Grade Me

Authors: Kamran Hussain, Zhaozhong (Alex) Wang, Kaz Nakao
Revision: 5.24.22

Introduction:

We propose a deep learning based, server-client application that autogrades essays. This program consists of 3 sub-applications for each aspect of the pipeline, a student interface, teacher interface, and server all written in Java. The application would include a desktop version for students where they can upload their essays to be automatically graded and can see their previous essays. A component teacher app would also be available where student essay scores can be seen by the teacher. Additionally, the teacher can upload new essay assignments for grading and set a rubric. A third component application would be on the server end. This includes a storage database for essays and loads the BERT (Devlin et al. 2017) binary, which is pre-trained and saved to a file.

Our application tackles the tediousness of grading essays when a lot of the grading elements are redundant. Our target audience is high school students and teachers. By automating grading, students can get rapid feedback allowing them to make corrections and continue learning. This also allows teachers to gather more data on Students progress and modify lesson plans accordingly.

Through BERT, our application uses semantic topic extraction to grade papers. This is essentially creating a mathematical representation of the student document and comparing the meaning of student document sentences to the mathematical representation of the rubric. This allows us to test if the student mentions items outlined in the rubric. If gibberish is inputted, the grader would be able to identify the input essay has no semantic relation to the rubric, and would thus grade the essay accordingly.

Instructions:

Students:

Run the ClientApp jar and enter name and ID number. You will then be able to select from a list of classrooms. From those classrooms, you will be able to see a list of your past submissions. You will be shown a list of past submissions that you have made. You will have the option to view those essays. If you click “submit new assignment”, you will be able to make a submission to an existing assignment in the classroom. You will be given a list of assignments to choose from. From here, select a simple text file or any file that has simple text input to submit to the classroom.

Teachers:

Run the ClientApp jar, and choose the Teacher option. You will be able to choose from a list of classrooms. You will also have the option to create new classrooms in the screen to select a classroom. After selecting a classroom, you can view all the submissions that have been made to any assignment that has been made on the database. Upon selecting a submission that has been made to a particular assignment, you will be able to view the contents of the submission. To create a new assignment, press the “Create New Assignment” button in the screen to select an assignment to view.

Upon clicking, you will be asked to input a name of an assignment. You will also then be prompted to add a csv file for the rubric. The format should be as follows:

	5	4	3	2	1
Criteria 1	Requirements to get 5 points	Requirements to get 4 points	Requirements to get 3 points	Requirements to get 2 points	Requirements to get 1 point
Criteria 2	“”	“”	“”	“”	“”
Criteria 3	“”	“”	“”	“”	“”

The top-leftmost square should be left empty. On the top, the header should contain the criteria in decreasing order. The leftmost column should contain the titles of the grading criteria. The corresponding “cells” inside should contain the requirements for obtaining each score.The number of rows and columns can be changed freely.

Server:

This is not a client configurable pipeline. The server end uses maven dependencies to load the grading model and start the server. The status of the server can be viewed through the server viewer GUI.

Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER):

Must-have Features:

- Be able to upload a text file of a submission to be uploaded to the server under the proper classroom.
- Desktop app uploads and stores student essays in the database to be graded.
- Teacher app can view student scores from the server by accessing a specific classroom. All student assignments and their grades can be viewed.
- Server can load the machine learning binary and grade the submissions that are sent to the firebase server. The machine learning binary is a pretrained model, custom weights and annotations are
- Clients connect to the server and access the specific information that they should be able to access. (i.e. Students can only access their assignments and grades)
- Students will be able to log in using a Name and ID number to access their submissions and grades to the submissions.

Want-to-have Features:

- Plagiarism checks against other student essays by checking copy pasted chunks of text using regular expressions. Also checks for cosine similarity between the students essay vector representation and that of other students. Cosine similarity is the measure of the angle between two vectors. If the angle is 90 degrees, the documents are not similar at all, if the angle is less than 31.5 degrees, the documents are probably plagiarized.

- Allow for pdf submissions. The pdf will be converted to a simple text format and then uploaded to the database.
- Faster server with parallelized processing across multiple CPUs or GPUs
- Dark mode that changes the color of the GUI upon toggle.
- Students will be able to log in automatically by caching the student's data that was entered for the first time into a file. This would include the student name and ID.

Stretch Features:

- Add sentiment analysis and grammar checks to the grading. Add suggestions to the submitted text file and add the suggestions
- Summarize the contents of the essays for the teacher
- Submit essays from google drive through linking google.

Class List:

Data

Classroom - Models a classroom. A classroom has a list of students, teachers, and assignments of the classroom. This is the datatype that is stored in the database.

Rubric - Models an assignment in the classroom. Holds the name of the assignment as well as the grading rubric that is associated with the assignment.

RubricRow - Represents one row of the grading rubric. Carries an ArrayList of Strings. The first entry is the name of the grading criteria. The rest are the rubric grading guidelines.

User - A basic user model. Holds an ID number and a name.

Student - Models a student user of the app. Each student has a list of submissions that they have made. Extends User.

Teacher - A teacher in a classroom. Mostly used as a placeholder class for administrative purposes in classrooms. Extends User.

Submission - A submission made to a classroom. Contains the title of the submission and the content of the submission (the essay). It also contains a date in which the submission was made. Also contains the grade that the submission was given. By default, it is marked as "no grade"

DatabaseChangeListener - A listener for Firebase Database. Triggers different listener methods when actions are performed in the Database such as having items added, removed, modified, etc in the database. Keeps track of the database through a HashMap of the locations of the classrooms in the database and the classroom objects themselves.

DatabaseModifier The class used to push changes and read from the database. Has only static methods as there should not be more than one listener object. There are methods that allow users to push new classrooms to the database and rewrite existing entries.

Client
Main - Client runs from main, contains the Java main method entrance point.

HomeScreen - The basic model for the startup screen. Shows the list of classrooms and adds an option to view the classroom. Asks for a name and ID number upon launching.

StudentScreen - Default screen for students. Shows a list of classrooms after being asked to input a name and id number. Extends HomeScreen

SubmissionScreen - For submitting an assignment. Will prompt the user to choose a file to upload.

TeacherScreen - Default screen for teachers. Shows a list of classrooms. Also has an option to create a new classroom. Extends HomeScreen

ViewSubmissionScreen - Allows a user to view a submission. Shows the contents, title and grade of the submission.

AssignmentViewer - A screen for teachers to select an assignment and view all the submissions or create a new assignment for the classroom.

SelectAssignmentScreen - A screen to let the user choose from a list of assignments to submit to. This only shows the assignments that are in a particular classroom.

StudentClassroomScreen - Shows the student a list of their past submissions to the classroom and a button to make a new submission to the classroom.

TeacherSubmissionViewer - Shows a list of submissions for a particular assignment. The page is separated into a list of ungraded and graded submissions to that assignment. Each submission can be viewed individually.

Server Client
Main - Runs the server application and grades essays

ServerGui - A GUI that shows the status of current training tasks. Displays loss and progress and graphs loss against time.

BertSemanticGraderModel - A BERT Model Version using the Deep Java Library Bi-directional Encoders for Transformers (BERT) Wrapper. This model uses the DistilBERT version of the greater BERT model that is lighter and faster with fewer parameters. We are using BERT's built in tokenizer

to convert the input strings into numerical tokens that capture a feature of the text. The model specializes in the BERT model for classification and similarity calculations.
This class also contains the model itself and the custom classification output layer, custom dual string input tokenizer, and

Grader - Automatically grades essays and checks for plagiarism. This loads the model and updates the students grade. it will loop through the ungraded essays and queue them into the grader. When the model returns, it finds the rubric item with the highest correspondence and assigns that grade. When the calculations are complete, the grade is updated in the database and the next grading batch is queued.

Credits:

- BERT (Devlin et al. 2017)
- Attention is All You Need (Vaswani et al. 2017)
- Kamran Hussain
    - Downloaded, loaded, modified, tuned, and trained the BERT Binary.
    - Added classification layers to BERT to specialize it for classification.
    - Modified the BERT tokenizer to accept multiple sentence inputs separated by a single separator token.
    - Implemented queued grading for model inference and scalable queueing.
    - Implemented plagiarism detection algorithms both semantically and literally.
    - Created training data for the model.
    - Trained the model on a variety of corpora.
    - Created the model server interface for scaled inference and efficient processing given the large resource sizes.
- Zhaozhong (Alex) Wang
    - Handle Firebase structure & information transfer to and from Firebase; made refreshing student/teacher views based on Firebase changes
    - Authored DatabaseModifier, DatabaseChangeListener, ServerGui
    - Managed GUI logic workflow; prevented misinput crashes & added navigation components
    - Implemented dark theme; polished GUI contents and finalized more user-friendly GUI. Fixed bugs in client app
    - Produce/edit presentation video
- Kaz Nakao
    - Wrote all of the data classes that are used by the firebase database
    - Created classes for the client application
    - Created a basic GUI for student and teacher
    - Authored Basic versions of most of the GUI including submissions and classroom selection, etc
    - Finalized classes for communicating with the Firebase Database and the client or server
    - Created training data for the model
    - Javadocs and jaring

- DeepLearning4J
    - Supplied the Word2Vec Model
    - Provided Model saving and loading frameworks
    - Provided Tensorflow and Keras capability in Java
- DeepJavaLibrary
    - Provided a Java interface for working with BERT
    - Provided a machine learning framework for setting up the model
    - Provided a training framework for the machine learning model
- Stanford Natural Language Inference Corpus
    - Provided a majority of the training data corpora
- Apache Commons CSV
    - For CSV input to upload the grading rubric.
- Firebase
    - All libraries associated with communicating between a Firebase server and client.
- FlatLaf
    - Look and Feel for Java Swing applications
    - Adds the darkmode for our application
- Jython
    - A Java library used to interface with the python native BERT Binaries
- Keras
    - Python based deep learning AI loaded in Java through DeepLearning4J that is used to train and load the model.
- John Shelby
    - Firebase demo