**TV Finder**

**Information Retrieval and Web Agent Final Report**

Section 466

Tammy (Siyu) Li [sli187@jhu.edu]

Yulun Zeng [yzeng33@jhu.edu]

­­­

**Project Summary**: we aim to provide an efficient and user-friendly platform for TV series enthusiasts to explore and search for specific series or retrieve a list of related recommendations. With an internal web crawler, we build the TV series data and help users check and search for information, by querying titles, actors, and other information, such as countries.

**Web Interface**: <http://127.0.0.1:5000> [running locally]

**Project Description**

Searching and querying on single platform TV series can provide lists of recommendations. However, users may encounter spelling mistakes and retrieving information from a single source platform can be limited. A combined source with similarity calculations in the backend could generate more useful retrieved information.

Driven by such an idea, we will focus on crawling and searching the TV series domain. Specifically, we will integrate data by crawling from three platforms: bmovies, dopebox, and moviecrumbs. Search results will be returned a list of the top 50 series that have the highest similarity score with the user’s query. Thus, users are able to search by titles, actors, and other information without concerns with spelling errors or limited data sources.

**User Guide**

The user will start with loading and preprocessing TV series data by running *cd website*, then *python tv\_crawler.py* and *python processData.py*. Then the user could run the frontend by running *python main.py* in the root directory. In the local terminal, the user will see a message Running on http://127.0.0.1:5000/, open this link in your browser, and the user will be able to access the website and perform information searching on the website.

The website contains a form, which asks for user inputs for titles, actors, and other information in text form, and three buttons corresponding to three functionalities: Search, Clear, and Refresh Database. After clicking the Search button, results will be displayed in a list with posters of series and brief information. Users can be navigated and watch the series by clicking the Go button on the top right corner of each result. To start with another search, users could click on the Clear button to clear all inputs and outputs. Users could also click on Refresh Database to fetch the most recent database on the backend in order to be consistent with data updates.

**Project Strengths**

1. Accurate Search Results – The application uses a cosine similarity score with TF-IDF weights assigned to retrieve the top 50 most relevant TV series based on the user’s search query. This relevance calculation combination is shown to provide the highest relevant recall and precision. Using similarity calculation instead of direct keyword search ensures that users receive accurate search results even if user inputs contain spelling mistakes.
2. User-friendly Interface– The frontend web application is mainly created using a Bootstrap and results are displayed with visually appealing posters of series and brief information. Such a clear designed interface makes it easy to use.
3. Efficient Crawling and Indexing – The application crawls TV series information from three different domains, ensuring that users have access to a wide range of data. Locations of data are pre-determined, although not a good choice for constantly updating websites, it can significantly speed up the information extraction process. The data is then processed into efficiently indexed and organized JSON formats to allow for quick retrieval of search results.
4. Scalability – The Flask framework used in this project is highly scalable and can handle a large number of users and requests. This ensures that the application can be easily scaled up to meet the needs of a growing user base. In addition, Flask allows programmers to create responsive web apps without using any JavaScript. And it is much more efficient in HTML rendering and deployment. We can also run Python functions without the need of setting up a cloud server.
5. ﻿﻿﻿Flexibility – The application is highly flexible and can be easily customized to meet the specific needs of different users. For example, users can search for TV series by title, actor, or other information, depending on their preferences.
6. Consistence – The application has a Refresh Database functionality which allows users to manually update data in order to keep up with the latest TV series.

**Limitations and Future Improvements**

1. Limitation in Search Approach – The accuracy of the similarity score calculation depends highly on the quality and quantity of data available on the crawled domains. And due to the large data set(over 10,000 documents in data.json), it takes enormous computation time to process documents, as a result, we were not able to implement further extensions such as N-gram(which could be very beneficial for word ordering).
2. Limitation in Inconsistent Update – The crawling process may take time to load data and process because it is all done in one session, so users will not be able to do anything else while the application is still gathering and processing data, and the application may not be able to automatically keep up with the latest TV series updates. This issue can be fixed if we have a dedicated server, which can enable background refresh while handling HTTP requests.
3. Improvements of User Login – Implementing user authentication and authorization to restrict certain features to registered users only can help increase the security of data access such as by controlling token expirations.
4. ﻿﻿﻿Improvements with Larger Data – Add more domains to crawl and increase the database of TV series so that data size is also scalable as the number of different users increases.
5. ﻿﻿﻿Improvements on User Interface – Three more features could potentially be added to make the interface more user-friendly. Implement a feature to allow users to rate and review TV series. Such feedback could be viewed as relevance feedback and help increase recall by training the search agent to learn from past results. Moreover, add another feature to recommend similar TV series to the users based on their search history and ratings. We could also allow users to add TV series to their watchlist and receive notifications when new episodes are released.
6. Improvements on Result Organization – Currently only the top 50 results are displayed and are sorted by relevance. Future implementation can also add other sorting and filtering method to enable customizable output.

**Output Evaluation**

Output is evaluated by the hand-written testing query.raw and query.rel. The application performs cosine similarity and TF-IDF with a weighting of (title = 4, actor = 2, other = 1). An output matrix containing different precision and recall calculations is generated, which shows good numerical values of above 90% and 80%. Based on analysis of the data, we could conclude the TV series information retrieval system effectively retrieves relevant results for users based on their search queries

**Citation**

Inspired by the previous homework of the course, the output evaluation incorporated partial codes from our own codes from HW2 of this course. The crawling procedure incorporated partial codes from our own codes from HW4 of this course.

**Screenshots**

[1] Flask Python Frontend Design

A picture containing text, website, software, web page

Description automatically generated

[2] Bootstrap Frontend Code

A screen shot of a computer program

Description automatically generated with low confidenceA screen shot of a computer program

Description automatically generated with medium confidence

[3] Data Crawling and Processing

A screen shot of a computer

Description automatically generated with medium confidenceA screen shot of a computer program

Description automatically generated with low confidenceA screen shot of a computer code

Description automatically generated with low confidenceA screen shot of a computer program

Description automatically generated with low confidenceA screen shot of a computer program

Description automatically generated with low confidenceA screen shot of a computer program

Description automatically generated with low confidence

A screen shot of a computer program

Description automatically generated with medium confidenceA picture containing text, screenshot, software, multimedia software

Description automatically generated

[4] Partial Processed Raw Data Output Format

A screenshot of a computer

Description automatically generated

[5] Searching Algorithm Incorporating Weighted Similarity Calculation

A picture containing text, screenshot, software

Description automatically generated

[6] Partial Test Output Evaluation Table

A screenshot of a computer screen

Description automatically generated with low confidence