Benefits of Modularity in an Automated Essay Scoring System

Jill BURSTEIN
Educational Testing Service
Princeton, NJ
jburstein@ets.org

Daniel MARCU
ISI/University of Southern California
Marina Del Rey, CA
marcu@isi.edu

Abstract

E-rater is an operational automated essay scoring application that combines several NLP tools for the purpose of identifying linguistic features in essay responses to assess the quality of the 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 reuse of *e-rater*'s topical analysis module.

1 Introduction

The Electronic Essay Rater (*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, so that they incorporate 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 three main independent modules for the following feature identification: syntax, discourse, and topic. The application is designed to identify features in the text that can be linked to writing qualities defined in scoring criteria, used by human readers for manual essay scoring. Each of the 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 separate model building module to select and weight predictive features for essay scoring. The model building module reconfigures the feature selections and associated weightings for individual test questions. The scoring module is also an independent procedure. All modules are called from a main driver program. Each independent module can be run as a stand-alone program. 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 us to capture *syntactic variety* in an essay.

2.2 Discourse Module

E-rater identifies discourse cue words and terms, and uses them to annotate each essay according to a discourse classification schema (Quirk, et al, 1985). The annotation marks the beginnings of arguments (the main points of discussion) within a text, as well as the type of discourse relations associated with the argument type and its development (e.g., *contrast 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 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 similar final outputs that can be used for model building, and scoring. Specifically, counts of identified syntactic and discourse features are calculated. 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 the form of a vector. Vectors from each module are stored in independent output files.

To build models, a training set of human scored sample essays 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. Features from the syntactic, discourse, and topical analysis vector files are input to the regression program. This program automatically selects the features which are predictive for a given set of training data (from one test question). The module outputs the features and associated regression weightings. This output composes the model which is then used in scoring. In an independent scoring module, an equation is used to compute final essay score, whereby a sum is calculated using the product of each regression weighting and its associated feature integer.

2.4.1 *Advantages of Modularity for Model Building & Scoring*

In the **model building program**, the vector files to be input to the regression can be modified as needed to include only the feature vector files desired for a particular model building run. That is, one can choose to use all the feature vector files for a particular run, or some subset of feature vector files. 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 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] The same is true for cross-validation data, that is, if vectors are generated for an independent test set, then the scoring module can be re-run independently, to test new models on an independent data set.

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* (Marcu and Burstein, in preparation). 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.

E-rater had been run on these 20 essay samples previously, so all of the standard vector information that *e-rater* outputs already existed. For the RST vector files, separate files were output for each category of feature count (tokens, types, and ratios). In this way, the feature count categories could be evaluated individually or in combination in model building. Because the model building component *in e-rater* can be run independently once all vector information exists, the process of building and evaluating the models where RST features have been integrated is quickly and easily done. Accordingly, the evaluation of experimental models on independent test sets is also conveniently done with the *e-rater* scoring module. 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 identification, extraction, and generation of formatted vector files for 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. The score from the topical analysis by-argument module is one of *e-rater's* strongest predictor 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. It is to some extent the job of the computational linguistic researcher to determine not only what makes sense from a theoretical perspective, that is: *What linguistic features map to the concept, <u>organization of ideas</u>, for instance?* But, from a computer science point-of-view we must 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, hypothesis about new features) toward application development.

A second argument for the modularity of the system is to be able to re-use *e-rater* tools and data for related applications (e.g., automated scoring of short answers), or the example that we used in the paper, toward the adaption of *e-rater* as an application that provides feedback.. 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 usefulness of the data. Since the topical analysis component is an independent module, no modifications were required to run the experiment.

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). Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging, Natural Language Processing Using Very Large Corpora. Dordrecht: Kluwer Academic Press.

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. and Burstein, J. (in preparation). Using Rhetorical Structure Theory in Automated Essay Scoring to Increase Statistical & Theoretical Validity

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.