Grade Me

Authors: Kamran Hussain, Zhaozhong (Alex) Wang, Kaz Nakao
Revision: 5.13.22

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:

Launch the desktop app and select the Student option. Enter your name and your ID number to log in. There will be options to choose from past submissions that you have under submissions. At the bottom, there is a button to submit a new submission. It will prompt the user to select a file. Then, add a title to the submission in the big text box in the middle.

Teachers:

Launch the desktop app, and choose the Teacher option. You should be able to see all of the past submissions that have been made to the classroom. You will be able to see the scores of each of the submissions as well as their contents.

Server:

This is not a client configurable pipeline. The server end uses a maven built jar to load the grading model and start the server. When the server is up, its status will show up on the client apps.

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 and docx submissions. The pdf and docx 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.

Database Structure

Classrooms

(Classroom) with code name

(Student)

Sadahsduhasuih (Submission)

Class List:

Data
Classroom
Rubric
Student
Teacher
Submission

Client
Main
StudentScreen
SubmissionScreen
TeacherScreen
ViewSubmissionScreen

Server Client
Main
DatabaseChangeListener
DatabaseModifier
ModelLoader
Model
Queue

Credits:

- BERT (Devlin et al. 2017)
- Kamran Hussain
    - Wrote ML model
    - Wrote the code required for the grader
- Zhaozhong (Alex) Wang
    - Handle Firebase structure & information transfer to & from Firebase
    - Authored DatabaseModifier and DatabaseChangeListener
- Kaz Nakao
    - Wrote basic classes for database types
    - Created classes for the client application
    - Created a basic GUI for student and teacher
    - Authored StudetScreen, SubmissionScreen, TeacherScreen, ViewSubmissionScreen so that there are the necessary basic elements for the client application.
- DeepLearning4J
- DeepJavaLibrary
- Firebase
- John Shelby

- Firebase demo