CS410 Progress Report

Group name: AlphaFold3
Group captain: Yang Duan (yangd4)
Group members: Yang Duan (yangd4), Chongyu Wang (chongyu4), Haoyu Su (haoyus2), Yulei Li (yuleili2), Jiaye Wang (jiayew3)

Chosen topic: Intelligent Browsing

Progress made:

- **Dataset and Pre-processing:**

  We have downloaded all the papers mentioned in the lecture as pdf files and converted the pdfs to html, then from html to txt. In the meantime, we have kept both the pdf files and the download links. To process the paper documents, we need to reformat them from pdf files to txt files, which are the format we can easily read by python. In fact, the difficulty is that many papers have subfields (columns), which are hard to process. The Optical Character Recognition (OCR) method is hard to implement and does not have a good performance for our dataset. We decided to utilize the tree hierarchy of html documents to distinguish different parts of paper documents. We changed the format of paper documents from pdf to html, then from html to txt. In this way, we stored our papers in txt format at the end.

- **Chrome Extension:**

  We've implemented the basic framework for the frontend of the google chrome extension using the react app. Currently, with the framework, the chrome extension is able to display basic information such as its title and a search button. It also allows users to enter their input as texts.

- **Ranking Algorithm:**

  We have implemented the algorithm which processes the paper documents from pdf to txt format, and we have implemented PLSA to extract the keywords in each paper document.

  We designed our rank algorithm based on PLSA in MP3. We implemented our modified PLSA which has 15 documents as input and ten topics. It traversed all text documents in the folder and iterated to likelihood convergence with maximum iteration of 100. We derive P(w):

$$P(w) = \sum_{j}^{k} \pi_{d,j}\, P(w|\vartheta_j)$$

It is the probability of words appearing in each document. Then, we extract the word with the highest probability as the keyword in that document.

- **Server:**

  In the previous weeks, we learned how to use Node.js to integrate the functionality of System IO and File System. With the powerful features of Node.js, we implemented the basic framework of the backend server. It now can receive the request from browsers with well-structured headers and bodies. The child process module of Node.js was also utilized to run the Python prediction program with trained models locally and capture the output of Python programs. Currently, the server program uses JSON to transmit data among the frontend programs and the Python prediction program.

Remaining tasks:

- **Dataset and Pre-processing:**

  The tasks for collecting datasets and preprocessing information are currently completed. If the size of our current dataset cannot fulfill our needs, we will include other resources and files appropriately according to the situation at that time.

- **Chrome Extension:**

  The frontend needs to be able to recognize the content highlighted by the user and automatically set it as the default input for the text box in our extension.

  In the integration process of combining the frontend, backend, ranking models and the database programs, we still need to implement methods to send and request the data to and from our backend server, and these methods should help the backend server keep track of the text message that the user inputs to our python prediction program. The extension should also have the function of receiving the output data from the backend server and displaying it correctly for users to easily navigate.

- **Ranking Algorithm:**

  We will implement the query matching algorithm in the next several weeks, to get a similarity of our input query words with the paper documents. Then we will implement the algorithm to get the context for a keyword in the paper documents. After that, we can show the context in the paper with the query word on our extension.

- **Server:**

  There are several things we need to do in the following weeks. We think two of these things are the most important and time-consuming regarding the server program for us.

First, we need to find a way to handle the connection between Chrome Extension and the server. Currently, the server can only handle the connection from browsers. It seems like Chrome Extension has its different strategy and APIs compared to browsers regarding sending requests, visiting links and sending messages.

Second, we need to implement a sub-module of the server program as the web crawler to grab information from Wikipedia and Google Scholar so that the server can send these pieces of information with the result of Python ranking or prediction programs together to Chrome Extension to display the list of papers, links and online information that the users need.

Challenges/issues being faced:

- **Chrome Extension:**

  Since everyone has a different background of knowledge in computer science, most of the group members have very limited knowledge about building the frontend programs. Learning how to build the frontend costs us plenty of time. We need to figure out the structure of building the frontend, connecting Chrome Extension to the server, and the functions of every piece of files.

  Especially in the process of learning different frontend features we might need, different examples adopt different framework models, and integrating them is quite challenging for us as beginners.

- **Ranking Algorithm:**

  The challenges include the precision of PLSA, the running speed of our algorithm, and how to deal with the case that the query word is not in paper documents but has a synonym in paper documents. To solve the first two problems, we need to adjust the parameters of our model and replace all computation with matrix manipulations to accelerate the algorithm. For the last problem, our group will discuss it in the next several weeks. One sight is to use some embedding methods to find the similarities with query words and documents.

- **Server:**

  In order to achieve the goals or TODOs for the server program we planned above, we need to learn new online lessons first. We lack the experience and knowledge related to web development and web crawler which are essential in these planned tasks.

  First, we need to learn web development to better understand the web connection, web request and web response. We also need to learn how Chrome Extension or even Chrome Browser works to connect with the server in the aspect of sending and receiving messages, which seems complicated and difficult for us.

Second, we need to learn how to use the web crawler to grab online information automatically. It is totally different from how we manually visit online websites. In addition to the traditional web crawler, we may also need to learn how to use APIs provided by these online websites in our programs.