Information Retrieval Report

João Tomazoni

December 2, 2021

Project number 26: Chat Search Engine

Task

Chat Search Engine: a user wants to search on chat forums and websites like Gitter.im and Reddit for a particular topic. You have to build a system that gathers a large collection of chat transcripts or dialogues on a few topics and enable to search them. The system should be able to perform a field search on chats. The system has to provide the best interface for searching, browsing, and presenting the data to the user. The system should also have the following additional feature:

- User Relevance Feedback: after presenting the search results to a user, the user may provide a positive or negative feedback on the results (i.e. mark relevant and irrelevant documents). Based on this feedback the search results have to be updated and presented again.

To accomplish that, I have gone through these steps:

1. Create two spiders (or crawlers) to crawl two chat focused websites.
2. Create a collection and handle it with Solr.
3. Create a user interface to submit search queries and present the results.
4. Set up the query and implement the basic relevance feedback feature
5. Uploaded all the files to the github page together with a readme file
6. User evaluation performed by three users (out of seven) that were present in class on 01/12/2021.

The websites that I have used to get the material to create a collection and make the project work properly are the following:

1. Reddit, a network of communities where people can dive into their interests, hobbies and passions. (https://reddit.com)
2. Quora a social question-and-answer website (https://quora.com)

Crawling

I've chosen to crawl for specific topics, that can be found inside the csv file called *keyword.csv*, rather than starting at a random page and following the links from there, as in my opinion, it makes more sense when browsing a chat engine, you would want to know what people have to say about a specific topic in which you're interested on.

Crawling Reddit was as simples as it gets, the robot.txt is friendly towards crawlers and the css structure is easy to identify and use when setting up the crawler parameters.

Crawling Quora however, was trickier but not by much, the website requires the unique string value you get from longing in to the website so it knows the requests we're making are coming from a verified person, such string is cookie'd and easy to retrieve, but then, sometimes, the crawler might stop working and requires a new string to keep going.

The Professor also suggested a website called Gitter.im, I looked into it but the site has very few topics and they aren't very consistent, so I chose to pick quora instead.

I also tried to scrape Twitter, but that was no good, apparently they have their own business that you can pay for the data of what is being said about the topic you're paying for, and therefore its on their best interest to go against crawlers. I also ended up crawling google for the top 10 results on the same topics I crawled for on reddit and quora, when I was desperately attempting implementations of the relevance feedback. You can find the core called greddit, which is the core holding the topics to be displayed should a user clicks on the "more like this" button while using the searching on reddit feature. More on this in the following subsections, but I wanted to keep this implementation regardless on one of the searches just so my work wasn't wasted, for quora the implementation is a bit more towards the right direction.

Indexing

With the data scraped from the before mentioned websites, the next step was to build and index the collection with SOLR, I chose to have two independent cores, one specific for reddit and the other for quora. This decision was due to reddit being very random and off topic in many ways, be it the topics or the answers, whereas in quora you can expect more serious answers and takes towards the questions. SOLR automatically detects the proper columns and rows from the csv files and provide a nice structured json with every row having a unique ID to be easily identifiable. With the now indexed and structured data on SOLR the next step is to build the website to host the queries and provide results.

Web UI

The Dodge Search Engine has a simple CSS box model with few HTML elements, most of its content is displayed through javascript after sending the query to SOLR and retrieving the response to be displayed to the user in order of relevance as ranked by SOLR. When accessing the Doge Search Engine, the user can see a search bar in the middle of the page, that holds the information regarding whether the searches are going to go through reddit or quora, and he can choose to modify the website to be searched on by selecting it on the top left corner. As seen in Fig.1.

After querying something in the search bar, say, squid games, the query words are used with the SOLR API to retrieve the most relevant page in the appropriate collection, in this case, the words squid and games are going to fill the

```
${query}
```

parameters in the link similar to:

```
http://localhost:8983/solr/quora/select?indent=true&q.op=OR&q=Comment_5%3A
    ${query}%20OR%0AComment_4%3A${query}%20OR%0AComment_3%3A
    ${query}%20OR%0AComment_2%3A${query}%20OR%0AComment_1%3A
    ${query}%20OR%0APost_Text%3A${query}&rows=200‘;
```

Which will then be used by SOLR to perform a query on all the posts, comments, answers and keywords in the collection. This response is then displayed to the user as seen in Fig.2.

Each result is clickable, and clicking on one will expand it and display the URL from which the information is coming from, the Post text, the Post comment if the Original Poster has done one, followed by up to 5 most voted comment or answer in the day that the information was retrieved. As well as the two buttons attempting to implement the relevance feedback featured, discussed in the next subsection.

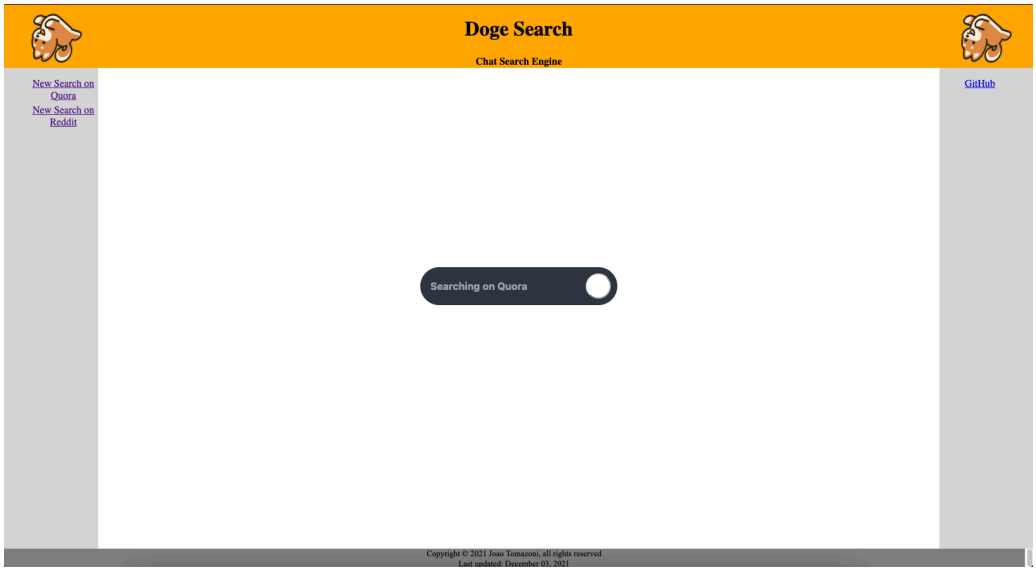

Figure 1: How the index page looks like for the user when visiting the website

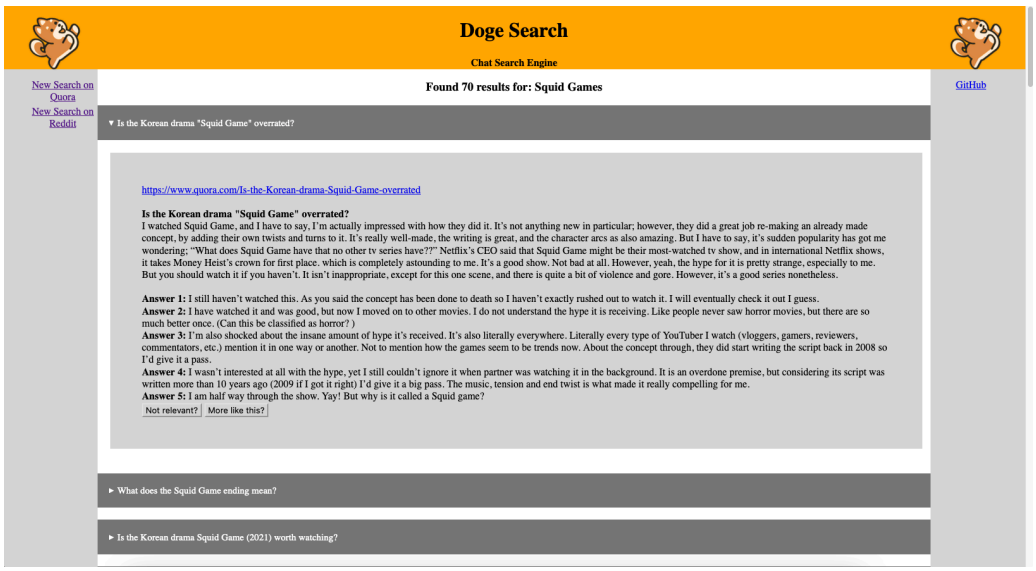

Figure 2: Results from the query "Squid Games" when searching on Quora

Additional Feature: User Relevance Feedback

There was an attempt at implementing it. Ideally you would want a query such as "cute animals", that when given a positive feedback, would identify the most common terms in that document, boost said terms in weight, and retrieves new results based on the new weights. Such that if dog was the most present term, the user probably likes seeing cute dogs a lot, and therefore posts about cute animals that include dogs should be ranked higher. There is a way to do so in SOLR, one has to use the Luke Request Handler, discussed in (https://solr.apache.org/guide/8_8/luke-request-handler.html), that allows us to do just that, but unfortunately, due to lack of time and the handler not promptly working on few attempts, I couldn't proceed with it to properly achieve a more adequate relevance feedback.

Instead I had to rely on what I already have, and used the structure of my documents as a way to boost the query, I have at my disposal which comments or answers were the most voted by the community, therefore I chose to use the EDIMAX feature to boost the terms that are presente in the first comments more than the other less voted comments, in a way to give priority to what the community decided to be more relevant to the topic to be the most relevant to the user as well, as opposed to the rank already provided by SOLR.

For the negative feedback, I also went with the simplistc and not optimal solution. It's a map structure in javascript that will store the user's query, and if he chooses a document as irrelevant, that document will have its unique ID removed from the collection, then a new query is performed ignoring that unique id that was removed. The new query value is then mapped as the key to the initial query, then whenever the user inputs the same query, the system will find the query in the map structure and use the new query instead, with all the irrelevant documents already excluded. Fig.3. Shows the console log for the initial query and the new query with the unique id being subtracted from the list of relevant documents, after the user pressed "not relevant" to one of the documents

I only implemented the described positive relevance feedback to the quora core, to the reddit

core I chose to keep the old implementation that I did in order to have something happening when a user pressed the "more like this" button. The system will simply use the query words to search and google and provide the top 10 results from them instead. I kept that for two reasons, firstly to not put the work to waste, and secondly to have the users evaluate it during the user evaluation task and see their opinions.

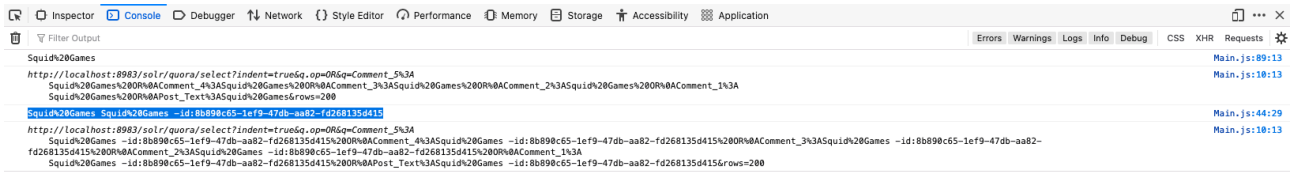

Figure 3: Highlighted is the query and the query with an excluded id, both are mapped together

User Evaluation

In class I had 3 of my peers to do a quick user evaluation of my system. I provided 3 tasks:

1. Query any topic of your liking to both quora and reddit.
2. Press the "not relevant" button on one of the documents.
3. Press the "more like this" button on one of the documents.

Followed by the following questions, that could be answered from 1 to 10:

Question:	User 1:	User 2:	User 3:
Were the results relevant?	10	10	9
How useful is the not relevant button?	10	9	10
The more like this feature returned more relevant documents?	5	1	1

Clearly the positive relevance feedback is not ideal and requires a re-work, I intend to continue improving this search engine as it was quite fun working on. Also, apparently the indexing and ranking provided by SOLR is good, as the users have given a high score to the first question. Their answer to the second question I wasn't expecting to be this high, apparently not having irrelevant documents to be displayed in the future is good enough for them.
Of course the data set of three users is too small to draw any proper conclusion, but can give an idea on how the systems perform, as for the third questions, the answers were very conclusive, specially because the 5 was due to the user receiving more like this from google.