The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center
Portland, OR . June 23 - 26, 2024
Paper ID #41462

An Interactive Platform for Team-based Learning Using Machine Learning Approach

Tony Maricic, New York University Tandon School of Engineering
Nisha Ramanna, New York University Tandon School of Engineering

Nisha Ramanna is a student at New York University, pursuing her Bachelor's and Master's in Computer Science with a concentration in Machine Learning and Artificial Intelligence. She is passionate about all areas of Machine Learning, including Natural Language Processing.

Alison Reed, New York University Tandon School of Engineering
Dr. Rui Li, New York University

Dr. Li earned his master's degree in Chemical Engineering in 2009 from the Imperial College of London and his doctoral degree in 2020 from the University of Georgia, College of Engineering.

Jack Yang, New York University Tandon School of Engineering

An Interactive Platform for Team-based Learning Using Machine Learning Approach

Abstract

This complete evidence-based paper explores the feasibility of developing an interactive platform with chatbot feature to facilitate project-based learning. Teamwork pedagogy is widely used in engineering courses, particularly in first year (cornerstone) and senior-year (capstone) design courses, but also across the curriculum. Faculty have several aims for teaching in teams, one of which is to improve students' collaborative abilities. Engineering expertise, as well as pedagogical goals such as greater learning and motivation, are under consideration when building an effective team pedagogy. CATME, and other platforms have long been used to facilitate the process of monitoring team performance. The comprehensive data that the platform provided has enabled faculty members to analyze the problems in detail. Also, it is very helpful when documenting the team performance from year to year. At New York University, 700 students are taking a fundamental engineering course on an annual basis. The students are asked to form project teams after the first two weeks and work on a semester-long project on a weekly basis. Overall, there are 50-60 teams each semester. CATME has been implemented to monitor the team's progress. It has been reported by the faculty members that it took time to evaluate the students' peer comments and ratings as there are 2000 - 3000 comments each semester. Human errors can also occur when reviewing those comments. To reduce the workload of faculty members for analyzing the student comments and taking actions accordingly, an interactive team-monitoring platform is built to serve the purpose. This platform consists of two major components, which is built on React and Fast API. The platform can potentially be integrated with CATME or other team-monitoring software. A group of CATME users were asked to try out the platform and fill in a user experience survey. The survey results gave some constructive feedback for the developers. Overall, the project can deliver a feasible solution for course instructors to handle many student project teams. In the future, a generative AI feature - CHATME will also be available on the front end to help the user check the status of each student group, which is built using NLTK and TensorFlow. Moreover, if a team issue arises, the platform will alert the users, and provide constructive suggestions on how to improve the group performance.

Introduction

In engineering education, fostering collaborative skills [1] among students is crucial, and team-based learning has become the primary approach. It is an approach particularly prevalent in foundational courses, such as first-year cornerstone courses and senior-year capstone design courses, but it also finds application across the entire engineering curriculum. The overarching goal for implementing team-based learning is to enhance student's abilities to work effectively in groups[2], aligning both with the demands of their future professional endeavors and broader educational objectives.

Platforms like (Comprehensive Assessment of Team Member Effectiveness) have played a significant role in streamlining team performance monitoring [3]. It provides a platform for students to report team issues, enabling faculty members to understand the team dynamics. This facilitates the analysis of strengths and areas for improvement. This data also allows faculty to track team performance trends throughout the academic semester.

A language is a collection of rules and symbols used to convey or broadcast information. Natural Language Processing (NLP) is ideal for users who lack the time to learn and master machine-specific languages. NLP is a field of Artificial Intelligence and Linguistics that aims to help computers understand human language statements and words [4]. Natural Language Processing (NLP) consists of Natural Language Understanding and Natural Language Generation. Its purpose is to facilitate user interaction with computers using natural language. Natural language processing (NLP) has recently gained popularity as a method of computationally interpreting human language. NLP has a wide range of applications, including translation and answering questions[5].

Approximately 700 students enroll in a fundamental engineering course annually at New York University. The cornerstone of this course is a team-based semester-long design project. Each semester, 50-60 teams are formed, necessitating robust tools for team monitoring. While CATME has reliably supported the team ratings, the high volume of student comments—ranging from 2000 to 3000 per semester—has presented challenges for faculty evaluation and introduced the potential for human errors. To address this issue, a responsive and interactive team-monitoring platform has been developed. This platform comprises two major components, using React to develop the front end and Django for the backend. Designed with scalability in mind, the platform is designed to seamlessly integrate with existing team-monitoring software, enhancing its versatility.

In essence, this platform represents a dynamic evolution in team-based learning, addressing the challenges associated with large-scale student collaborations. Through its innovative features and integration of generative AI, it not only streamlines the evaluation process for faculty but also empowers students with a tool that enhances their collaborative experience.

Experimental Methods

Front End

In the development of the front-end for the interactive platform, a set of methods were utilized to ensure a smooth user experience. The core framework for the front-end development was

React.js, chosen for its efficiency in building dynamic user interfaces. To integrate user data from .xlsx or .csv files , the SheetsAPI from Google was employed, serving as a robust tool for retrieving raw data from the file. This connection facilitated the real-time synchronization of team-related information. Using React.js form input box, users could input specific group IDs. This input was crucial for identifying and retrieving relevant team-related data from sheetsAPI , with the potential to post the group ID to the FastAPI backend, enhancing the platform's interactivity and responsiveness. Visual representation of data was achieved through the incorporation of Chart.js, a powerful charting library. This enabled the generation of graphs to provide users with insightful visualizations to comprehend team dynamics and performance effectively.

Virtual Assistant

To optimize ease of use, we initially decided to create a chatbot user interface. Our goal was to allow users to prompt the chatbot with a question about a project or student and, in response, the chatbot would identify and return the relevant information. The chatbot classifies the intent of a user prompt using a Tensorflow Sequential model. The neural network is composed of one embedding layer, a global average pooling layer, and 3 dense layers (the first two have 16 nodes and use Relu as an activation function, the output layer has a node for each possible user intent and uses a softmax activation function). After intent classification, we use NLTK to perform Name Entity Recognition, extracting the project id or student name from the user's query. Now that we have the intent and subject of the query, we make the appropriate call to the API and return the results to the user. Ultimately, we hope to provide summaries of student feedback, flag students who are struggling, recommend students who are performing especially well as potential project leaders, and offer professors advice on how to advance a given project based on the student feedback.

Back End

Once classification is complete, we can then provide the necessary information needed for course and individual analysis. This is fed to the front end via multiple endpoints. For model training, 800 student comments were imported from CATME, then were broken down into sentences, and fed through a text classifier to build a list of training examples for both the efficiency and deficiency models. These sentences were then reviewed and correctly classified by a team of students. The results of our model can be found in the next section.

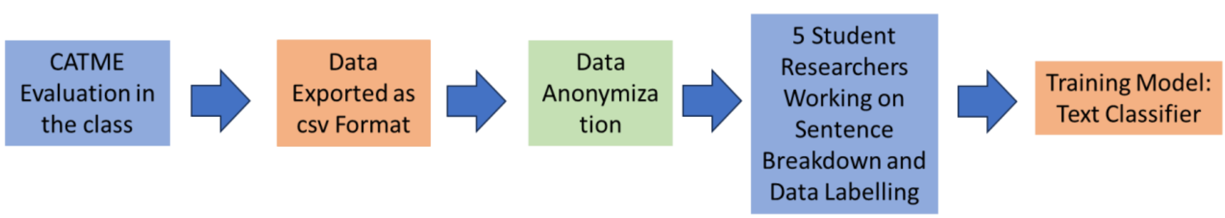


Figure 1. Workflow for model training and outcomes

Multi-label text classification and sentiment analysis are the methods used in the back end to analyze the large quantity of student submitted text. First, sentiment analysis is done using the standard NLTK Sentiment Intensity Analyzer. Upon processing these results, we then send the text over to one of two classification models depending on sentiment. The multi label text classifier was trained using Tensorflow's Keras model[6]. This is a neural network containing 5 layers: Embedding Layer, Convolutional Layer, Global Max Pooling Layer and two dense layers(one using ReLu as an activation function with the output layer using sigmoid). The classification labels used in this study: "Contributing to the Team's Work", "Interacting with Teammates", "Keeping the Team on Track", "Expecting Quality", and "Having Relevant Knowledge, Skills, and Abilities".

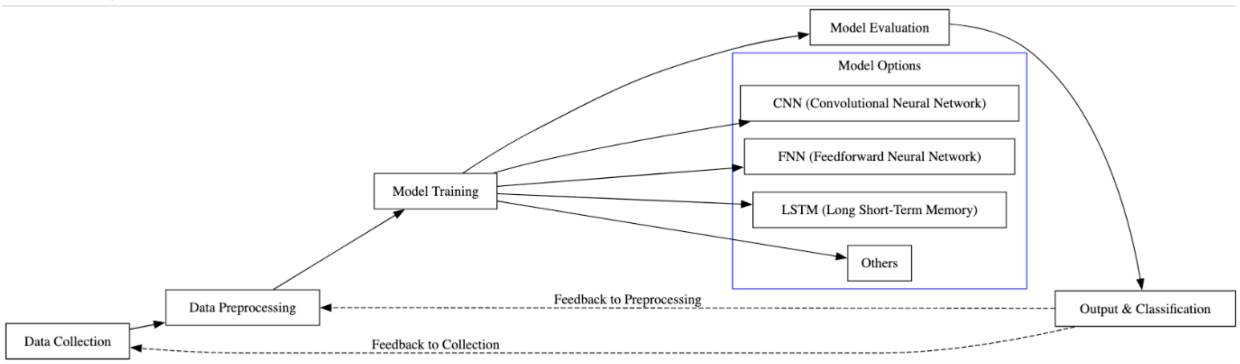


Figure 2. Training models used in this study

One example of the model structure that has been used in this study is shown in Figure 3.

```
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 30, 50)            73400

 conv1d (Conv1D)             (None, 28, 64)            9664

 global_max_pooling1d (Glob  (None, 64)                0
 alMaxPooling1D)

 dense (Dense)               (None, 32)                2080

 dense_1 (Dense)             (None, 5)                 165

=================================================================
Total params: 85309 (333.24 KB)
Trainable params: 85309 (333.24 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 3. The CNN model structure used in this study

Results

Front End Display

The website successfully retrieved data from a Google Sheet, identified by its unique spreadsheet ID. Utilizing the fetch API, the application fetched the necessary data and parsed the JSON response. The data, organized in rows, was then processed to extract relevant information. To enhance user interaction, the application allowed users to input a project group ID. Subsequently, the system queried the fetched data to identify and extract raw comments associated with the provided project group ID. The retrieved and processed data was presented in a tabular format. The table included three columns: "ID," "Raw Comments," and "Processed Results." The "ID" column displayed the project group ID entered by the user, providing context for the subsequent data. The "Raw Comments" column contained the extracted comments associated with the specified project group ID. The "Processed Results" displayed the comments from the back end using the GET method from FastAPI. In instances where no data was available for the provided project group ID, a user-friendly message was displayed to communicate the absence of relevant information. This feature aimed to enhance user experience by providing clear feedback in cases where the system couldn't retrieve matching data. The graphical representation of the results took the form of a pie chart generated using the Chart.js library. The chart visually depicted sentiment distribution based on predefined categories. Each category was assigned a distinct color, contributing to a clear representation of sentiment proportions within the dataset. The application seamlessly integrated both tabular and graphical representations, providing users with a comprehensive view of the sentiment analysis results. The user interface allowed for real-time interaction, enabling users to input different project group IDs and observe the corresponding changes in the displayed results.

(a) Chatbot design (b) User Interface

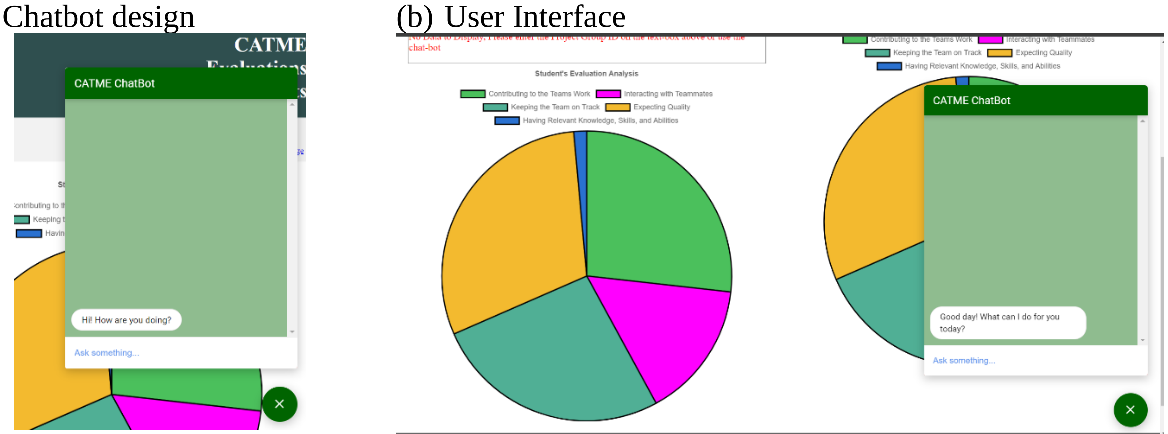


Figure 4. Proposed Chatbot Interface CHATME.

Framework for Virtual Assistant

As we built the chatbot, we began to question if a chatbot would provide the easiest and most intuitive user experience. A chatbot would require users to type the project ids and/or first and last names of students whose status they wanted to check. Users would also need to know what functions our web app offers and specify the type of information that they were interested in, be it summary of student feedback, advice, or identifying struggling or excelling students. The user would also have to prompt the chatbot multiple times for each project to access the full information about the project that our site provides.

Once the user uploads their export of student feedback from CATME, we have all the information we need from the user to provide our site's services. Instead of continually prompting the user for more information with a chatbot, we decided that a standard web interface would offer a better user experience. The user's input is limited to a file input box at the top of the site. Every project from the user's uploaded files will be listed below with multiple sorting options - alphabetical, by date, by status. The user can click on a project id to expand it. The summary of the student feedback for the project and for each team member will be listed below along with potential student leaders in the group, struggling students, and advice for the user.

Dataset and Preprocessing

The dataset used in this study was shown in Figure 5. The student comments were directly imported from a Google Sheet. The raw data was imported from CATME. The comments then manually broken down into sentences for data classification.

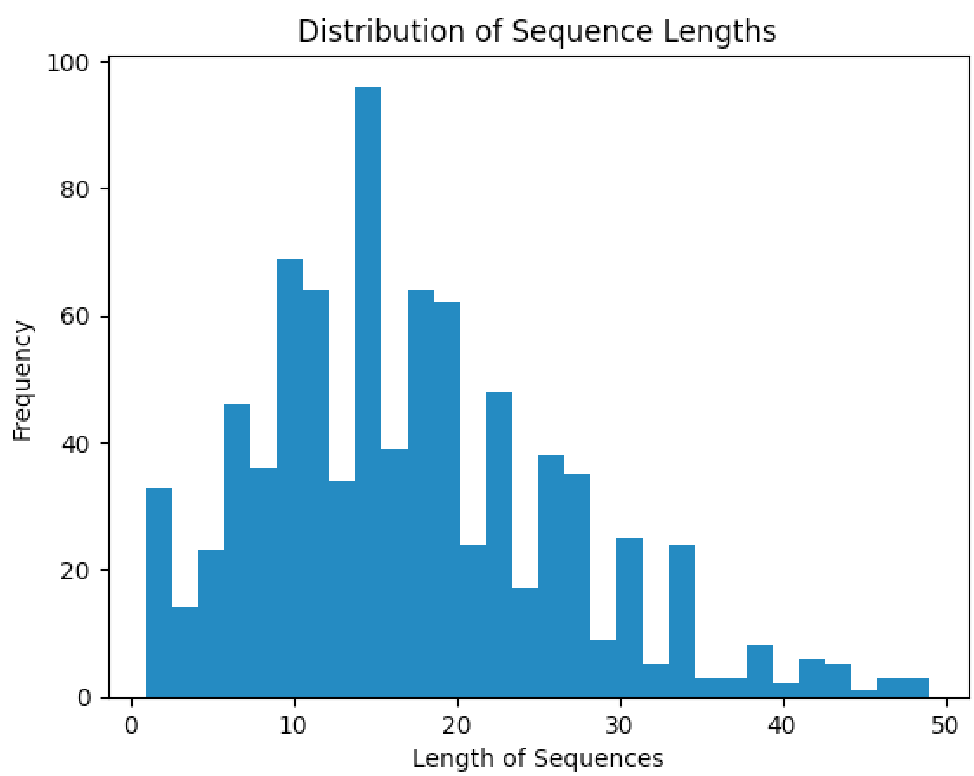
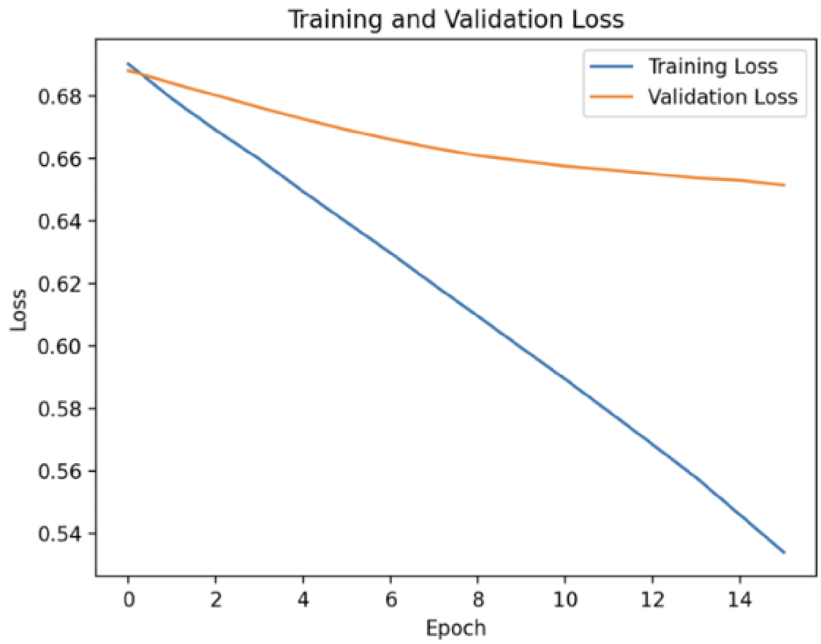


Figure 5. Data distribution in terms of the length of the student comments.

Training and Validation

The results of the classification leave some room for improvement. The initial testing has 73% accuracy. Some examples of the initial training results are shown in Figure 6.

(a) Training Result 1



(b) Training Result 2

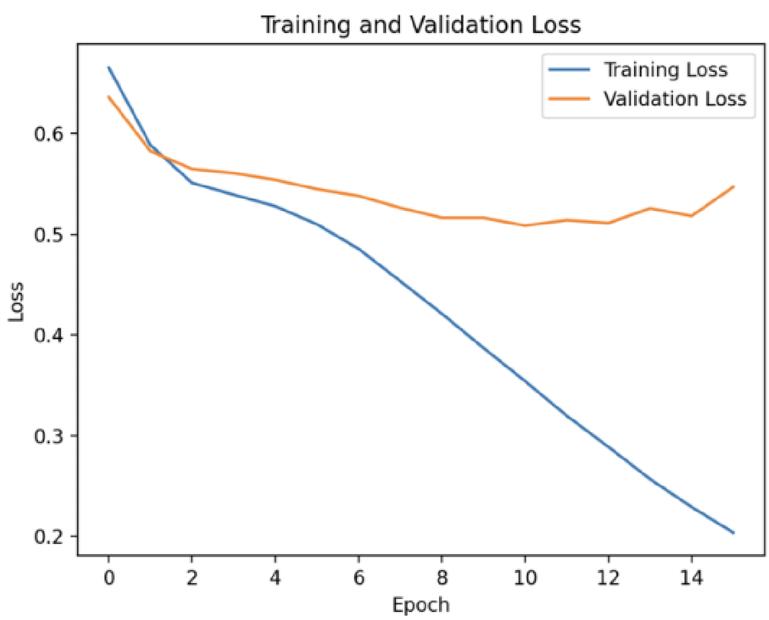


Figure 6. Two sets of training results using the Tensorflow's Keras model[6].

Field Test Validation

Hamming Loss (HL) Equation was used for measuring accuracy:

$$HL = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \Delta Z_i|}{|L|} \quad (1)$$

Where,

(x_i, Y_i) is instances of multi-label dataset for i = 1,2,3….,m

$Y_i \subseteq$ L is the set of true labels

L is the set of all labels

Z_i is the set of labels that are predicted by an algorithm

Precision, Recall, and F1-Score are calculated for each label and then averaged. Moving forward, more iterations of CNN and FNN would be necessary to see if the hamming loss could be improved, which is already close to 80%. For training, 640 data points were used. For testing, 160 data points were used. In theory, prediction becomes more accurate with more data input. In other words, it is possible to continue to maximize its learning efficiency.

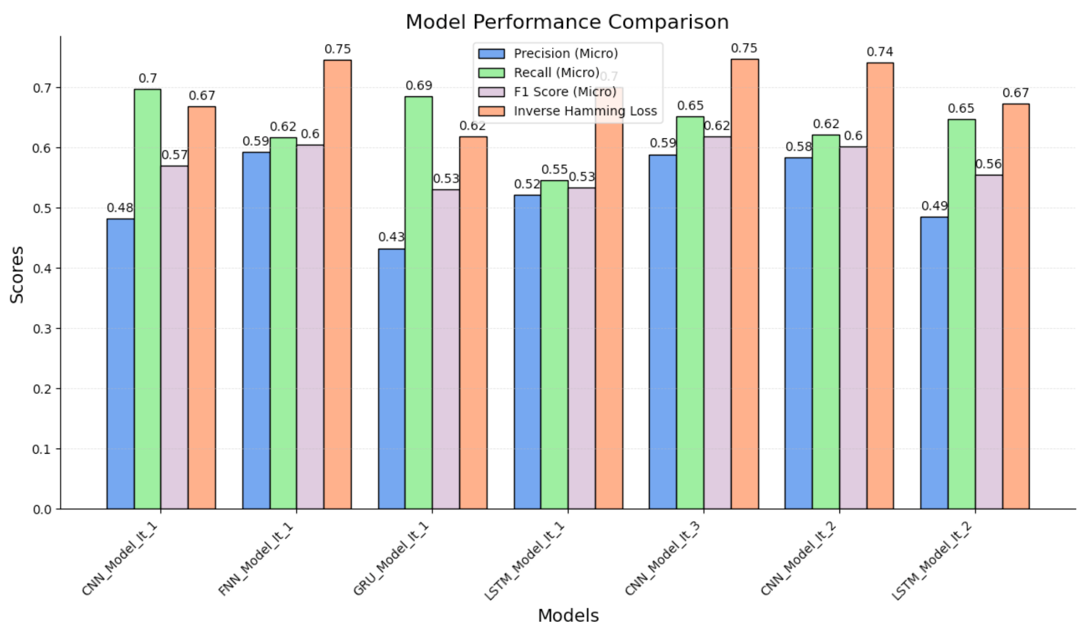


Figure 7. Training results with different models: CNN, FNN, GRU, and LSTM.

Discussion

Model Training

The current model only has 73% accuracy on initial testing. Currently, 800 written sentences were used in this study. Although this seems like a lot, given the nature of classification, this is not enough to properly fit a model with this level of complexity. We need more training examples to improve accuracy. Given the fact that many different people are classifying these training examples, there is bound to be some level of bias when it comes to which categories people feel a particular sentence belongs to. A stricter rubric for classification should be developed for the future. Although the hyperparameters is good, there is always room for improvement.

Potential benefits and risks

There are many benefits this system could offer. The instructor could monitor the entire class via the pie chart shown in Figure 4. The virtual assistant offers direct support to the instructor to look into the performance of specific project groups. On the other hand, the student could use the virtual assistant to plan out project ideas, time management, meeting schedule and personal research assistant. However, some of the functionality is still under development and the details will not be included in this paper.

Limitations

The current platform is not running as a fully integrated system. There are several limitations in the study. First, the dataset labelling is not equally distributed. Second, the student researchers who performed the data labeling could have different opinions. For example, if a student commented, “She helped to manage the group activity”, which could be interpreted as “contribution to the team” and “keep the team on the right track” by one researcher, or just be interpreted as “keep the team on the right track” by another researcher. On the other hand, if the validation error increases with time or if the training error is significantly smaller than the validation error, the network could overfit the training data and is becoming less generalizable.

Future work

This platform will incorporate a generative AI feature[7, 8], created using NLTK and TensorFlow. This addition aims to provide real-time insights into the status of each student group for faculty members. Moreover, when team issues arise, the platform will proactively alert faculty members and provide constructive suggestions for improving group performance.
We plan to make a generative chatbot that can assist students in group projects. Meanwhile, more data collection with unbiased labeling and synthetic data. Another possibility is to adjust the number of classes in order to improve the classification accuracy and efficiency.

Some functions which the chatbot could offer:
Project Ideas: Students can submit their area of interest and the skills they would like to use and the chatbot will return potential project ideas.
Time Management/Schedule Creation and Maintenance: Students can enter their project goal, project deadline, number of group members, and each member’s weekly time commitment to the chatbot and the chatbot will: break down their project into a series of tasks, and appropriately schedule each task so that the students meet their deadline; assign subtasks to each group member, taking their weekly time commitment into account; suggest whether completing the project by the deadline with the given number of group members is realistic or if the project is ambitious enough to keep all group members busy up to the deadline; if the project is unrealistic or lacks ambition, suggest alterations to the project; flag if students fall behind and revise the schedule accordingly; highlight internal deadlines, which if missed, will jeopardize the completion of the project by the final deadline; send reminders to students prior to deadlines; alert the professor if a student has missed a certain number of deadlines; alert the professor if the group is in danger of missing their final deadline.

Schedule Meetings: The chatbot can collect all group members' individual schedules and return ideal meeting times; The chatbot can also schedule meetings with the group and the professor, with individual members and the professor, or with a subset of members.
Research Assistance: The chatbot can point students towards relevant resources.
Note Taking: If we build voice recognition into the chatbot, it can listen in on meetings, take notes, and update its other offerings; students can submit feedback about their project and group members to the professor through the chatbot.

Conclusion

An initial platform has been built to test the feasibility of utilizing student team comments to build a classifying model. The interactive chatbot feature has been explored. This platform uses React for the front end, Fast API as the back end, and the model training is on Tensoflow Keras . The platform can potentially be integrated with CATME or other team-monitoring software. The current model only has 73% accuracy on initial testing with 800 sample comments. A future improvement would be to use more and balanced data. A generative AI feature - CHATME - will also be available at the front end to support the course teaching.

Acknowledgment

This study is under the IRB Exemption: IRB-FY2022-6435.

References

[1] K. Bosworth, "Developing collaborative skills in college students," *New directions for teaching and learning,* vol. 59, pp. 25-31, 1994.
[2] P. R. Harris and K. G. Harris, "Managing effectively through teams," *Team Performance Management: An International Journal,* vol. 2, no. 3, pp. 23-36, 1996.
[3] M. L. Loughry, M. W. Ohland, and D. J. Woehr, "Assessing teamwork skills for assurance of learning using CATME team tools," *Journal of Marketing Education,* vol. 36, no. 1, pp. 5-19, 2014.
[4] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *Multimedia Tools and Applications,* vol. 82, no. 3, pp. 3713-3744, 2023/01/01 2023, doi: 10.1007/s11042-022-13428-4.
[5] N. S. Khan, A. Abid, and K. Abid, "A novel natural language processing (NLP)–based machine translation model for English to Pakistan sign language translation," *Cognitive Computation,* vol. 12, pp. 748-765, 2020.
[6] N. K. Manaswi and N. K. Manaswi, "Understanding and working with Keras," *Deep learning with applications using Python: Chatbots and face, object, and speech recognition with TensorFlow and Keras,* pp. 31-43, 2018.
[7] D. Baidoo-Anu and L. O. Ansah, "Education in the era of generative artificial intelligence (AI): Understanding the potential benefits of ChatGPT in promoting teaching and learning," *Journal of AI,* vol. 7, no. 1, pp. 52-62, 2023.
[8] E. Brynjolfsson, D. Li, and L. R. Raymond, "Generative AI at work," National Bureau of Economic Research, 2023.