A Scalable Learning Analytics Platform for Automated Writing Feedback

Jacqueline Feild
McGraw-Hill Education
Boston, MA
jacqueline.feild
@mheducation.com

Nicholas Lewkow
McGraw-Hill Education
Boston, MA
nicholas.lewkow
@mheducation.com

Neil Zimmerman
McGraw-Hill Education
Boston, MA
neil.zimmerman
@mheducation.com

David Boulanger
Athabasca University
Edmonton, CA
david.boulanger
@dbu.onmicrosoft.com

Jeremie Seanosky
Athabasca University
Edmonton, CA
jeremie
@rsdv.ca

ABSTRACT

In this paper, we describe a scalable learning analytics platform which runs generalized analytics models on educational data in parallel. As a proof of concept, we use this platform as a base for an end-to-end automated writing feedback system. The system allows students to view feedback on their writing in near real-time, edit their writing based on the feedback provided, and observe the progression of their performance over time. Providing students with detailed feedback is an important part of improving writing skills and an essential component towards solving Bloom's "two sigma" problem in education.

We evaluate our feedback system in two ways. First, we evaluate the effectiveness of the feedback for students with an ongoing pilot study with eight hundred students who are using the learning analytics platform in a college English course. In addition, we process an existing set of graded student essays and analyze the performance feedback. Results show a correlation between feedback values and human graded scores.

Keywords

Analytic Tools for Learners; Automated Essay Feedback; Scalable Analytics; Performance Feedback; Natural Language Processing

1. INTRODUCTION

Performance feedback is essential for self-regulated learning, which is an attribute of highly effective learners [3, 18]. Bloom has shown that providing formative feedback to students increases performance, compared to only providing final feedback [1]. This allows students to develop and implement actionable strategies for improving performance as they progress. Formative feedback is even more effective if it can be given in near real-time [7, 13].

In this paper we describe a scalable platform for learning analytics called OpenACRE (Analytics Collaborative Research Environment) which is currently in development to be released as open source. OpenACRE allows for ingestion of heterogeneous educational data from multiple source systems, long-term storage of raw data, running arbitrary models on the raw data using a parallel analytics engine, and short-term storage of resulting analytics for use by students, teachers, and researchers. As a proof of concept, we implement an end-to-end writing feedback system utilizing OpenACRE. Writing feedback is especially hard to provide in real-time and at scale as it is computationally expensive, making it well suited for the capabilities provided by OpenACRE.

There are several other existing writing feedback systems which provide various feedback to students, for example Revision Assistant, WriteToLearn, and Writing Pal [17, 14, 12]. While these systems provide useful information, they are either commercial black boxes which do not allow for modification, or are intelligent tutoring systems which provide writing instruction through customized modules. OpenACRE stands apart by providing the ability to develop and deploy new analytical models at scale, making it useful for researchers to test new feedback algorithms, predictive models, or reporting dashboards on a large number of students.

To evaluate our proof of concept system in a classroom setting, an efficacy study is currently underway to investigate the usefulness of the feedback to improve student performance. The study consists of eight hundred college students who are learning English at VNR VJIET in India. Additionally, we evaluate the feedback from 13,000 existing student essays and compare it to the human graded scores.

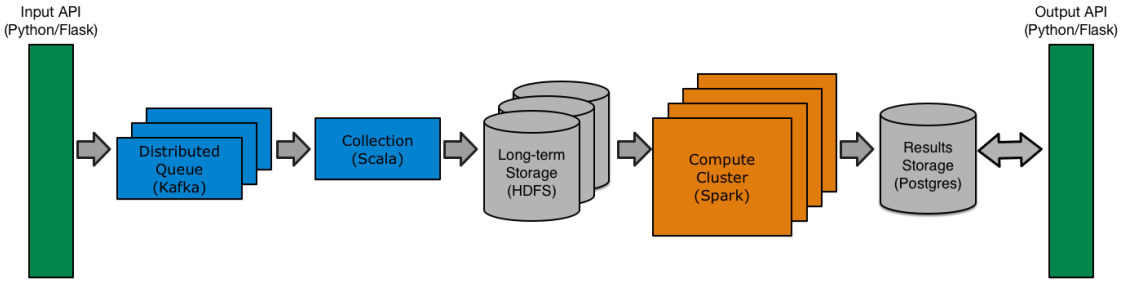

Figure 1: Architecture diagram of the learning analytics platform, corresponding to the middle box in Figure 2. Data is ingested by the input API and placed into a distributed queueing system which is implemented using Kafka. A collection service, implemented in Scala, pulls data from the queue and stores it in long-term storage, which is implemented using Hadoop Distributed File System (HDFS). The compute cluster runs models in parallel on the data in long-term storage and persists output views to the results store, implemented in PostgreSQL. Output views can then be accessed through the output API. Both the input and output APIs are RESTful and implemented in Python using Flask.

2. OPENACRE

The OpenACRE platform consists of an input and output API, long- and short-term databases, and a parallel computation cluster. A low-level diagram is shown in Figure 1. This platform is designed to handle the challenges of scalability, resiliency to data loss, and fault tolerance. Additionally, OpenACRE is built to be extensible for future models, without the need for drastic modification to the system as a whole. For example, models which perform machine learning algorithms, complex aggregations, and graph analysis could all be implemented to run on OpenACRE. These models could include traditional classroom statistics, score predictions, or personalized learning recommendations.

Learning event data is ingested into OpenACRE through the input API and persisted to the long-term data store. The input API for OpenACRE is implemented in a RESTful fashion using Python with the Flask package. RESTful APIs are used because they are stateless, easily extended for future functionality, and agnostic to programming language. The input API accepts event data from external sources and temporarily stores the events in a queueing system. We utilized open source Apache Kafka for our queueing system as it is distributed, durable, and supports APIs in several commonly used languages. Next, a collection service takes events from the queue and stores them in a long term data store. Here we use the open source Hadoop Distributed File System (HDFS) since it is distributed and fault tolerant. The collection service in OpenACRE is written in Scala, but any language supported by the Kafka and Hadoop APIs could also be used. The event data stored in HDFS is kept in its original "raw" form and is never altered. Storing unaltered event data allows for arbitrary computation and the implementation of future models without knowledge of those models beforehand.

Next, the computation engine runs analytical models by taking data from the long term store and performing transformations/aggregations to create new output views. These output views can be accessed by users through the output API. Open source Apache Spark was used for our computation engine as it allows for user-friendly parallel computation, horizontal scalability on commodity hardware, and contains a rich set of APIs ranging from simple map-reduce to machine learning algorithms. Additionally, Apache Spark currently implements APIs in Java, Python, and Scala.

Output views from a given model are written to the results store database which is implemented using PostgreSQL in OpenACRE. PostgreSQL was used as it is open source, has APIs in several languages, and provides a familiar SQL interface for queries. From the results store, output views are provided to external users through the output API. Similar to the input API, this is implemented as a RESTful API so it is stateless and can be easily accessed from the majority of modern languages. The output API can then accessed by other backend systems or user facing systems, such as dashboards. The combination of all the OpenACRE components listed above results in a learning analytics platform which can ingest arbitrary learning event data, apply parallel analytic models to the data, and provide the results of the analytics to external systems and dashboards in a generic fashion.

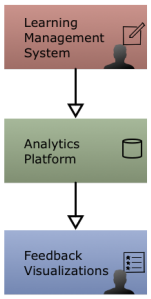

Figure 2: High-level diagram of the end-to-end writing feedback system. The learning management system and feedback visualizations are student-facing while the learning analytics platform stores writing data and computes feedback.

While any type of data format could be ingested into OpenACRE, we chose the standardized learning event format called Caliper, supported by IMS Global [4]. Caliper defines a set of standard learning events composed of actor-action-object triples. An example event is 'student-submits-quiz1'. While actor-action-object triples are also used in other standardized learning event formats like TinCan [16], Caliper has a significant benefit in that is uses JSON-LD, which is a schema-based JSON format. In addition to being schema-based, JSON-LD allows for easy mappings from JSON to domain-specific ontologies.

3. END-TO-END WRITING FEEDBACK SYSTEM

As a proof of concept, we built an end-to-end writing feedback system with OpenACRE at the core. Writing feedback is an excellent use case for OpenACRE as it is very computationally expensive, requiring approximately 12 seconds per essay for our feedback model. This large processing time results in almost 7 days of computation for a single assignment in a large MOOC of 50,000 students. Implementing our writing feedback model on OpenACRE allows that computation time to be cut to hours or minutes, depending on the size of the computation cluster.

The end-to-end proof of concept system includes the student facing system, which collects student writing data from their learning management system (LMS) and displays the automated feedback visualizations, and the backend system built on OpenACRE, which stores and analyzes the student data. Figure 2 shows a high-level view of this system, including both the student facing and backend systems.

The typical workflow for a student using this system includes:

1. Log in to writing course using an LMS
2. Start a writing assignment
3. Save the writing assignment
4. View visualizations of writing feedback
5. Edit writing assignment based on provided feedback
6. Save the writing assignment
7. Repeat steps 4-6 as needed
8. Submit assignment

This workflow provides feedback to students at regular intervals and gives students the opportunity to improve their writing before submitting their assignment. The ongoing pilot provides feedback in 24 hour increments due to cost constraints on the size of the computation cluster. Since the LMS which students are using is instrumented to directly collect writing data, there is no need to use an additional feedback system. This allows for an intuitive interaction between the student and their LMS, while collecting data for feedback at the same time. In our implementation, we utilized Moodle for our LMS as it is open source, familiar to both students and educators, and was easily instrumented to collect writing data as Caliper events and send those events to OpenACRE.

We designed a custom dashboard to display feedback visualizations to students and instructors. These include both a snapshot of overall feedback and the progression of feedback over time.

3.1 Feedback Competences

The feedback provided by our system is composed of seventeen writing competences which have been developed over the last several years [9]. These include traditional writing metrics such as spelling and grammatical accuracy as well as more advanced metrics that capture sentiment and writing flow. In the following sections, we describe several groups of writing metrics and define the competences we implement within them.

3.1.1 Traditional Metrics

Traditional writing metrics include competences that are often used by teachers to evaluate student writing. The competences implemented in our system from this category include vocabulary, spelling, grammatical accuracy, and lexical diversity. The vocabulary competence represents the amount of unique words in the student's text. As the student uses more unique words in their writing, the vocabulary competence increases. The spelling competence measures the percentage of incorrectly spelled words used. This competence increases as the percentage of misspelled words in the text decreases. Similar to spelling, the grammatical accuracy competence measures the percentage of grammatical errors in the text. This competence value increases as the percentage of grammatical errors decreases. Finally, the lexical diversity competence measures the percentage of unique words in the text. The value increases as students use more unique words relative to the size of the text.

3.1.2 Advanced Metrics

Advanced writing metrics highlight more subtle and complex characteristics of English writing. While not always explicitly listed in a writing rubric, these metrics are important for proficient English writing. The competences implemented in our system from this category include modifier complexity, noun phrase complexity, and tense agreement. The modifier complexity competence represents the amount of noun or verb modifiers which are used in the student's text. A high number of noun or verb modifiers indicates that the writing is more complex and expressive. The noun phrase complexity competence analyzes the number of noun phrases in the student's text. This metric attempts to measure the linguistic complexity for a piece of writing, as more noun phrases typically indicates richer sentences. Finally, the tense agreement competence measures the consistency of verb conjugations in the text. This competence value increases when verbs are conjugated consistently throughout a piece of writing.

3.1.3 Flow Metrics

Writing flow metrics measure how ideas are connected both within adjacent sentences and throughout entire pieces of text. The competences we implement in this category are

local cohesion, global cohesion, and connectivity. Local cohesion tracks the flow of ideas from sentence to sentence. Writing that contains adjacent sentences with similar nouns and verbs receives a higher local cohesion score. Similarly, global cohesion tracks the flow of ideas throughout an entire piece of writing, which is also measured by the similarity of nouns and verbs throughout the text. Connectivity measures the use of phrases that connect ideas to one another. Text with more coordinating conjunctions receives a higher connectivity score.

3.1.4 Descriptive Metrics

These writing metrics measure how descriptive a piece of writing is in several different ways. The competences we implement in this category are concreteness, imagery, familiarity, and conciseness. Concreteness measures the degree to which the text refers to tangible objects. Higher concreteness scores are obtained by using more words that refer to tangible objects. Imagery gives a measure of the amount of words within the text which evoke a mental image. Similarly, familiarity measures the amount of words in a text that are commonly used. The calculation of concreteness, imagery and familiarity are based on pre-defined scores in each category for commonly used words. These pre-defined scores were determined experimentally by asking human subjects to rate words in these three categories [5]. Conciseness measures the ratio of content words in the text. Writing that includes more nouns, verbs, adverbs and adjectives receives a higher conciseness score.

3.1.5 Sentiment Metrics

Sentiment metrics reflect the tone or feel of a piece of writing. These are computed using state-of-the-art techniques with the Stanford CoreNLP library [10, 15]. The required sentiment may vary based on the type of writing or subject matter. The competences we implement in this category include negative tone, neutral tone, and positive tone. The negative tone competence describes the degree to which the writing exhibits negative sentiment. Similarly, the neutral and positive tone competences describe the degree to which the writing exhibits neutral or positive sentiment. All three of these competences measure the amount of negative, neutral, or positive words in the writing.

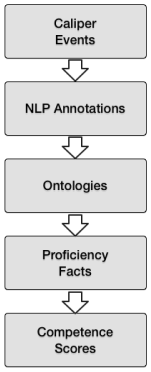

Figure 3: High-level diagram showing the flow of the openSCALE algorithm from caliper events to competence scores.

3.2 OpenSCALE

The analytics model implemented in this automatic writing feedback system is called OpenSCALE [2]. This model parses text with Stanford CoreNLP library [10], creates ontologies and facts from the annotated text, and aggregates the facts into competence scores for students.

A high-level view of the transformations which go from text to competence scores is described in Figure 3. First, the text is annotated using the Stanford CoreNLP library [10]. The annotations include tokenization of the text into words and sentences, part of speech tagging, syntactic parsing and sentiment analysis. These annotations are used to create an ontology of the relationships between words, sentences and paragraphs in the text, including both their structure and semantic meaning. For each piece of text, openSCALE creates one ontology using the open source Apache Jena library.

Next, each ontology is put through an inferencing layer, which looks for patterns in the ontology that show evidence of students having a particular skill/competence and creates proficiency facts. Each fact includes information about the degree of competence (weight) for a unique student-assignment attempt-time. Many facts are generated from a single ontology going through the inferencing layer. The inferencing layer in openSCALE is implemented using VIStology's BaseVISor framework [11]. BaseVISor works by passing a set of rules dictating how facts are generated for a given ontology. The ability for users to specify specific rules allows for great flexibility as different instructors could potentially dictate what is seen as evidence of different skills/competences. The current implementation of openSCALE uses a default set of rules which are used by BaseVISor.

Finally, the proficiency facts are aggregated to generate final scores for each competence. The main flow of the fact aggregations for student, competence, assignment attempt, and time is:

1. Sum the weights for all facts with the same student-competence-assignment attempt-time
2. For each fact F (student S - assignment attempt A - competence C - time T):
   (a) Find all facts at or before time T with the student S - competence C
   (b) Keep the facts of the newest attempt for each assignment
   (c) Sum the competence weights and update F

The final, aggregated facts are used to generate the competence progression view in the results store. The view displaying a snapshot of overall feedback is created by taking the latest aggregated facts for each competence.

4. PILOT RESEARCH STUDY

We are currently running a pilot research study to test the usefulness of the feedback system for increasing student writing performance. Eight hundred first year engineering students at VNR VJIET in India are using our system to complete up to twenty writing assignments.

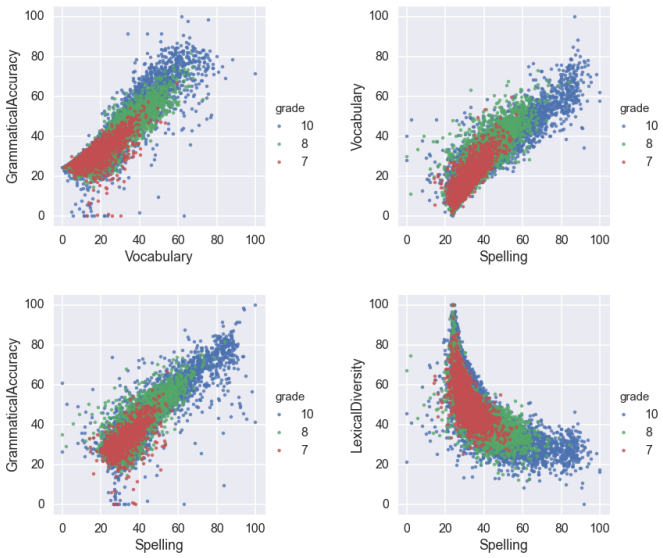

Figure 4: Scatter plots showing two competence scores plotted against each other all essays in the dataset. Data points are colored to distinguish essays from 7th, 8th, and 10th grades.

The current pilot study is an observational study and will use the method of propensity score analysis to determine the effectiveness of the feedback visualizations [6]. Students will also fill out surveys about the feedback they received and its usefulness.

5. ANALYSIS WITH EXAMPLE STUDENT ESSAYS

While the pilot is in progress, to additionally evaluate the usefulness of the writing feedback system, feedback was generated from a dataset containing about 13,000 anonymized student essays which have been graded by humans. The dataset was obtained from the Kaggle competition for automated essay scoring [8] and includes essays for students in 7th, 8th, and 10th grade. A total of eight different groups of essays are contained within the dataset, each with a different writing prompt and grading rubric. For our experiments, all essays were mixed together, grouped only by grade of the student, and all human grades have been normalized to range between 0-100.

First, we investigated correlations between competence types. Figure 4 shows competence vs competence scatter plots for grammatical accuracy, vocabulary, spelling, and lexical diversity. Data points are colored to distinguish between 7th, 8th, and 10th grade essays. Strong linear relationships can be seen for both plots containing grammatical accuracy in addition to vocabulary vs spelling. Additionally, an interesting relationship between lexical diversity and spelling can be seen in Figure 4. This plot shows that no students have high values in both lexical diversity and spelling simultaneously. To achieve high scores in the spelling competence, a longer essay is required with the majority of the words spelled correctly. In contrast, long essays tend to have lower lexical diversity competence values as more words are repeated in longer writings. The resulting balance of these two competences can be clearly seen in Figure 4.

Next, we plotted competence values against human graded scores. Figure 5 shows competence values for connectivity, grammatical accuracy, modifier complexity, and noun phrase complexity plotted against the graded score. Connectivity, grammatical accuracy, and noun phrase complexity all show the trend that increased competence values correlate to higher graded scores. The plot displaying modifier complexity shows the graded score initially increasing with competence value. There is a point which this trend stops and the average score stays constant, or even decreases, as the competence value increases. This data suggests that essays with a lot of complex modifier usage score the same or even lower than corresponding essays with moderate modifier usage. The above analysis gives us confidence in the usefulness of the competence feedback for improving performance.

6. CONCLUSIONS

Providing real-time feedback to students is an important component to solving Bloom's two sigma problem. In this paper we described a scalable learning analytics platform (OpenACRE) which is able to ingest educational data from multiple external systems and provide analytics on that data in near real-time. We demonstrated the usefulness of this platform with the implementation of a writing feedback system and are currently running a pilot research study to evaluate its effectiveness with eight hundred first-year engineering students at a university in India. We also showed that competence values correlated with human graded scores on a set of existing student essays. Development is currently underway to release OpenACRE as an open source project for other educational researchers.

7. ACKNOWLEDGMENTS

This paper is based on work supported by the McGraw-Hill Education Digital Platform Group (MHE DPG). Despite provided support, any opinions, findings, conclusions

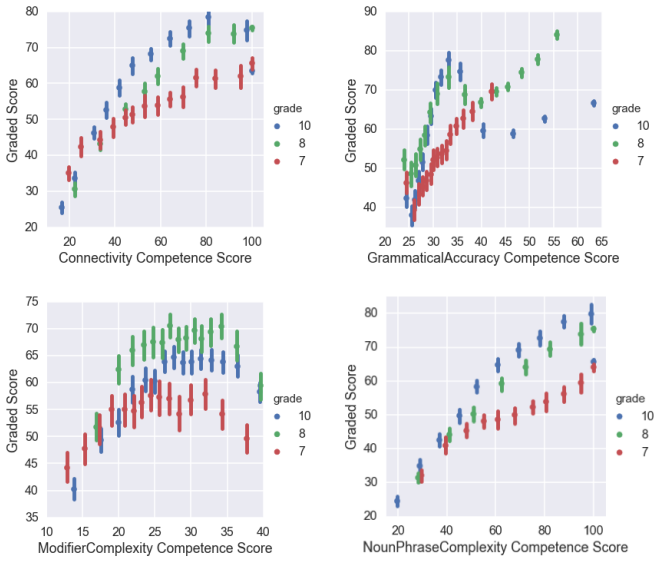

Figure 5: Average essay score as a function of several competence values. Error bars display standard deviation from the mean score. Data points are colored to distinguish essays from 7th, 8th, and 10th grades.

or recommendations expressed in this paper are those of the authors and do not necessarily reflect positions or policies of the company.

8. ADDITIONAL AUTHORS

Additional authors: Mark Riedesel (McGraw-Hill Education, Boston MA, email: `mark.riedesel@mheducation.com`), Alfred Essa (McGraw-Hill Education, Boston MA, email: `alfred.essa@mheducation.com`), Vive Kumar (Athabasca University, Edmonton CA, email: `vive@athabascau.ca`), Kinshuk (Athabasca University, Edmonton CA, email: `kinshuk@athabascau.ca`) and Sandhya Kode (IIIT Hyderabad, Hyderabad, India, email: `sandhya.kode@gmail.com`).

9. REFERENCES

[1] B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, pages 4–16, 1984.

[2] D. Boulanger et al. Scale: A competence analytics framework. In *State-of-the-Art and Future Directions of Smart Learning*, pages 19–30. Springer, 2016.

[3] D. L. Butler and P. H. Winne. Feedback and self-regulated learning: A theoretical synthesis. *Review of educational research*, 65(3):245–281, 1995.

[4] I. G. L. Consortium et al. Learning measurement for analytics whitepaper, 2013.

[5] K. J. Gilhooly and R. H. Logie. Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words. *Behavior Research Methods & Instrumentation*, 12(4):395–427, 1980.

[6] S. Guo and M. W. Fraser. Propensity score analysis. *Statistical methods and applications*, 12, 2015.

[7] J. Hattie and H. Timperley. The power of feedback. *Review of educational research*, 77(1):81–112, 2007.

[8] Kaggle. The hewlett foundation: Automated essay scoring. https://www.kaggle.com/c/asap-aes, 2012. Accessed: 2015-10-01.

[9] V. Kumar et al. Mobile computing and mixed-initiative support for writing competence. *Intelligent and Adaptive Learning Systems: Technology Enhanced Support for Learners and Teachers: Technology Enhanced Support for Learners and Teachers*, page 327, 2011.

[10] C. D. Manning et al. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[11] C. J. Matheus et al. Basevisor: A triples-based inference engine outfitted to process ruleml and r-entailment rules. In *Rules and Rule Markup Languages for the Semantic Web, Second International Conference on*, pages 67–74. IEEE, 2006.

[12] D. S. McNamara et al. The writing-pal: Natural language algorithms to support intelligent tutoring on writing strategies. *Applied natural language processing and content analysis: Identification, investigation, and resolution*, pages 298–311, 2012.

[13] D. J. Nicol and D. Macfarlane-Dick. Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in higher education*, 31(2):199–218, 2006.

[14] Pearson. The research behind writetolearn, 2007.

[15] R. Socher et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[16] R. Software. Tin can api. https://tincanapi.com, 2015.

[17] turnitin. Turnitin revision assistant, 2015.

[18] B. J. Zimmerman. Self-regulated learning and academic achievement: An overview. *Educational psychologist*, 25(1):3–17, 1990.