Course Project Report

CSCI 426 - Spring 2022 - Information Retrieval

New York Institute of Technology - Department of Computer Science

---

Title : Reddit Search Engine

Team Leader : Aloysius Arno Wiputra

Members :

1. Aloysius Arno Wiputra 1244139
2. Andrew Molina 1255187
3. Ashley Valdez 1257891
4. Devaaum Shah 1310191

ABSTRACT

The site known as reddit is widely known for its inside communities that connect users with similar interest to posts on its site where they can share information and see posts by other users based on their popularity within that community. We wanted to create a search engine that gives users results that are better tailored to their search, without the need to add a parameter into google as we have noticed many users do. After our base implementation and running a limited amount of tests we came to the conclusion that our search engine does possibly give more accurate results based on sorting but it also has much to expand and improve on for the future.

1.0 Project Overview

1.1 Background

The site known as Reddit is widely used for users to search and discuss mutual interests through posts, pictures and or videos. Within these communities the most popular posts which have received the most upvotes will appear at the top, while least popular posts are moved to the bottom of the list. The issue we are addressing with our project is how users feedback on reddit is the posts they search for do not show on the results page. With our project we will focus on optimizing the results to generate useful information from reddit.

1.2 Motivations

Our motivation for this project comes from the issues users face when trying to navigate reddit. We have attempted to solve this by optimizing the results where we obtain useful information from the site using what we have learned and gone over in class. Methods such as lemmatization, stemming, tokenization and stopword removal are a few of the techniques used during the creation process of the search engine.

1.3 Related work

In order to understand the issues and react to what does not work in the reddit search engine, we looked at similar search engines such as Google, the leading number one search engine, along with sites such as Yahoo and Bing, which are on more of a similar level of popularity. Based on this research we had come to the conclusion that most users will leave the reddit site and use google search engine with the parameter "site:reddit.com" in order to get better search results than they would just searching on reddit.

2.0 Methodology

2.1 Description of the dataset

The original dataset consisted of 1889 json files totalling 91.8MB. Each file has 18 columns consisting of "title", "name", "url", "selftext", "score", "upvote_ratio", "permalink", "id", "author", "link_flair_text", "num_comments", "over_18", "spoiler", "pinned", "locked", "distinguished", "created_utc" and "comments". Each file corresponds to a particular submission and the columns represent all the metadata that reddit stores and makes available publicly through its API. This dataset is then converted into a Pandas Dataframe and converted into one csv file for convenient access.

2.2 Method of collection

An open source tool called the Bulk Downloader For Reddit (BDFR) created by Ali Parlakçı was used in the collection of data. The BDFR fetches directly from the Reddit API and has built in redundancies which makes it a more attractive tool than iterating our own in the short time that we have. Windows Powershell and Python 3 were used to install the tool using the command: $python3 \ -m\ pip\ install\ bdfr\ –upgrade$ directly into our machines.

```
PS C:\Users\aloysius_w> python3 -m pip install bdfr --upgrade
Collecting bdfr
  Downloading bdfr-2.5.2-py3-none-any.whl (54 kB)
     ---------------------------------------- 55.0/55.0 KB 2.8 MB/s eta 0:00:00
Collecting dict2xml>=1.7.0
  Downloading dict2xml-1.7.1.tar.gz (6.6 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting pyyaml>=5.4.1
  Downloading PyYAML-6.0-cp310-cp310-win_amd64.whl (151 kB)
     ---------------------------------------- 151.7/151.7 KB 4.6 MB/s eta 0:00:00
Collecting click>=7.1.2
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
     ---------------------------------------- 96.6/96.6 KB 5.8 MB/s eta 0:00:00
```

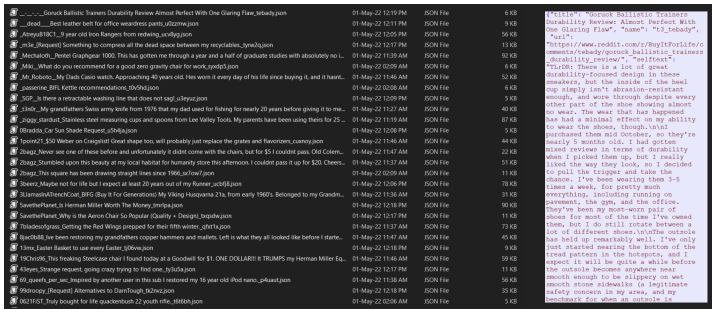
BDFR supports three methods of data downloading: “download”, “clone”, and “archive”. “Download” acquires only the post title and text into a .txt file without any additional metadata while “archive” retrieves it as is from the API. “Clone” does both actions at once, downloading both a .txt file of the basic post information and also the raw json file. All were downloaded into a local hard disk and then moved to Google’s cloud solution, Google Drive, for storage and easier interfacing with Google Colab.

2.3 Issues Encountered

Several issues were encountered in using this tool. We found that in spite of the default download limit being described as unlimited, instead of fetching a full collection of all the posts in the subreddit, the tool only grabbed a limited amount of submissions. We worked around this by running the tool several times and with different configurations, such as a different sort order in which it grabbed the files.

2.4 Samples

Below is a screenshot of a sample showing all the json files and a raw preview of its contents.



3.0 Approach

3.1 Initial setup

Several things had to be taken care of before beginning work, mostly in initializing the dependencies in which this project was built on. We imported python packages such as pandas for its dataframe tools and numpy for certain calculations. For our function to process the raw json files we used three utilities: glob, os, and json. We also used nltk for natural language processing as well as datetime to calculate decay factor and drive, a Colab utility to interface directly with Google's Drive. In each step, we saved the progress into a new processed csv file so as to make it easier for us to resume work in each block without having to run all the commands from the beginning. All we needed to do was run the setup and then any particular block.

3.2 Dataset conversion

We uploaded all 1889 files to Google Drive and mounted it using the aforementioned Colab library. We then used a function to fetch all the files in the particular folder with a .json extension and added it to a python data structure. It was then converted into a native pandas dataframe and saved into a .csv file so we do not have to rerun the processing step from the beginning.

3.3 Cleaning the data

Steps were taken to clean the data to avoid issues in the future. Null values were replaced with empty strings such as " " so as to avoid any issues in processing and certain columns, such as boolean columns that reddit uses to markdown posts were removed as they offered no relevance in the project. These include but are not limited to "over_18", "spoiler", or "pinned".

The download also marked the timestamp in a raw UNIX format and we used a pandas function to convert it into a readable general format. We left the timezone as the default UTC.

3.4 Language pre-processing

We applied several language pre-processing techniques that were learned in class. Before that, however, we combine the title of the post and the body of each post to make one new column entry to create a full representation of the text in that submission. We then normalized it by converting all letters into a lowercase and performed stemming. We then tokenized the text and separated each word into a python list. We then wrote a function to store a vocabulary of stopwords and iterated through each list to remove the stopwords

and other unused symbols. We also converted it into a set and used its natural property to remove duplicates.

3.5 Calculating decay factor

We included an additional column that factors in exponential decay to adjust each submissions' score. We figured that as some of these submissions go back to 2015, the contents might not be as relevant as ones fetched in recent days. So we used a common decay formula

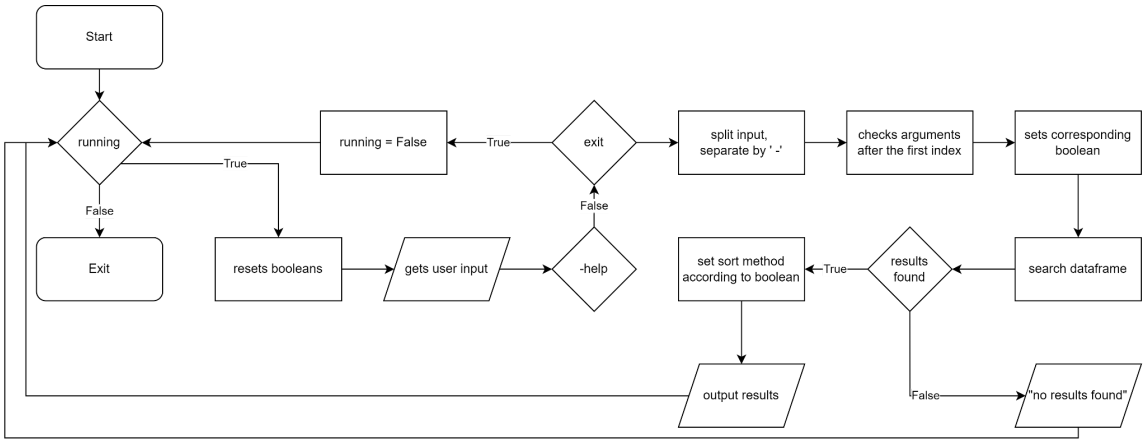
$$e^{-a\,(current\ time\ -\ rating\ time} \quad where\ e\ =\ rating,\ and\ a\ is\ a\ constant$$

We then used a python library to fetch today's date and time and perform adjustments everytime the colab project is run. This results in older posts getting a slightly lower point compared to newer entries based on how old they are.

3.6 Search engine implementation

Our implementation of a search engine is essentially a keyword search that uses a partial best match model. It searches through the keyword column for each row and finds if it finds at least a partially matching term in the query. We had planned to do query expansion but did not find ample time to implement it properly.

We did however find time to create a basic terminal interface that will parse your input and accept several arguments. A basic flowchart detailing how it works is included below:



Upon initializing the script, the program starts with a try catch block and a boolean condition that is by default true to ensure the program keeps running. The variables

including the boolean are reset every run to ensure that the results stay consistent. Upon submission of the query, the program checks if the user wants to request “-help” and then “-exit”, this is to ensure that if the user would like to search for the query “help”, they can do so. It then parses the input and separates the entries into a list, with the different argument separated by the dash prefix.

The program then runs checks for redundancy, verifying that there is an additional argument aside from the query and if there is a query in itself, then sets boolean values accordingly, for example if the user would like to sort by decayed rating, then they can input either $-d$ or $-decay$. A switch case statement would have been preferable to an if else, but python’s implementation does not seem to allow for multiple switches for one output at the moment.

If the checks are valid and there is an argument, then the query is then passed on the search function which goes through the dataframe. It then stores the results into a separate search result dataframe, which is then sorted according to the argument and boolean functions, but by default it sorts based on the points in descending order. As it is a terminal, we could not use panda’s dataframe visualization and thus used print statements to output the results neatly. However, due to some of these results being able to reach 30 posts, we also included a string argument to limit the number of output posts so as not to clutter the terminal unnecessarily. This is taken into account when doing the boolean checks.

4.0 Results

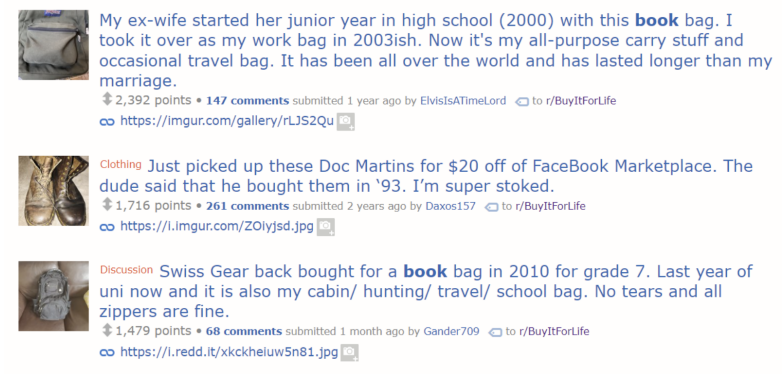
4.1 Experiments ran

We performed searches with several different queries and all of them output as expected. Some do not have a body in them and we adjusted the output accordingly so as not to give off an empty value.

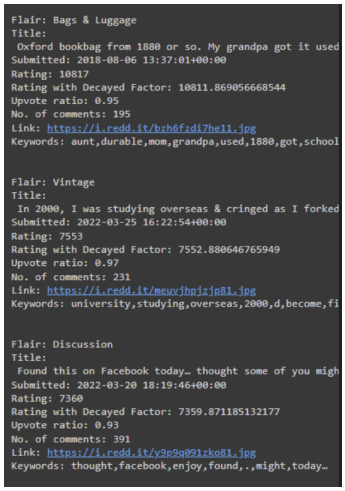
We tested each sorting argument and all outputted as expected, including no argument and post display limits.

4.2 Data comparison

We then tested Reddit's native search engine with the same parameters, the query is "book" and the sort order is "top"



For comparison, here are the results from our search engine.



4.3 Discussion

We found the results to be a bit of an anomaly. We found that reddit's results, in spite of using the exact same parameters, were displaying a limited amount of posts. For example, our top result had 10817 points assigned to it from 2018. Reddit's top post for the same keyword was one with 2392 points and from 2021. For additional checking, we queried the search term into google with the argument $site:reddit.com$ as well as following up on the url and found the post we returned with the correct points.

This could very well be an anomaly in the system when we did our checks or it could be an example of how unreliable reddit's native search engine is.

5.0 Conclusion

5.1 What worked

After the completion of our search engine project we came to the conclusion that based on testing sorting and the use of different methods such as lemmatization, stemming, tokenization and stopword removal our results possibly give better, more accurate results than reddit, our base search engine. What we have learned from these tests is how powerful language processing is and the effect it can have on search results, as well as more improvements we have not had time to approach but will most likely improve our search engine results in the future.

5.2 What did not work

Unfortunately due to time constraints we were not able to implement advanced search methods such as TF-IDF to show document relevance on relevant words over irrelevant and cosine similarity showing the resemblance between two sets of samples. Another step

we were not able to complete was the front end of our search engine, using a library to implement web applications and flask for backend. Had our schedule been extended we had planned to learn and use flask to develop our project even further using python.

5.3 Future work

After the finalization of our search engine we reviewed what worked and what did not work in the process, in doing so we came up with future work that would further improve our search engine to bring in more accurate results. From that list of future features we have included expanding our search parameters, such as filtering dates, search comments, etc. We also would have liked to add additional search methods such as similarity cosine and TF-IDF, as well as using live search data instead of the archived data we were working with.

5.4 Member contributions

Aloysius Arno Wiputra:

- Designed the layout and created the presentations for both proposal and final report, wrote the methodology, approach, experiments, and conclusion slides
- Outlined the whole project report and wrote the methodology and technical aspects of the report
- Researched the tools necessary and how to use them for this project
- Designed and coded and debugged the whole project from scratch with references from Google and the class hands-on as well as ran the downloading tool on personal machine as well as designed the flowchart and program flow
- Organized member presentations and divided up roles in who speaks what

Andrew Molina

- Aided in project organization
- Created and presented the problem, method, and definition slides
- Participated in discussions

Ashley Valdez - Research, collection

- Researched the problem and overview for the project
- Created and presented the introduction slides for the final report, and edited the proposal presentation

- Co-wrote the project report, focusing on the introduction, overview, and conclusion of the slides as well as wrote the abstract

Devaaum Shah

- Participated in member discussions
- Performed some literary research

REFERENCES

Aliparlakci. "Releases · Aliparlakci/Bulk-Downloader-for-Reddit." *GitHub*, https://github.com/aliparlakci/bulk-downloader-for-reddit/releases.

"BDFR." *PyPI*, https://pypi.org/project/bdfr/.

Khalid, Irfan Alghani. "Create a Simple Search Engine Using Python." *Medium*, Towards Data Science, 21 Sept. 2020, https://towardsdatascience.com/create-a-simple-search-engine-using-python-412587619ff

.LZP, Jason. "Simple Way to Extract Reddit Comments in Python." *Medium*, Geek Culture, 17 Apr. 2022, https://medium.com/geekculture/simple-way-to-extract-reddit-comments-in-python-c8cb2afe2fce.

"NumPy Documentation." *NumPy Documentation*, https://numpy.org/doc/.

"Pandas Documentation¶." *Pandas Documentation - Pandas 1.4.2 Documentation*, https://pandas.pydata.org/docs/.

Rais, Zayed. "Build Your Semantic Document Search Engine with TF-IDF and Google-Use." *Medium*, Analytics Vidhya, 23 Feb. 2021, https://medium.com/analytics-vidhya/build-your-semantic-document-search-engine-with-tf-idf-and-google-use-c836bf5f27fb.

"Reddit Inc. Homepage." *Homepage*, https://www.redditinc.com/.

"Reddit." *Reddit*, https://www.reddit.com/.