# ITS-Chatbot Spring 2021 Project Proposal Draft

## Group Membership

| Member | | |
|---|---|---|
| Shen-En Chen (achen353@gatech.edu) | Skills | Python; familiar with machine learning algorithms and framework/libraries (e.g. scikit-learn, Tensorflow/Keras, PyTorch) |
| | Responsibility | Research on methods to remove redundant text aside from using vector similarity scores |
| Phat Tran (ptran74@gatech.edu) | Skills | Java: Data Structures and Algorithms;<br>Python/HTML/PHP: basics |
| | Responsibility | Update requirements.txt file; get familiar with the chatbot implementation and Jupyter Notebook/Jupyter Lab/Google Colab |
| Zihuan Wu (zwu372@gatech.edu) | Skills | Java, Python: some experience with machine learning projects<br>SQL: basics |
| | Responsibility | Get familiar with the chatbot implementation |
| Elias Reta (ereta3@gatech.edu) | Skills | Java, Python, HTML/CSS, familiar with Datastore in Google Cloud |
| | Responsibility | Get familiar with the chatbot implementation and Jupyter Notebook/Jupyter Lab/Google Colab |

## Project Goals

The main purpose of ITS is to help students succeed in the class by providing or directing them to the resources based on the strengths and weaknesses of the student in regards to the content of the course. While TAs serve as an integral resource for students, the team believes that a chatbot can significantly reduce the workload of TAs by providing data-driven responses to students' questions. Our goal is to develop a chatbot that will help TAs to handle high volumes of questions during the course and especially before deadlines and exams where the number of posts typically increases as well as provide a more personalized experience.

While the chatbot team has made significant progress in the past few semesters, the chatbot only performs fine on logistical questions. As a result, we have brainstormed and ranked a few milestones/phases for the project based on how needed the functionality or the work is for the chatbot implementation so far. For this semester, we decided to focus on three goals: (1) remove repetitive question-answer pairs from Piazza data reduce confusion by the model, (2) optimize the code to speed up transformer-based models, and (3) build a classifier for question types–Logistical Questions, Conceptual Questions, and Reasoning Questions–to tackle different questions more specifically and enabled phase deployment if future VIP members are to work on integration with ITS. We have provided a more detailed timeline below.

## Milestones

Note: All the weeks are based on the official VIP-ITS schedule.

| Phase | | |
|---|---|---|
| 1 | Week 5 - 7 | Remove repetitive question-answer pairs from Piazza data |
| | Week 5 | Research on methods to remove repetitive text |
| | Week 6 | Prototype using Jupyter Notebook/Jupyter Lab |
| | Week 7 | Integrate the data cleaning code to the chatbot program |
| 2 | Week 8 - 10 | Optimize the code to speed up transformer-based models |
| | Week 8 | Research on the time and space efficiency of different data structures and function calls that are currently being used in the chatbot program |
| | Week 9 | Prototype using Jupyter Notebook/Jupyter Lab and document the time/space saved compared |
| | Week 10 | Update the chatbot program with the more efficient implementations |
| 3 | Week 10 - 14 | Build a classifier for question types: Logistical Questions, Conceptual Questions, and Reasoning Questions |
| | Week 10 | Research on ways to automating the labeling process for the Piazza entries |
| | Week 11 | Label the Piazza entries manually/automatically depends on the research result |
| | Week 12 | Label the Piazza entries manually/automatically depends on the research result |
| | Week 13 | Label the Piazza entries manually/automatically depends on the research result |

| | | |
|---|---|---|
| | Week 14 | Build and evaluate a question-type classifier |
| 4 | If time permits | Incorporate math embeddings with current Word2Vec document embeddings |
| 5 | If time permits | Establish 3 different chatbot workflows for three different types of questions |
| 6 | If time permits | Implement a keyword quick search functionality for simple and straightforward questions |
| 7 | If time permits | Establish a database for more efficient data storage and retrieval |

# Project Description

## Problems

Here is a brief explanation and reason for each of the ideas proposed above:

1. **Remove repetitive question-answer pairs in the Piazza datasets**
   We focused more on building the model last semester and thus did not put much time on cleaning the Piazza data further when we collected more semesters of CSV files. Removing repetitive entries would allow us to remove some noise from the model and hopefully achieve better performance.

2. **Optimize the code to speed up transformer-based models**
   As we discovered last semester, Transformer-based models perform generally better than the Word2Vec model built in Spring 2020. But on a typical Intel i5 core, it takes a bit longer to generate the answer. On Intel i7 10700K CPU, the processing time is significantly reduced. However, if we are going to deploy it on some server in the future, we can expect the server to run super fast to compensate for a large amount of processing required.

3. **Build a classifier for question types (Logistical Questions, Conceptual Questions, and Reasoning Questions)**
   Since a transformer-based model cannot handle all different types of questions, we can establish different model workflows for each type of question and improve the model on one type of questions at a time. This also should improve the time efficiency since if we can determine the type of the input question then we can direct the model to only a subset of the data we have when searching for candidate contexts.

4. **Incorporate math embeddings with current Word2Vec document embeddings**
   Very little research has been done on converting math equations into vectors as we did

for text. We might not be able to get to this idea this semester. Some relevant research includes *Math-word embedding in math search and semantic extraction* (https://link.springer.com/article/10.1007/s11192-020-03502-9#Sec5) and *Equation Embeddings* (https://arxiv.org/pdf/1803.09123.pdf).

5. **Establish 3 different chatbot workflows for three different types of questions**
   Combining all the 4 ideas above, we can make the chatbot execute different tasks based on input question type:

   a. Logistical Questions:
      i. Use the current Dual Transformer to retrieve an answer directly
   b. Conceptual Questions:
      i. Use Dual Transformer to retrieve an answer
      ii. If the answer is not good enough by some metric, use improved Word2Vec to find the most similar textbook paragraph and refer the user to that specific chapters and sections of the textbook
   c. Reasoning Questions:
      i. Use improved Word2Vec to find the most similar textbook paragraph and refer the user to that specific chapters and sections of the textbook

6. **Implement a keyword quick search functionality for simple and straightforward questions**
   This is a function that is widely available on the chatbot plug-ins of many communication software such as Slack and Discord. When our chatbot is performing well enough, we would like our users to be able to perform quick searches on simple questions like “Syllabus”, “Schedule”, “Due Dates” etc.

7. **Establish a database for more efficient data storage and retrieval**
   The reason for a database is simple: more efficient data storage and retrieval. With a database, we hope to make the storage more extendable and more efficient with reduced processing time for the model.

## Potential Solutions

We are currently researching potential solutions to each of the milestones. For Milestone 1 and 2 in particular, we have thought of a potential solution to each.

For Milestone 1, a possible solution is to neglect any symbols in the text first and convert the entries into embeddings. We can use the Word2Vec model to compare the similarity and remove repetitive questions or answers. To evaluate this, we will use human judgments and test the model with questions that are highly likely to have been responded to many times each semester. By inspecting the top-k answers that the chatbot calculated, we can see if we have removed enough repetitive answers and questions.

For Milestone 2, a possible solution is to make sure we are using more efficient data structures and operations such as Numpy Array over Python Lists.

## Foreseeable Challenges

The greatest foreseeable challenge in our timeline lies in Milestone 3. Given that a supervised learning task requires the data to be pre-labeled, we will need to find an efficient way to label our Piazza entries into three different question types as defined above.

## Implementation and Collaboration

This project is mainly developed in Python and will continue to be so. We will fork the Chatbot-v2 repository on Github and manage our files in a new repository. Each member should create their own respective branches on the repo and add more if needed. Development should be done in member's respective repositories and merged into the master branch on a weekly basis. Review by at least one team member other than the Pull Request initiator is required to merge changes into the master branch. For communication, we will use Microsoft Channels for messages and video calls for weekly internal meetings.