Christopher Rech - 124000677

Andrew Allen - 124008219

Project Proposal: GroupMe Search Bot

The problem being solved is the lack of an existing search functionality in the GroupMe messaging app. Existing popular messaging platforms such as Facebook Messenger, Slack, and Discord all have an option to search through past messages, but GroupMe has no such functionality. This project will allow GroupMe users to search through their prior messages, as they would be able to in other apps.

The data on which the app is being tested is a collection of GroupMe chats. Although the total number of messages in these chats does not quite reach the targeted 100k mark, there is more than enough data to demonstrate a basic search functionality, even for less common queries. The data on which the app could potentially be used in the future is any existing GroupMe chat, if the search tool is invoked on by one of the members in the chat.

The project will involve creating a bot for GroupMe to automatically handle search queries. The components of the project include message crawling, search infrastructure, and query handling. The bot will continually crawl its groups for messages to stay up to date in its retrievals and to be notified of queries. Messages of a special format in the group will be considered queries. Whenever a query is processed, the bot will perform a search on its database and respond to the query.

The data crawling will be done through the built-in GroupMe API, which can be queried through the requests library. Messages will be parsed from the Json results and converted into a format such that they can be handled by Solr. The functionality will include searching by both user and keyword. All messages will be stored in the database such that they are partitioned by group, since a search always occurs on just one group. In order to handle user searches, messages will be indexed by user, sorted on timestamp. Whenever a user search is performed on a group, messages from that user will be returned, with most recent messages being shown first. In order to handle keyword searches, the program will use the functionality built into Solr for searching on keywords. Messages will be returned ordered by relevance, with ordering by recency if possible.

Three metrics will be used to evaluate success: relevance of results, easy of use, and quickness of data availability. Relevance of results will be measured by our evaluation of sampling edge cases, capabilities in domain specific searches (for example, searching a group message for dates/times of events or for links to useful resources sent within the group), and our bots to spell-check and/or synonym match (to handle those "on the tip of my tongue" moments). Ease of use relates to bot and website interactions with a focus on the ability of a user to add the bot seamlessly to group messages and/or log in via our website. The quickness metric is specific to the problem we are facing. We won't have access to the data until a user adds the bot to a chat or logs in. Thus, we want to provide a user with search capabilities as quickly as possible, meaning we must crawl (imitate scrolling up) and index the data within a reasonable amount of time.

We expect to submit a bot which is able to crawl chats it is added to and/or any chats for a given user if they submit their credentials on our website. We will allow users to search said chats via a public search, everyone in the group chat can see query and results, or by sending the bot a private message specifying what chat to search.