ITS-Chatbot Spring 2022
Project Proposal

Group Membership

- Subteam 1:
    - Members: Aahan, Yueqiao (Christina), Pat
    - Tasks:
        - Exploring more algorithms that do not need to use the Transformers model (GPU)
        - Test different questions of the Chatbot
            - Examine the outputs of the retrieving and extracting processes
- Subteam 2:
    - Members: Rishi, Lukas, Jin, Devang
    - Tasks:
        - Explore and implement vector search engine
        - Clean up existing code
        - Add more features to the existing TutorBot

Member		
Phat Tran (ptran74@gatech.edu)	Skills	Java, Python, C
	Credit	2 credit hours (8 hours/week)
	Responsibility	Assist and answer questions from new team members, implement a new model that does not depend on the GPU
Aahan Kerawala (akerawala3@gatech.edu)	Skills	Java, python, JavaFX, C
	Credit	2 credit hours (8 hours/week)
	Responsibility	Improve the accuracy and speed of the chatbot with newer models
Yueqiao Chen (ychen3221@gatech.edu)	Skills	Java, Python, C (beginner), JavaFX
	Credit	1 credit hours (4 hours/week)
	Responsibility	Explore the Tutorbot and related models

Jinwoo Park (jpark955@gatech.edu)	Skills	Javascript, React, node.js, Java, C
	Credit	2 credit hours (8 hours/week)
	Responsibility	Fix chatbot image search, evaluate vector search databases
Lukas Olson (lolson6@gatech.edu)	Skills	Python, ML/DL, Docker/K8s
	Credit	2 credit hours (8 hours/week)
	Responsibility	Evaluate vector search databases, possibly assist with deploying on GA Tech server with Docker
Rishi Nopany (rnopany3@gatech.edu) 11:30	Skills	Python (Flask), JavaScript (React.js)
	Credit	1 credit hour (4 hours/week)
	Responsibility	Exploring and implementing alternative vector search engine (Weaviate)
Devang Ajmera (devangajmera@gatech.edu) 11:30	Skills	Java, Python, SQL
	Credit	1 credit hour (4 hours/week)
	Responsibility	Clean up existing code, Add more features to the existing TutorBot

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 have implemented a new algorithm called BM25 to retrieve relevant documents to feed it to the Transformers model to extract answer(s) from context paragraphs and create APIs .

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 (Jan 13 - Jan 26): Project planning, team building, proposal drafting
- Week 4-10: New chatbot model implementation, TutorBot feature implementing
- Week 11 (March 23): Spring Break
- Week 12-15: Test the results of the new model, continue implementing new features
- Week 16-17 (April 25 - May 3): Final Presentation

Project Description

What we have implemented:

- Retriever: get the top-n relevant documents to feed it into the reader
    - Word2Vec + relevance metric
    - BM25 algorithm
- Reader: extracting answer(s) from an input question based on the context
    - Transformers model using pre-trained model from Hugging Face
    - Currently using "deepset/bert-large-uncased-whole-word-masking-squad2"
    - Will most likely switch to a DistilBERT model to make it lighter and quicker

- Ability to save and load necessary variables to reduce pre-processing time (30 seconds => ~5 seconds)
- Image Scalper: retrieve the top-n links from Google Image search query
- An API: integration between TutorJS and Chatbot => TutorBot.

Problems:

The problems are in the order of importance, and some problems are from last semester.

1. **Explore and Implement Vector Search Databases:**

   Introduction to vector databases:

   https://www.pinecone.io/learn/vector-database/

   Multiple options explored here:

   https://towardsdatascience.com/milvus-pinecone-vespa-weaviate-vald-gsi-what-unites-these-buzz-words-and-what-makes-each-9c65a3bd0696

   Weaviate (one possible option)

   Read more about it here: https://www.semi.technology/developers/weaviate/current/

   What it is: A vector search engine. “Weaviate is created around the concept of storing all data objects based on the vector representations (i.e., embeddings) of these data objects” (https://github.com/semi-technologies/DEMO-text2vec-openai).

2. **Test the result(s) from implemented models**

   Model to test: BM25 + Transformer and Word2Vec + Transformer. Right now, we only test them visually with one question ("What is a first-difference filter?"). We may need to:

   1. Test with more questions
   2. Test with a defined metric such as the SQUAD 2.0 dataset (https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/)
      a. What it is: https://rajpurkar.github.io/SQuAD-explorer/

3. **Test different pre-trained models**

   Find the model for best accuracy/speed trade-off.

   a. Currently using "deepset/bert-large-uncased-whole-word-masking-squad2"
   b. Find a DistillBERT model from https://huggingface.co/models?pipeline_tag=question-answering&sort=downloads
   c. Find a different model from the recommendation section in Weaviate: https://www.semi.technology/developers/weaviate/current/getting-started/installation.html

4. **Add CUDA (GPU) support for all Transformers model (Quick)**
   Extracting an answer from thousands of paragraphs usually take around 15 seconds. However, with the help of a GPU, the processing time becomes roughly 0.5 seconds. In the transformers.py file, change the generate_transformer_response() to be similar to the generate_transformer_response_bm25() method.

5. **Clean up the existing code**

   Currently, there are two different models that are used to return an answer given context paragraphs and an input question. We may need to initialize both models at the same time in the future for a rating system to find out which model returns a better answer.

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.

8. **Improved performance (errors with updating the state)**
   For some reason, there is an issue with the image links not updating properly, so they do not load at all. This is due to React handling state updates asynchronously, so to restore that functionality, we will need some work on that front. The UI/TutorBot side members should also add any additional functionality for the chatbot that may be added over this semester.

9. **Add more returned components for students (JSX graphs, etc)**
   The TutorBot is capable of returning specific React components in its response. We can use this functionality to have the TutorBot display dynamic graphs and other useful features. The main challenge comes from figuring out what sort of graphs we would want to implement.

10. **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.

Foreseeable Challenges

The greatest foreseeable challenge in our timeline lies in implementing the Weaviate model. It is completely different from all models we have implemented so far, so we expected serious delays.

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