays. Each test essay is evaluated one argument at a time. Each argument is converted into a vector of word weights and compared to the 6 vectors in the training space. The closest vector is found and its score is assigned to the argument. This process continues until all the arguments have been assigned a score. The overall score for the test essay is an adjusted mean of the argument scores.

2.4 Model Building and Scoring

The syntactic, discourse, and topical analysis modules each yield numerical outputs that can be used for model building, and scoring. Specifically, counts of identified syntactic and discourse features are computed. The counts of features in each essay are stored in vectors for each essay (test candidate). Similarly, for each essay, the scores from the topical analysis by-essay, and topical analysis by-argument procedures are stored in vectors. The vectors generated from each module are stored in independent output files. The values in the vectors for each feature category are then used to build scoring models for each test question as described below.

To build models, a training set of human scored sample essays is collected that is representative of the range of scores in the scoring guide. For the type of essay generally scored by *e-rater*, the scoring guides typically have a 6-point scale, where a "6" indicates the score assigned to the most competent writer, and a score of "0" indicates the score assigned to the least competent writer. Optimal training set samples contain 265 essays that have been scored by two human readers. The data sample is distributed in the following way with respect to score points: 15 1's, and 50 in each of the score points 2 through 6.[1]

The model building module is a program that runs a forward-entry stepwise regression. Feature values stored in the syntactic, discourse, and topical analysis vector files are the input to the regression program. This regression program automatically selects the features which are predictive for a given set of training data (from one test question). The program outputs the predictive features and their associated regression weightings. This output composes the model that is then used for scoring.

In an independent scoring module, a linear equation is used to compute final essay score. To compute the final score for each essay, the sum of the product of each regression weighting and its associated feature integer is calculated.

2.4.1 *Advantages of Modularity for Model Building & Scoring*

In the model building program, one can choose to use all the features for a particular run, or some feature subset. This flexibility makes it relatively easy to introduce new sets of features into the model building procedure for research and development purposes. The model building module can be run independently. Therefore, once *e-rater* has generated feature vector files for training samples, the model building module can be revised accordingly, so that numerous runs can be performed on data sets, using various feature combinations for model building, without rerunning the entire application.[2]

Once new models have been built, they can be easily cross-validated on an independent data set. Specifically, once the feature vector information has been generated for the independent data set, it can be scored quickly using any model desired to test the performance of the model. For each new model, the vector information, (e.g., counts of syntactic clauses) is recombined in the linear equation using the model-specific predictive features and regression weightings. Therefore, given the same set of test data, performance may vary across models.

The design of an independent scoring module is also useful for tracking down changes in performance that occur when making revisions to the code. Code changes can have unexpected affects on feature assignment which can alter vector counts. If vector counts are affected for a feature used in the model, then this may affect the final essay score. Simple comparisons can be made between the scoring equation variables in a previous version of the code, and the revised version. Such comparisons are often useful to trouble-shoot the unanticipated affects of code changes on specific feature variables, and final scores.

3 Benefits of Modularity for Application Development

As discussed earlier, a goal in *e-rater* application development is to enhance the current feature set by adding new features that correspond to characteristics of writing defined in the scoring guide criteria. Currently, *e-rater* features represent these scoring guide criteria: *syntactic variety*, *organization of ideas*, and *vocabulary usage*. *E-rater* discourse features capture the criterion, *organization of ideas*, at a high level. However, the existing discourse features are linear, and do not express relationships across a text. Hierarchical discourse relations can be expressed with rhetorical structure theory (RST) features (Mann and Thompson, 1989).

In an experiment, we evaluated the potential use of RST features in *e-rater*. An existing rhetorical parser (Marcu, 1997) was used to generate parse trees for essay samples from 20 test questions to the GMAT. A program was written to identify the RST features in essays, compute counts of tokens, types and ratios of the features, and to store the three categories of feature counts in vectors for each essay. For the RST vector files, separate files were output for each type of feature count (tokens, types, and ratios). The model building program was modified to introduce the new RST variables. In this way, the RST feature variables could be evaluated either individually or in combination during model building -- as specified in the model building program.

E-rater had been run on these 20 essay samples previously, so all of the standard vector information that *e-rater* outputs already existed. The model building component in *e-rater* can easily be run independently once all vector information exists, so the process of building new models after RST feature variables had been integrated was quickly and easily done.

Accordingly, the evaluation of experimental models on independent test sets is also conveniently done with the *e-rater* scoring module. Specifically, the predictive features and their associated regression weightings from the new models that include RST features are introduced into the linear equation used in scoring.

So, in experimental runs (of which we do many!), only the additional pieces, in this case the rhetorical parser, and RST feature extraction program, were required for feature generation, and extraction, and creation of formatted vector files used as input to the model building and scoring programs. This particular experiment provided strong evidence that the RST features would serve to enhance the current application.

Running model building and scoring independently on an essay sample (training and cross-validation[3] sets) for a single prompt takes approximately 5 seconds. To build a model and score the same essay sample would take up to an hour. The independence of the model building and scoring programs allows unlimited flexibility for continued research and development of the application with regard to the addition of new features.

4 Re-Using *E-rater's* Topical Analysis Module

A strong motivation behind *e-rater* application development is to adapt the system so that it generates feedback along with an essay score. In a recent experiment, we re-used the *e-rater* topical analysis module, and the essay data to evaluate the saliency of text in automated essay summaries (Burstein and Marcu, 2000). The score from the topical analysis by-argument module is amongst *e-rater's* strongest predictors of essay score. That is, it is almost always selected in the model building process. Furthermore, by itself, the topical analysis by-argument score agrees with human reader scores approximately 85% of the time, on average.[4]

Within the context of adapting *e-rater* to generate feedback, we hypothesized that summaries could be used to determine the most important points of essays. We envisioned at least two possible uses of essay summaries. First, for any essay question, one can, for example, build individual summaries of all essays of score 6 (the most competent essay); use sentence-based similarity measures to determine the topics that occur frequently in these essays; and present these topics to a test-taker. Test-takers would then be able to assess what topics they might have included in order to be given a high score. Second, for any given essay, one can build a summary and present it to the test-taker in a format that makes explicit whether the main points in the summary cover the topics that are considered important for the test question. One way of doing this might be to present to test-takers, summaries of other essays that received a high score. Test-takers would be able to assess whether the rhetorical organization of their essays makes the important topics salient.

For the experiment, the training and cross-validation sets from the 20 GMAT essay samples were run through an existing discourse-based automatic text summarizer (Marcu, 1999). Summaries were generated at different compression rates: 20%, 40% and 60%. For each of the 20 samples, the topical analysis module was run on training and cross-validation sets. We evaluated the performance of the topical analysis by-argument score on all summaries.[5] The performance of the topical analysis by-argument measure was higher for 40% and 60% summaries than using the full text of essays. The re-use of this *e-rater* module for evaluating the saliency of essay summaries proved to be informative.

5 Discussion and Conclusions

In this paper, we have discussed the importance of modularity in an automated essay scoring system for research and development. Modularity, especially with regard to the model building and scoring functionality, is critical to application development. Unlike other NLP tools, such as part-of-speech taggers and syntactic parsers, for which there is a reasonably well-defined and standard feature set, the feature set that will become part of *e-rater* will be determined by continued experimentation. Though *e-rater* currently contains linguistic features that have been shown to be highly predictive of essay score, the interests and queries from the writing community require further experimentation with new features (such as RST features).

As was discussed in the paper, the new types of features that could become used in the system reflect qualities of writing that appear in scoring guide criteria. These criteria are "fuzzy" in some sense, in that they describe general qualities of writing (e.g., organization of ideas), but do not state specifically what form of linguistic feature will reflect a particular quality. Therefore, repeated experimentation with new features is critical in order to discover how to represent these criteria computationally.

From a purely linguistic perspective we must first ask: *What linguistic features map to the concept, <u>organization of ideas</u>, for instance?* But, in addition, from the computational linguistic view we must also ask: *What are the linguistic features that map to a scoring guide criteria that can be reliably captured by NLP-based tools?* To further develop *e-rater*, we must be able to handle both points-of-view; hence, a modular system is required in which we can easily test the use of new features (or, hypotheses about new features) toward further application development. The ability to easily modify *e-rater's* model building module, so that models can be easily reconfigured with new feature combinations allows us to conveniently evaluate the performance of new features. This is shown in the experiment in which RST features were introduced into *e-rater* models. This approach also allows us to quickly evaluate feature performance within the linear regression modeling technique. What we have also learned through our continued research is that alternative measures outside of the linear regression may also be useful to characterize the competency of an essay with regard to its rhetorical structure. Similar research is on-going that employs alternative methods of evaluating the relevance of essay vocabulary using measures independent of the regression. It is critical to have the ability to evaluate the reliability of different approaches for representing and evaluating features of writing as they relate to writing competency.

A second argument for the modularity of the system is to be able to re-use independent *e-rater* tools and data for related applications (e.g., automated scoring of short answers). Alternatively, in the summarization experiment, we were able to re-use the essay data for the purpose of generating summaries, and also to re-use the topical analysis tool to evaluate the performance of the tool on the summaries. Since the topical analysis component is an independent module, no modifications were required to run the experiment.

Acknowledgements

We thank Martin Chodorow and Dennis Quardt for helpful discussions and insights. We are also grateful to Magdalena Wolska for writing the programs that perform the model building and scoring functionality for our research.

References

Abney, S. (1996) Part-of-speech tagging and partial parsing. In Young, S. and Bloothooft, G. (eds), Corpus-based Methods in Language and Speech. Dordrecht: Kluwer, 118-136.

Brill, E. (to appear). <u>Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging</u>, Natural Language Processing Using Very Large Corpora. Dordrecht: Kluwer Academic Press.

Burstein, J. and D. Marcu (2000). Toward Using Text Summarization for Essay-Based Feedback. In the *Proceedings of TALN 2000, Swiss Federal Institute of Technology,* Lausanne, Switzerland, October, 2000.

Burstein, J., Kukich, K., Wolff, S., Lu, C., Chodorow, M., Braden-Harder, L., and Dee Harris, M. (1998). Automated Scoring Using A Hybrid Feature Identification Technique. In the *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, Montreal, Canada.

Grishman, R., Macleod, C., and Meyers, A. (1994). "COMLEX Syntax: Building a Computational Lexicon", Proceedings of Coling, Kyoto, Japan. (available for download at:http://cs.nyu.edu/cs/projects/proteus/comlex/)

Mann, W.C. and Thompson, S.A. (1988). Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text* 8(3), 243–281.

Marcu, D. (1999). Discourse trees are good indicators of importance of text. In I. Mani and M. Maybury eds., *Advances in Automatic Text Summarization*, pp. 123-136.The MIT Press.

Marcu D. (1997). The Rhetorical Parsing of Natural Language Texts. *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 96-103.

Quirk, R., Greenbaum, S., Leech, S., and Svartik, J. (1985). A Comprehensive Grammar of the English Language. Longman, New York.

Salton G. (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Publishing Co.

[1] Essays at score point 0 are not required as these tend to contain no text at all, or to be off-task in some way.
[2] In practice, we wrote a program that performs the functionality of the model building and scoring modules. It is in this program where code revision actually occurs, not in the application code.
[3] Cross-validation samples usually contain about 500 essays.
[4] Agreement statistics are for the 20 GMAT essay samples discussed. The agreement indicates that the human reader and topical analysis scores are within 1-point. This is a standard measure of agreement between 2 human readers. Additionally, two human readers agree within 1 point of each other approximately 92% of the time.

[5] The performance of the topical analysis by-argument scores is approximately 5% higher than the scores from the topical analysis by-essay procedure.