ITS-Chatbot Fall 2021
Project Proposal

Group Membership

Member		
Phat Tran (ptran74@gatech.edu)	Skills	Java, Python, familiar with machine learning (NLP) algorithms
	Credit	2 credit hours (8 hours/week)
	Responsibility	Separate the current ChatBot process and help to create an API
Aahan Kerawala (akerawala3@gatech.edu)	Skills	(in order of proficiency): Java, python, JavaFX, C
	Credit	1 credit hour (3-4 hours/week)
	Responsibility	Work on improving the chatbot system by ensuring accurate results from the textbook with the piazza posts to simply support the primary response as more relevant questions and responses that could relate to the initial inquiry. As well as potentially testing new algorithms to improve confidence levels of responses from the textbook, improving relevancy of textbook excerpts.

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.

In the last semester, we attempted to remove repetitive question-answer pairs from Piazza data by using different topic modeling techniques such as LDA and CorEx. For this semester, however, we take a different approach in order to improve the Chatbot outputs. The current Chatbot program will be separated into two different processes to answer different types of questions. The first process uses a transformer-based model to answer input conceptual

questions using the textbook paragraphs only. The second process uses the Word2Vec model that compares and outputs the most similar Piazza discussion posts to the input query. In order to achieve these two processes, we decided to focus on three goals: (1) modify the DataCleaner class to satisfy the paragraphs format that the algorithms such as BERT and Word2Vec work best in, (2) separate the Piazza posts and textbook paragraphs sources into two different algorithms to produce two distinct results, and (3) create an API to receive a user query and send the output. We have provided a more detailed timeline below.

Notice: if we have more time before the end of the semester, we will research or solve the problems that were not completed last semester.

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

- Week 1-3: Project planning, team building, proposal drafting
- Week 4-6: Familiarize with the ChatBot processes and NLP concepts.
    - Fix the error in the Word2Vec model.
    - Get familiar with tools and models.
    - Get familiar with the current processes of Chatbot
    - Learn the underlying concepts of the ChatBot and work on the DataCleaner class (understand the data format that Word2Vec and BERT wants).
    - Prototype using Jupyter Notebook/Jupyter Lab and show what the output would look like and plan how to change them.
- Week 7-9: Implement changes to the core code
    - Integrate the data cleaning code
    - Separate the transformer-based information retriever and Word2Vec model.
    - Integrate the two new models of the chatbot program
    - Explore and evaluate different pre-trained Question Answering Model on huggingface.co
    - Explore and evaluate different types of BERT (SciBERT, DistillBERT, RoBERTa, XLNet, etc.)
- Week 10-14: Create an API and integrate with TutorJS
    - Create an API to receive a user query and send the output
    - Explore and evaluate different pre-trained Question Answering Model on huggingface.co
    - Explore and evaluate different types of BERT (SciBERT, DistillBERT, RoBERTa, XLNet, etc.)
- Week 14-17: Breaks and Final Presentation

Project Description

Problems

Here is a brief explanation and reason for each of the ideas proposed above. They are in the order of importance, and some problems are from last semester. We will focus on the first three

problems, and if we complete those problems before the end of the semester, we will work on problem #5 or #4.

1. **Update the DataCleaner class**
   We discovered that different algorithms like to handle the format of paragraphs differently. For instance, the Word2Vec model performs better when all stopwords are removed, while the transformer-based model likes to keep all words in the paragraphs to use as context. How these models perform depend greatly on the dataset that they were trained on, so we will need to inspect and mimic the dataset format.

2. **Separate the output of the Word2Vec and transformer-based model**

   The current system returns an answer with the highest confidence score from both models. It tries to find the answer from the textbook paragraphs first, and if the confidence score of that answer is not high, then it will find the answer from the Piazza discussion posts. The problem with this approach is the repetition of the question-answers pairs. A question about a certain topic can be asked multiple times, and choosing one post out of many can lead to incomplete or wrong answers.

   In order to solve the problem mentioned above, we separate the current system into two processes. The first one would answer the conceptual questions from users using the textbook paragraphs, and the second one would produce a list of similar question-answers posts compared to the input query using Piazza data. Then, the user will search the result to find the appropriate answer.

3. **Create an API**

   In order to integrate the Chatbot with other projects, we need to create an API to simply call the bot and get the result. The output would be stored in JSON (or similar) format.

4. **Optimize the code to speed up transformer-based models** (from last semester)
   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 the 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.

5. **Establish a database for more efficient data storage and retrieval** (from last semester)
   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.

6. **Incorporate math embeddings with current Word2Vec document embeddings** (from last semester)
   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).

7. **Build a classifier for question types (Logistical Questions, Conceptual Questions, and Reasoning Questions)** (from last semester)
   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 question 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.

Foreseeable Challenges

The greatest foreseeable challenge in our timeline lies in Milestone 1. We are a little bit new to the Chatbot project so we might spend more time than intended.

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 text messages and Microsoft Channels for messages and video calls for weekly internal meetings.