

# FINAL PROJECT REPORT

## NOT SO ROTTEN TOMATOES- SCRAPING AND RANKING ROTTENTOMATOES

UIUC: FALL 21 CS 410 - TEXT INFORMATION SYSTEMS

**Topic:** Scraping and Ranking RottenTomatoes

**Theme:** Intelligent Browsing

**Team Members:** Jeremy Wisuthseriwong (jrw7), Supriya Puri (puri6), Munesh Bandaru (muneshb2)

**Team Captain:** Munesh Bandaru (muneshb2)

**Web application url:** <https://not-so-rotten-tomatoes.herokuapp.com/>

**Source url:** <https://www.rottentomatoes.com/top/bestofrt/?year=2021>

### BACKGROUND

Vertical search engines have become increasingly popular, now-a-days, as they sift through limited databases for information. A general web search cannot accommodate all of the users' searches when it comes to specific topics without implicit assumptions. In particular, using a vertical search, a user can extensively use query based searches to get the desired results with high user ratings and reviews. One major example for searching a specific topic is "**Rotten Tomatoes**" - a review aggregation website for movies and television series. Its content is specialised for users browsing information on top rated movies and television entertainment - genre, cast, network or the critic and user ratings. Results from services like Rotten Tomatoes allow a user to rank results by User Reviews, Critic Reviews, Genres, Audience Score. Being able to track user experience for various movies and tv series could lead to a larger audience and greater profits .

### INTRODUCTION

A user experience is all about how browsing a web page becomes valuable and meaningful for a user. Finding the right things on the internet is not easy - there isn't a way to effectively search and rank a list of the top 100 movies for 2021 based on a given user query. Returning things that are of relevance to the user along with filtering the content based on the popularity is a challenge. Any general search engine would parse all the pages related to the query and search in a breadth-first manner to collect results. A query-specific

search more efficiently searches a small subset of content by focusing on a particular requirement.

Through this project, we are trying to improve the user experience of browsing the content based on the user's interests. On the Rotten Tomatoes web page, users can find various pre-defined ratings and rankings for movies and TV series like Best Movies of 2021, Popular Shows on Netflix but there isn't a way to effectively search and rank for a list of movies within that particular ranking matching the user query/interest. Here we are making our system to provide an intelligent way of browsing the content within the top 100 movies filtered and sorted based on the user query.

### Example:

Lets say a user is browsing the "Top 100 movies of 2021" on Rotten Tomatoes.

**TOP 100 MOVIES OF 2021**

**BEST OF RT**  
BEST OF ROTTEN TOMATOES  
Movies with 40 or more critic reviews vie for their place in history at Rotten Tomatoes. Eligible movies are ranked based on their Adjusted Scores.

Year:  Sorted by Adjusted Score

Rank	Rating	Title	No. of Reviews
1.	93%	Nomadland (2021)	418
2.	96%	Judas and the Black Messiah (2021)	340
3.	98%	The Father (2021)	279
4.	94%	In the Heights (2021)	350
5.	99%	Summer of Soul (...Or, When the Revolution Could Not Be Televised) (2021)	202

**TODAY'S TOP RATED MOVIES**

CERTIFIED FRESH IN THEATERS

- 97% Pig
- 96% CODA
- 90% The Suicide Squad
- 92% Shang-Chi and th...
- 100% Sabaya
- 86% I'm Your Man

But the user is more interested in looking for thriller movies available on Hulu from this ranking list of 100 movies. When looking further, he could not find a way to filter and sort this list and needed to go through each movie description to find out whether a movie meets his interests or not.

### **Sample query: "thriller movies on Hulu"**

Our application helps the user to actually find a ranked list of Top 10 movies according to the query provided.

## **FUNCTIONAL OVERVIEW**

Our code scrapes the urls of movie reviews listed on the Rotten Tomatoes' Top 100 movies of 2021 webpage. Additionally, it scrapes content from each movie page, such as movie title, genre, rating, cast, synopsis, and critic reviews. We have a web application that is built on

top of the BM25 ranking algorithm that allows the user to input a search string and display the 10 best matching results on the page. Our application can be used for helping a user to find a ranked list of Top 10 movies from 2021 according to the query provided. This will improve the user experience of browsing the content based on the user's interests.

## TECHNOLOGIES

We chose **Python** as our primary programming language to design our algorithm and use its relevant libraries to assist with scraping the web pages, text processing and modelling tasks.

The Python packages that we primarily used include:

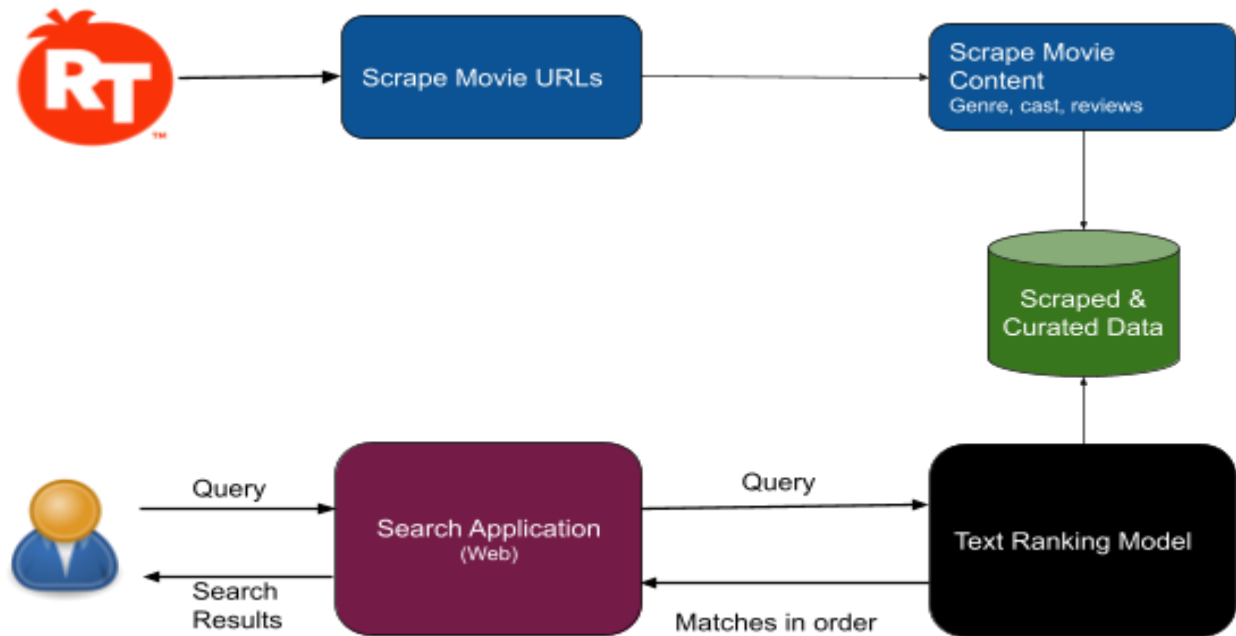
- Metapy
- BeautifulSoup
- Pytoml
- Scipy
- Selenium

For the web application, we worked with **Python Flask**, **HTML** and **CSS**.

For deployment of the web application we used **Heroku** as the cloud platform and **Gunicorn** "Green Unicorn" as a Python Web Server Gateway Interface HTTP server.

The instructions to install each package has been shared in [install.sh](#) file in the Github repo. Users can directly run [install.sh](#) and all the required packages and libraries will be installed for the next steps.

## SOFTWARE IMPLEMENTATION



The diagram above gives an overview of the steps involved in the implementation of the application.

### DATA SCRAPING:

This projects required 2 levels of scraping

- Scraping the movie URLs for the Top 100 movies of 2021
- Scraping the movie content like genre, cast, media from the 100 movie URLs scraped

We used the python package '*BeautifulSoup*' for scraping the top 100 movie URLs and the content for each movie and saved the data files for the ranking model.

[movie\\_dir\\_scraper.py](#) -

This python program returns the url for Top 100 movies of 2021 from RottenTomatoes <https://www.rottentomatoes.com/top/bestofrt/?year=2021>

[page\\_scraper.py](#) -

Using page\_scraper.py, we scraped the below content from each movie page.

- Movie title
- Synopsis
- Genre

- Cast
- Where to watch → Netflix, Amazon
- Critic Reviews
- Tomatometer rating

The data was last scraped on **16th November, 2021**

Data files are placed under below location:

<https://github.com/muneshb/CourseProject/tree/main/source/data>

**\*Note:** If webpage content is rescraped, then results are subject to change as new movies are added or updated on the Rotten Tomatoes site.

## TEXT RANKING MODELING:

Ranking of query is one of the fundamental problems in information retrieval (IR), the scientific/engineering discipline behind search engines. Given a query  $q$  and a collection  $D$  of documents that match the query, the problem is to rank, that is, sort, the documents in  $D$  according to some criterion so that the "best" results appear early in the result list displayed to the user.

Through the [search\\_rank.py](#) python program, users can apply the ranking algorithm in the load\_ranker function to generate accurate and relevant results according to the input query.

Input files:

Config.toml

Data files:

movie\_urls.txt, titles.txt, synopsis.txt, and ratings.txt

Output files:

avg\_p.txt, ndcg.txt

### Description:

The program contains a load\_ranker function that reads in a config.toml file and returns the ranked results using the BM25 algorithm with parameters  $k1=1.2$  and  $b=0$ . It also prints results of mean average precision and NCDG@k to the console and to the output files.

The **'process\_query()'** function defined here is called from the [library.py](#) program, which takes in the input query from the web app and then runs the ranker to display the results.

We evaluated below 3 ranking algorithms:

BM25, Jelinek-Mercer, and Dirichlet Prior.

From tuning parameters during testing, we identified optimal parameters for each ranking algorithm with the best MAP and NDCG@10 for our application:

Algorithm	Mean Avg. Precision	NDCG@10
<b>BM25 (k1=1.2, b=0)</b>	<b>0.6741369048</b>	<b>0.8299627915</b>
Jelinek-Mercer (Alpha = 20)	0.6113575708	0.7800326674
Dirichlet Prior (mu = 0.1)	0.6028754279	0.7756776975

Based on the results, we determined BM25 to be the best ranking algorithm to use in the implementation of our application.

## DOMAIN RESEARCH

To improve the precision of the results, we would generally use stop words to make sure these common words don't impact the query matches.

We remove the low-level information ([stopwords.txt](#)) from our text in order to give more focus to the important information

These common words might be different for different use cases.

In movie search, people tend to use some verbose words like movies, films while searching for the movies they are interested in.

For example:

**Query:** Comedy movies on Netflix

**Document 1:** comedy movies netflix movies action Comedy

**Document 2:** comedy movies netflix movies action movies romantic movies

In reality, the user is not looking for more matches for the 'movies' word but the algorithm considers **Document 2** as a better match as it has more matching words.

Here, we know that **Document 1** is a better match as it has more matches for comedy

To remove the low-level information and focus on the important words, we added more stop words to the **stopwords.txt** file based on the movie domain

More stop words: movies, films, cinema

## EVALUATION

We have manually verified the documents for the query matches and the corresponding query match counts and created the query judgements to verify the results of the model.

Query Judgements File:

<https://github.com/muneshb/CourseProject/blob/muneshb-queries/source/data/queryrelevance.txt>

We have used Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP) evaluation methodologies to validate the model results using the judgements file created above.

We have also computed average precision to validate the model performance for each query.

The Average precision values are saved in the file below:

[https://github.com/muneshb/CourseProject/blob/muneshb-queries/source/data/avg\\_precision.txt](https://github.com/muneshb/CourseProject/blob/muneshb-queries/source/data/avg_precision.txt)


The nDCG value is saved in the file below:

<https://github.com/muneshb/CourseProject/blob/main/source/data/ndcg.txt>


## USAGE OF THE SOFTWARE

Our search application allows the user to search a movie by :


### Search by Movie name / Genre / Cast / Review Content




*Top 100 Movies Of 2021*

[Wildland](#)  93%

Wildland vudu itunes Following a car accident, which kills her mother, 17-year-old Ida moves in with her estranged aunt and her aunt's grown sons. The home is filled with physical tenderness and love, but outside of the home, the family leads a violent and criminal life. When an unforeseen murder pressures the family and their loyalty to each other, tension builds as love and violence become impossible to separate. Ida is faced with the same question her mother faced before her: What are you willing to sacrifice for your family?

[Final Account](#)  92%

Final Account vudu netflix amazon-prime-video-us itunes FINAL ACCOUNT is an urgent portrait of the last living generation of everyday people to participate in Adolf Hitler's Third Reich. Over a decade in the making, the film raises vital, timely questions about authority, conformity, complicity and perpetration, national identity, and responsibility, as men and women ranging from former SS members to civilians in never-before-seen interviews reckon with -- in very different ways -- their memories, perceptions and personal appraisals of their own roles in the greatest human crimes in history.

[Rose Plays Julie](#)  95%

Rose Plays Julie vudu amazon-prime-video-us itunes Rose (Ann Skelly) is at university studying veterinary science. An only child, she has enjoyed a loving relationship with her adoptive parents. However, for as long as Rose can remember she has wanted to know who her biological parents are and the facts of her true identity. After years trying to trace her birth mother, Rose now has a name and a number. All she has to do is pick up the phone and call. When she does it quickly becomes clear that her birth mother has no wish to have contact. Rose is shattered. A stunned and depressed Rose attempts to keep her life together. Rose travels from Dublin to London in an effort to perfect her skills.

## SETUP AND DEPLOYMENT

The web application has been set up and deployed first locally on our individual systems basically to ensure that all the changes we made are running correctly on the server. After making sure that the results are satisfactory, it has been deployed for public use on Heroku - a cloud based platform to run and operate applications.

### Local host:

To run the application locally, the following steps can be followed:

1. Clone the github repository, to make sure that all the required programs and field are available on the user's machine:

`$ git clone https://github.com/muneshb/CourseProject.git`

2. Open the terminal and install all the required packages by running the file [install.sh](#) from the source folder:



`$ sh install.sh`

3. All the scraped information is available in the `./source/data` folder but if the user needs to scrape the most updated information from the source url, run the file [scrape.sh](#). The [movie\\_dir\\_scraper.py](#) and [page\\_scraper.py](#) files will run again and the results will be updated in the `./source/data` folder.

`$ sh scrape.sh`

4. Users can update the `movies_sample_queries.txt` with more queries of choice. To get the desired results for the sample queries, python file [search\\_rank.py](#) can be run and the results will be updated in [avg\\_p.txt](#), [ndcg.txt](#) and [rank\\_result.txt](#) files in the `./source/data` folder.

`$ python search_rank.py config.toml`

The steps 3-4 were mainly to run individual files and update the information in the data folder for improving the results for the application and end to end testing for calculating the mean .

5. For running the web application locally, the user needs to make sure to install all the packages from step 2 and then run the [run.sh](#) file in the source folder.

`$ sh run.sh`

The user will see the following result after running the script:

```
~/Projects/Text information system/final_project/CourseProject/source(venv) » sh run.sh
creating idx
now loading ranker
* Serving Flask app "library" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

6. The web application will be available at <http://127.0.0.1:5000/> .
7. Users can enter the search query and see the results for the top 10 movies. Each movie title leads to the RottenTomatoes movie page with all the movie information.

## Heroku App:

The application has also been deployed for public access on Heroku platform and can be accessed using the following url:

<https://not-so-rotten-tomatoes.herokuapp.com/>

## Steps for deployment:

1. Gunicorn is an application server for running your python application instance. Start the deployment by installing gunicorn which is a WSGI HTTP server, using:

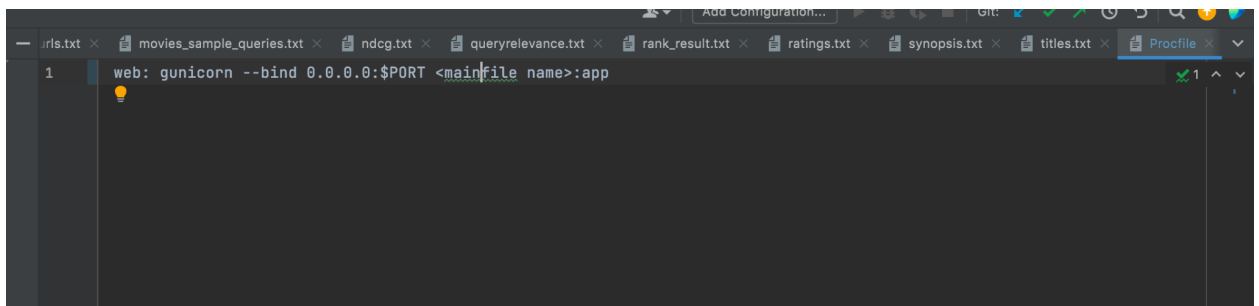
```
$ pip install gunicorn
```

2. Create the [requirements.txt](#) file to add all dependencies installed to run the application by using the following command:

```
$ pip freeze > requirements.txt
```

3. Create a new file, name it Procfile and save it in the root folder of the Github repo and type:

```
web: gunicorn --bind 0.0.0.0:$PORT <mainfile name>:app
```

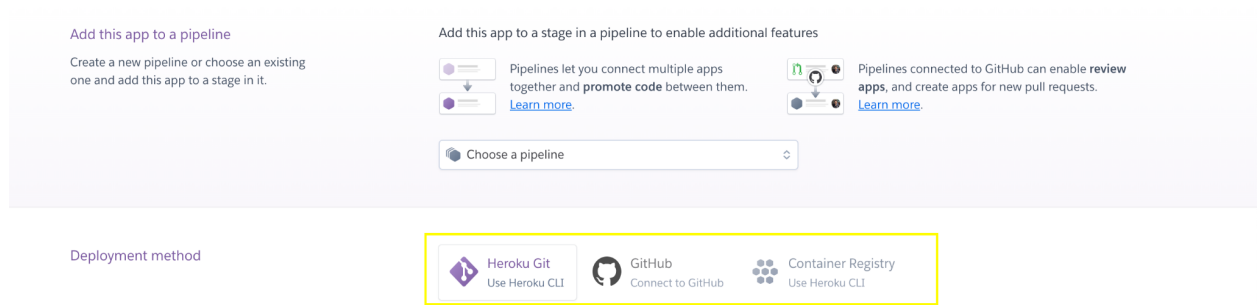


A Procfile is a text file (named Procfile) placed in the root of your app that lists the process types in the app. Heroku apps include a Procfile that specifies the commands that are executed by the app on startup.

4. Now login to [Heroku](#) and create a new app:

A screenshot of the Heroku 'Create New App' form. At the top, it says 'Salesforce Platform' and 'HEROKU'. Below that is a search bar with the text 'Jump to Favorites, Apps, Pipelines, Spaces...'. The main heading is 'Create New App'. The form has two main sections. The first section is 'App name' with a text input field containing 'new-app-name' and a red error message below it: 'new-app-name is not available'. The second section is 'Choose a region' with a dropdown menu showing 'United States'. Below these sections are two buttons: 'Add to pipeline...' and 'Create app'.

5. After creating the app on Heroku, user can choose the mode of deployment out of the options available:



### Heroku Git (using Heroku CLI):

1. For using Heroku Git, make sure that the package (main git repo) is version controlled by Git.
2. Install Heroku CLI (Command line Interface) on terminal:

For Mac OS : `brew tap heroku/brew && brew install heroku`

For other operating systems refer :

<https://devcenter.heroku.com/articles/heroku-cli>

3. Add a remote to the local repository. Run the following command on terminal to connect to Heroku app:

`$ heroku git:remote -a <your app name>`

4. Deploy the app to Heroku, using `git push` command to push the code from local repository's main branch to heroku remote:

`$ git push heroku main`

Use the above command to deploy the latest committed version of code to Heroku

## Github:

1. If the user is the owner of the Github repository, that repository can be directly linked to the Heroku application using “Connect to Github”.

The screenshot shows the Heroku deployment method selection interface. At the top, under 'Deployment method', there are three options: 'Heroku Git Use Heroku CLI', 'GitHub Connect to GitHub' (highlighted with a yellow box), and 'Container Registry Use Heroku CLI'. Below this, the 'Connect to GitHub' section is active. It prompts the user to 'Connect this app to GitHub to enable code diffs and deploys.' There are two input fields: 'Username on GitHub' (highlighted with a red box) and 'CourseProject' (highlighted with a yellow box). A 'Search' button is next to the second field. Below the inputs, there is a placeholder 'Username/github\_repository\_name' (highlighted with a red box) and a 'Connect' button (highlighted with a yellow box). A link 'Missing a GitHub organization? Ensure Heroku Dashboard has team access.' is also visible.

2. Once connected, choose the option for automatic deploys. This enables the code to be automatically deployed every time the user makes changes on the selected branch.

The screenshot shows the Heroku automatic deploys configuration screen. At the top, it says 'App connected to GitHub' and 'Code diffs, manual and auto deploys are available for this app.' Below this, there is a 'Connected to' section showing the repository name and a 'Disconnect...' button. A link 'Releases in the activity feed link to GitHub to view commit diffs' is also present. The 'Automatic deploys' section is active, with the text 'Enables a chosen branch to be automatically deployed to this app.' A blue box contains instructions: 'You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.' Below this, there is a section 'Enable automatic deploys from GitHub' with a note: 'Every push to the branch you specify here will deploy a new version of this app. Deploys happen automatically: be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.' A dropdown menu 'Choose a branch to deploy' is set to 'main'. There is a checkbox 'Wait for CI to pass before deploy' which is unchecked. Below it, a note says 'Only enable this option if you have a Continuous Integration service configured on your repo.' A button 'Enable Automatic Deploys' is highlighted with a yellow box.

The active deployment can be found under the Environments section of the user's Github repository.

**TEAM MEMBER CONTRIBUTION**

Team member	Contributions	Est. Time Contribution
muneshb2	<ul style="list-style-type: none"><li>• WebApp coding</li><li>• WebApp UI design</li><li>• RottenTomatoes scraper program to scrape urls from Top 100 Movies page</li><li>• Movie Critic Review scraping</li><li>• Wrote queries and provided explicit feedback for query relevance testing</li><li>• E2E testing</li></ul>	35 hours
puri6	<ul style="list-style-type: none"><li>• Search-rank program to create ranking results for each movie according to the sample queries and the query from the webapp.</li><li>• Calculate average_precision of the ranking for each query and the mean average precision for all the queries.</li><li>• Wrote queries and provided explicit feedback for query relevance testing</li><li>• Model evaluation using MAP and NDCG@k</li><li>• Movie Tomatometer Rating Scraping</li><li>• E2E testing</li><li>• Web application deployment using Heroku</li></ul>	35 hours
jr7	<ul style="list-style-type: none"><li>• Page scraper program to scrape content from each webpage url</li><li>• Wrote queries and provided explicit feedback for query relevance testing</li><li>• Model evaluation using MAP and NDCG@k</li><li>• Ranking algorithm tuning and assessment</li><li>• E2E testing</li></ul>	35 hours

## FURTHER IMPROVEMENTS

Through our application, “**Not So RottenTomatoes**”, we tried to improve users’ experience while searching for a movie according to their preference.

“Not-so-RottenTomatoes” can actually become “Not-So-<any platform>” as this is a generic system. We have tried to implement Intelligent Browsing and created a ranking system for a user to filter a movie based on the platform, genre or cast. With minimal changes, any other web platform can integrate this application to filter and rank any information like we implemented it for “Best movies in 2021 on RottenTomatoes”.

For instance our application can be adapted to below use cases:

- Best movies of all times on Rotten tomatoes-  
<https://www.rottentomatoes.com/top/bestofrt/>
- Barnes & Noble's Best Books of 2021  
[https://www.barnesandnoble.com/b/books/best-books-of-the-year-2021/barnes-nobles-best-books-of-2021/\\_/N-29Z8q8Z2w2g](https://www.barnesandnoble.com/b/books/best-books-of-the-year-2021/barnes-nobles-best-books-of-2021/_/N-29Z8q8Z2w2g)

We implemented our application for about 100 movies for now and returned Top 10 results but with more data, the ranking system can be improved to provide better results with improved precision with proper tuning.

Also, Google directly provides results in response to a vertical search query. If users try to search "science fiction movies on amazon" on Google, they get many useful pages. But the most accurate results will be displayed if there is a page available with a set of definitive answers. As a further improvement, users can leverage Google or information websites like <https://www.digitaltrends.com/> to evaluate the system's accuracy.

## REFERENCES

- Top 100 movies of 2021, Rotten Tomatoes:  
<https://www.rottentomatoes.com/top/bestofrt/?year=2021>
- Zhai, C. & Massung, S. (2016). *Text data management and analysis: A practical introduction to information retrieval and text mining*. ACM Book Series. Morgan & Claypool Publishers.
- N. J. Belkin and W. B. Croft. 1992. *Information filtering and information retrieval: Two sides of the same coin?* Commun. ACM 35, 12 (Dec. 1992), 29-38.
- Mark Sanderson. *Test collection based evaluation of information retrieval systems*. *Foundations and Trends in Information Retrieval* 4, 4 (2010), 247-375.
- Meta-ToolKit: <https://meta-toolkit.org/>
- Metapy Search and IR evaluation Tutorial :  
<https://github.com/meta-toolkit/metapy/blob/master/tutorials/2-search-and-ir-eval.ipynb>
- Implementation of the ranker functions:  
<https://github.com/meta-toolkit/meta/tree/master/include/meta/index/ranker>
- How to scrape websites with Python and BeautifulSoup :  
<https://www.freecodecamp.org/news/how-to-scrape-websites-with-python-and-beautifulsoup-5946935d93fe/>
- The Flask Mega Tutorial by Miguel Grinberg:  
<https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>
- Deploy app on Heroku using Git: <https://devcenter.heroku.com/articles/git>
- Deploying Flask app on Heroku using GitHub:  
<https://dev.to/lordofdexterity/deploying-flask-app-on-heroku-using-github-50nh>
- Deploy using Git - Deploy your Flask Application using Heroku-  
<https://www.youtube.com/watch?v=TEuPE5pUh2w>