Flickpedia: Final Project Report

Shashank Bandaru
sbandar1@vols.utk.edu

Isha Bhandari
ibhandar@vols.utk.edu

Jason Choi
jchoi38@vols.utk.edu

Justin Henley
jhenley9@vols.utk.edu

Pooja Masani
pmasani@vols.utk.edu

Ann McClure
amcclu13@vols.utk.edu

Nayana Patil
spatil12@vols.utk.edu

***Abstract*—This project aims to create a web-based tool that allows users to search for episodes of the TV series "Friends" by entering quotes from the show. By processing transcript data, the platform identifies and displays the corresponding season and episode, making it easier for fans to locate their favorite scenes.**

I. Objective

The primary objective of our project is to develop a web-based platform that enables users to easily identify specific episodes and seasons of television series by entering memorable quotes. Our goal is to provide a seamless and efficient user experience, making it easy for users to search for episodes without needing to sift through entire transcripts manually. The platform will feature an intuitive search interface that will quickly match user input to the corresponding season, episode, and other relevant details. This project is designed with accessibility and ease of use in mind, ensuring that users of all technical skill levels can interact with the platform effortlessly. By hosting this tool on a website, we aim to make the service widely available and highly responsive across different devices and browsers.

A. Future Enhancements

As the project evolves, we also plan to explore potential integrations with machine learning models to improve search accuracy and responsiveness. Beyond direct quote searches, we would like to implement an advanced search feature that allows users to input more general descriptions or summaries of scenes. This feature would leverage natural language processing or refined search algorithms, enabling users to find episodes even when they can't recall the exact wording of a quote. This enhancement would significantly improve the flexibility and power of the platform, catering to a broader range of user inputs. Additionally, incorporating external APIs for episode metadata and further refining the UI for a more dynamic and engaging experience is part of our long-term vision.

II. Data

We used a data set from Kaggle called "FRIENDS TV Series - Screenplay Script" [1] to get each episode's info and transcripts. We then created a database connection to store each aspect (show, season, episode, and transcripts). To prepare the data for search and retrieval, we formatted each transcript by standardizing the text and removing unnecessary or missing characters. This ensured consistency and improved the accuracy of search functionality.

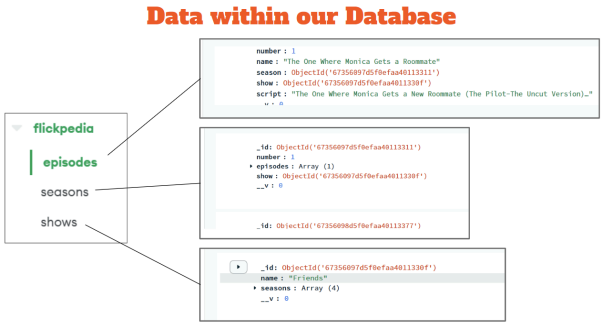

Fig. 1: Database structure.

For each new show, a show object was created in MongoDB. This object contained a unique ID and the show's name, which served as the primary reference point for linking the show's data to its corresponding seasons and episodes. When creating a season, the show ID was included in the season object, allowing MongoDB to store the association between the show and its seasons. Each season object also contained an array that would hold the IDs of all its episodes.

When adding an episode, the episode object included key details such as the episode name, episode number, show ID, and season ID. Upon creation, the episode ID was automatically added to the episode array within the appropriate season object, ensuring the data's relational integrity. MongoDB's structure made it easy to validate these relationships, ensuring that each episode was correctly linked to its respective season and show.

Quality assurance was performed to test the relationships between shows, seasons, and episodes, verifying that IDs were correctly assigned and all data was integrated seamlessly. This approach supported efficient querying for the current dataset and also made the system scalable for future expansions, such as adding new shows, seasons, and episodes.

III. Models and Algorithms

To implement the search functionality for our project, we utilized the Fuse.js library. Fuse.js is a lightweight JavaScript library designed for fuzzy searching, which allows approximate matches rather than requiring exact matches. This approach enhances the flexibility of the search tool, making it more user-friendly and accommodating to errors or partial inputs.

A. Key Features of Fuse.js:

- **Fuzziness Score**: Fuse.js calculates a fuzziness score by evaluating the location, distance, and threshold of the search term in the dataset. This ensures that approximate matches are identified even if the user's input contains minor errors or variations.
- **Key Weight (Optional)**: The library allows the assignment of weights to specific keys or fields, enabling us to prioritize certain attributes over others. For instance, we can assign higher relevance to matches in the "Title" field than in the "Script" field, improving the accuracy of results.
- **Field-Length Normalization**: Fuse.js applies field-length normalization to rank shorter field matches higher than longer field matches. For example, if a search term matches a "Title" field rather than a longer "Script" field, the "Title" match will score higher, reflecting its higher relevance.

B. Benefits of Fuse.js:

- **Ease of Implementation**: The library is straightforward to integrate, with extensive documentation and configurable options that allow us to customize its behavior for our project's needs.
- **Flexible Matching**: By scoring results based on multiple factors, Fuse.js provides a robust framework for returning relevant results even when the search term is not an exact match.
- **Enhanced User Experience**: Fuzzy searching with configurable weights and field normalization improves the overall user experience by delivering accurate and prioritized results.

This algorithm serves as the backbone of our search functionality, enabling users to efficiently locate quotes and their corresponding episodes, even with incomplete or imprecise inputs. In our project, we excluded the use of key weights from users and disabled the calculation of a fuzziness score with the location option.

IV. Results

Our project successfully delivers a functional search tool that meets the defined objectives. The finished product allows users to enter any word or quote and receive precise, detailed results from the Friends TV series dataset.

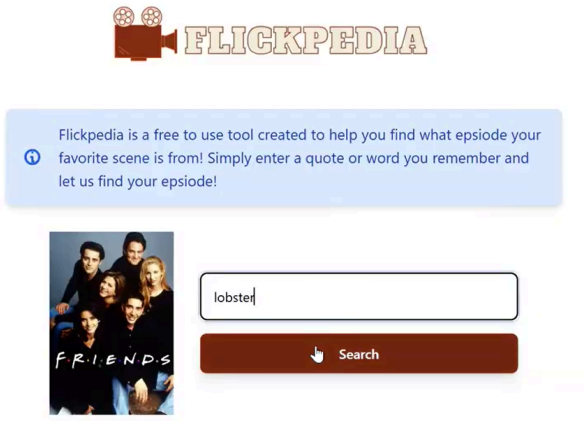

Fig. 2: Search Page: single word search.

A. Key features include:

- **Search Results**: The tool returns a list of matches, each displaying the corresponding episode name, season number, and episode number. It highlights the specific line(s) within the transcript that match the entered quote or keyword.

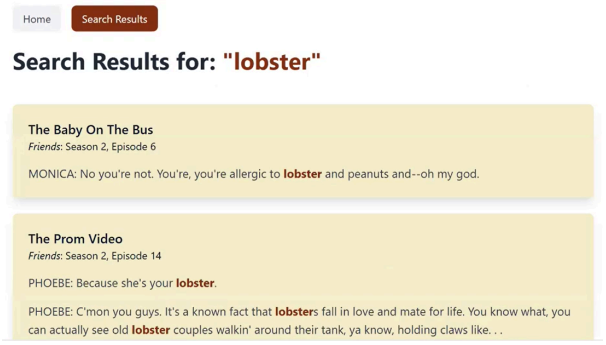

Fig. 3: Results Page: single word search.

Fig. 4: Results Page: quote search.

- **Comprehensive Matching**: The search engine is robust, returning all relevant results that match the entered word or quote, ensuring comprehensive dataset coverage.
- **Interactive Transcript Viewer**: Clicking on a result opens a dedicated box containing the full transcript of the selected episode. This feature provides users with

the complete context of the quote, enhancing usability and user experience.

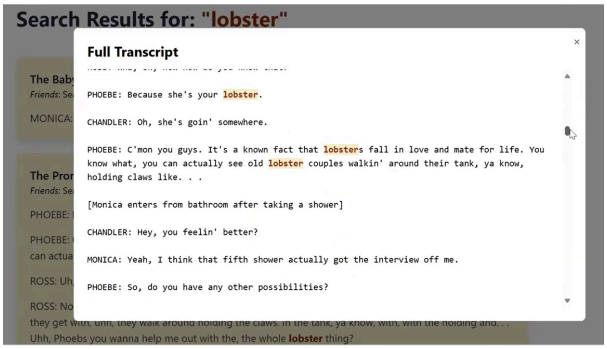

Fig. 5: Transcript Page: single word search.

By integrating MongoDB for data storage and retrieval, the application ensures fast and reliable performance even with large datasets. User testing confirmed that the tool is intuitive and responsive, effectively bridging quotes to their corresponding episodes in the Friends TV series.

V. Primary Issues

The development process encountered several challenges related to data availability, formatting, and stability:

- **Limited Availability of Transcripts**: Transcripts for TV shows, including Friends, are not readily available in structured formats. While a web scraper could automate the process of acquiring transcript data, building such a scraper is a substantial project on its own, requiring additional time and resources.
- **Improper Formatting and Grammar**: Many available transcripts suffer from inconsistencies, such as improper formatting or grammatical errors. These issues introduce noise into the data, which can reduce the accuracy of the search functionality by failing to match quotes as expected.
- **Special Characters and Program Stability**: The presence of special characters in the dataset caused unexpected issues, including crashing the program during testing. These characters required extensive pre-processing and error handling to ensure the stability of the search engine and prevent interruptions to user operations.

VI. Future Work

Our project lays the foundation for a robust search tool that links quotes to their corresponding episodes. Several ideas and improvements have been identified to expand its scope and functionality in the future:

- **Expanding to Multiple Shows**: Extend the tool to include transcripts from other popular TV shows, allowing users to search across a broader range of series. This would require scalable data integration and enhanced database design to manage diverse datasets effectively.
- **Advanced Filtering Options**: Implement user-friendly filters to narrow search results based on parameters such as show title, genre, release year, or season. This enhancement would significantly improve usability by enabling more refined searches tailored to user preferences.
- **Improve Searching Algorithm**: Adding a search function for users by adding features to specify search criteria based on characters, seasons, episodes, etc. We can improve the searching capabilities by testing and implementing other search libraries aside from Fuse.js like Algolia or Elasticsearch.
- **Enhanced Contextual Information**: Integrate external APIs to fetch additional details, such as actor information, production notes, or behind-the-scenes trivia, enriching the user experience and making the tool more interactive.
- **Dynamic Web Scraper Development**: Build a web scraper to automate the collection of transcript data from various sources. This would streamline data acquisition for future expansions and ensure consistency in formatting and structure.

These future developments aim to transform the current project into a comprehensive, user-centric platform for TV series enthusiasts.

VII. Organization Chart

A. Project Timeline

- Oct 3: Complete project proposal, outlining objectives, datasets, and methodologies.
- Oct 15: Finalize data collection and begin preprocessing, including text normalization and data cleaning.
- Nov 22: Deliver a working prototype, including a functional search feature that identifies relevant episodes based on user-entered quotes.
- Nov 30: Finalize all core features and polish UI/UX; complete user testing.
- Dec 1: Prepare and finalize the presentation, including analysis of project outcomes, insights, and future developments.
- Dec 3: Deliver the project presentation and submit all required materials

B. Member Responsibilities

Our group divided responsibilities to ensure efficient progress and collaboration, focusing on three primary components: User Interface (UI), backend development including web scraping, and the logic for connecting quotes to episodes. Each member is assigned based on their strengths and expertise, fostering a balanced workload and allowing us to leverage individual skills effectively.

a) *User Interface (UI):*

- **Ann, Pooja, and Justin** designed and developed the frontend, creating a user-friendly interface for seamless searching and displaying episode results. They focused on layout, aesthetics, and interaction design to provide an intuitive experience for users.

b) *Web Scraping and Backend Development:*

- **Nayana, Isha, and Shashank** are responsible for building the backend infrastructure, including web scraping to collect episode transcripts. They ensured the data was processed, cleaned, and stored efficiently to support the search functionality.

c) *Connecting Quote to Episode:*

- **Jason** focused on developing the core logic that links user-entered quotes to the corresponding episodes. This involved searching the processed scripts and integrating the results with the UI, as well as refining search algorithms for accuracy.

d) *Quality Assurance:*

- **Justin** focused on stress testing the code to ensure odd or incorrect entries would not appear for the user. This included simulating edge cases such as partial quotes, quotes with typos, and inputs with special characters to verify robust search functionality and prevent unexpected outputs.
- **Jason** concentrated on validating data integrity within the MongoDB database. He ensured that all transcripts and metadata were correctly stored, cross-referenced, and retrieved without discrepancies. Additionally, he reviewed query performance to optimize response times for user searches.
- **Ann and Pooja** performed usability testing on the user interface to identify potential flaws in layout or functionality. Their work included gathering feedback on the search experience and ensuring a seamless flow between entering a quote and receiving accurate episode results.
- **Nayana, Isha, and Shashank** led the efforts in debugging and resolving backend issues during integration, focusing on error handling and ensuring smooth communication between the frontend, backend, and MongoDB database.

References

[1] B. Densil, "Friends TV series - screenplay script." [Online]. Available: https://www.kaggle.com/datasets/blessondensil294/friends-tv-series-screenplay-script?select=S01E01+Monica+Gets+A+Roommate.txt