FilmWord: A Semantic Movie Search Engine

Seth Polsley, Ninghao Liu, and Someiah Bahrami

1. Introduction and Motivation

The entertainment industry is massive, with the US media and entertainment market alone posting over $500 billion dollars in revenue in 2014[1], a number which has only grown in recent years. Because of the popularity of media such as films in today's culture, multiple related markets have appeared taking forms such as movie reviewers like MetaCritic, knowledge bases like iMDb, or rating sites like RottenTomatoes. However, despite the popularity of these tools, searching and exploring is still extremely limited. Some websites allow searching by genres or may have a small section of "similar" films on a movie page, but their search feature is often restricted to names of films or people, meaning that many query terms people associate with a film that are not directly in the title will return no results.

FilmWord is a semantic movie search and exploration engine. It places emphasis on the words users have associated with a given movie based on reviews and plot synopses, while also supporting traditional searching methods using title and actor names. This combined approach allows FilmWord to return a set of highly relevant films to very vague queries. In addition to supporting flexible, free-text searches, FilmWord uses powerful visualizations to encourage users to explore. These include word clouds and charts that link to new, related query terms. Furthermore, related films are recommended from each movie's page based on popular terms associated with the film, capturing subtleties beyond similarities in genre or cast.

In the remaining sections, we'll briefly introduce some related work in the area of semantic retrieval and explore some of the existing movie search tools provided by popular websites today. After that, we will cover the implementation of FilmWord in greater detail, discussing the data acquisition, analysis and modeling methods, and the application. Finally, we'll close with some results studying the system performance amidst some concluding findings.

2. Prior Work

2.1. Semantic Retrieval

In semantic retrieval, given the query input by a customer, we want to extract the semantic meaning behind the query words and return the films that match relevant concept. The Latent Semantic Indexing (LSI) model [1], which is based on the SVD of the document-word matrix $X=UDV$, is an option for solving such problems. Here U is the document-feature matrix, D is a diagonal matrix and V is the feature-word matrix. Films of similar genres have similar latent semantic representation in U, and are close to each other in the latent space [2]. Another popular model for such problem is Latent Factor model [4], where $X=PQ$. Here P (document-feature matrix) and Q (feature-word matrix) are nonnegative matrices. We implemented the Latent Factor model with nuclear norm normalization. Given a query, the vector q is constructed as follows:

[1] http://selectusa.commerce.gov/industry-snapshots/media-entertainment-industry-united-states.html

$$q = \sum_{i=1} q(w_i)$$

where w_i is the i^{th} query word, and $q(w_i)$ is the corresponding column in Q. The relevancy score of a film j to the query is:

$$c_j = p_j \cdot q$$

so the films with larger c_j are returned on the top.

2.2. Crowdsourcing

After building the Latent Factor model, we found a more efficient alternative way for capturing the desired data. We found that the latent information of a film is already implicitly contained in the reviews provided by users and critics on websites like iMDb and RottenTomatoes. Given a user query, we can retrieve relevant films based on the the text in their reviews. This mechanism falls into the category of crowdsourcing. Defining the crowdsourcing system is a little challenging given the broad applications of crowdsourcing, but we apply the definition provided by [3] which refers to the system that "enlists a crowd of users to explicitly collaborate to build a long-lasting artifact that is beneficial to the whole community".

2.3. Existing Movie Search Engines

Most existing movie search tools have no support for semantic retrieval. For instance, RottenTomatoes, which aggregates reviews from critics and users into an overall film "freshness" score, searches only on movie title. The Internet Movie DataBase (iMDb) houses enormous amounts of factual data regarding almost every movie or television show ever created; iMDb searches only on title and actor names. This limitation is nearly all-pervasive in the movie search engine area, but Google recently began providing slightly more flexible movie searching. A Google query like "recent sci fi" films can return popular films of the specified genre by year. Unfortunately, that is about as complex as a query can be before no direct results can be found. FilmWord differs from all of these systems by having a semantically-inspired model of each film in its index.

3. Implementation

FilmWord was built in stages, beginning with the acquisition and analysis of our data. This data was then stored in the open source searching system Apache Solr. Solr powers the web frontend of FilmWord, which is written in Javascript and HTML/CSS. More details on each stage of the implementation are given below.

3.1. Data Collection and Modeling

The data collection consisted of gathering movie information like name, cast, writers, directors, and reviews by crawling RottenTomatoes top movie lists. As mentioned previously, FilmWord places emphasis on words of users to support semantic searches, so it is essential to find the keywords and their related weights from the reviews. We performed some additional analysis in Python in order to build an indexable model with the goal of storing each film with a collection of popular keywords. This model allows FilmWord to much more flexible queries than typical movie search engines.

For all reviews of a given film, we constructed a mini-corpus of terms that were tokenized and lemmatized. Initially, these terms were also stemmed, case-normalized, and matched against words in a large spell checking lexicon. Unfortunately, this level of processing lost too much key information between films; words like “terminator” and “terminal” may both stem to “term” but are very different. Hence, we stopped at lemmatization for the text normalization. Each mini-corpus was converted into a Frequency Distribution using the Natural Language ToolKit (NLTK) module for Python. The top 30 terms were then selected and filtered to only words of length 4 or more characters. The keywords were stored with the film information and reviews directly into Apache Solr using the Python library SolrPy.

3.2. Methods and Algorithms

FilmWord is a web application. All retrieval and processing is handled by the Java-based Solr on the backend while the frontend performs final processing in Javascript. Users interact through two main pages: a search page and a movie page.

3.2.1. Search Page

The search page is the heart of FilmWord. We elected not to distract the user with fields like “Search by title” or “Search by actor”, opting instead to present a single, clean, integrated search bar. Once a user starts to type, query suggestions are provided through three tiers of interaction with the Solr server. First, Solr’s “suggest” feature is used to find potential film title matches through a fuzzy analyzer. Second, suggest provides possible actor names using a word-level analyzer to find names that match any part of the search query. Both of these suggestion features are powerful, but they cannot provide suggestions in all cases since the search bar is free text. For this reason, we rely on Solr’s “spell” feature as a third tier. The spell checker is based on a dictionary of around a million user reviews, and can provide common term suggestions for nearly any input the user provides.

After a user presses enter or selects from the dropdown menu of suggestions, a query is sent to Solr. Queries are themselves performed in two tiers. First, a direct match is sought in the collection of movie data. This includes matching with titles, actor names, and terms in the synopses or keywords lists. Keywords already provide some semantic context to search terms at this stage, but the next tier can handle even more vague queries. Should no matches be found in the movie collection, an altered query is sent to the database of user reviews. Here, we seek reviews containing the query terms, which are then grouped into potential film matches according to Solr’s “grouping” feature. This level of searching allows users to find films by terms which may be commonly associated with a film that are not popular enough to be classified as a “keyword” in our model.

From a user’s perspective, this experience is transparent. Users simply input a query, receive appropriate suggestions, and get a list of relevant films. Most interaction on the search page is concerned with the resulting movies and the visualizations. Movies are listed on the left in order of relevance, and visualizations are provided on the right. With each movie is the critic,

audience, and overall score (computed as 1-part critic score to 3-parts user score); the release year; and the synopsis.

Two major forms of visualization are provided with each query. The first is a set of two donut charts which shows the distribution of genres and keywords in the result set, using Solr's "facets" feature. The second visual is a word cloud for each film. These word clouds are weighted according to the keyword frequency in the user reviews, and they immediately show the most popular terms associated with each film on the page and their relationships with each other. To make interaction even simpler, users can click on any term of interest and receive a new result set for the selected term. This makes exploring very easy, and users can learn a lot of information about related films without ever leaving the search page.

3.2.2. Movie Page

Once the user wishes to investigate more about a given film, he or she can click on any movie in the result set from the search page to bring up the movie page. Movie pages are dynamically generated for each film. All of the database information is displayed, again including the title, ratings, year, and synopsis. The complete synopsis is shown on a movie page, along with all other information like the film's genre, directors, writers, and cast. Rather than being static text, each of these items links to a new query on that term. This allows users to return a set of films associated with a given actor simply by clicking on a name or to find other films in the same genre. The word cloud based on keywords is also shown on the movie page; because there is more screen space, the word cloud is much larger so users can see and interact with more key terms.

Figure 1: A portion of the movie page for "Monsters, Inc."

In addition to receiving all the movie information from Solr, two other features are used on the movie page. First, the "You May Also Like" section is created using Solr's "More Like This" feature on the keywords field. This allows FilmWord to provide recommended similar films that capture much more information than just genre or rating. For instance, keywords allow differentiating animation films like "Tangled" and "Nightmare Before Christmas" so that both

films have appropriate recommendations. These recommendations are also very effective at finding other films in the same series or projects with the same actors, directors, or themes.

Finally, the movie page also provides a mechanism for viewing the reviews for a film. Again, facets are used to generate a histogram of ratings from 1-5 stars, and Solr's sorting and filtering features are used to show both a positive and negative review directly on the movie page. By selecting the "See all reviews…" link, users can read each review stored for a film.

4. Results and Findings

We have seen in the previous discussion of related works and implementation how FilmWord differs from other movie search systems, but let us now consider some specific examples and results to evaluate the overall system performance. Because Solr is very adept at matching key terms, a sample search like "lord of the rings" can return all three Lord of the Rings films (including the three Hobbit films) at the top of the list. Likewise, searching by an actor or director name will return related projects. These types of searches are simple and readily available through RottenTomatoes and iMDb since they only match against title terms or names.

FilmWord is a much more powerful search engine when users are putting in free text terms where semantics matter. For instance, a user putting in the query "wands" may be anticipating films which have wizards and magic. Rotten Tomatoes simply returns results based on the movie titles (maybe with some spell checking), while FilmWord recognizes the magic flavor of "wand" and returns the Harry Potter movie series. Because RottenTomatoes could not find any English films with the term "wands" in the title, several obscure German movies were the only results returned, which are unlikely to be relevant to the user.

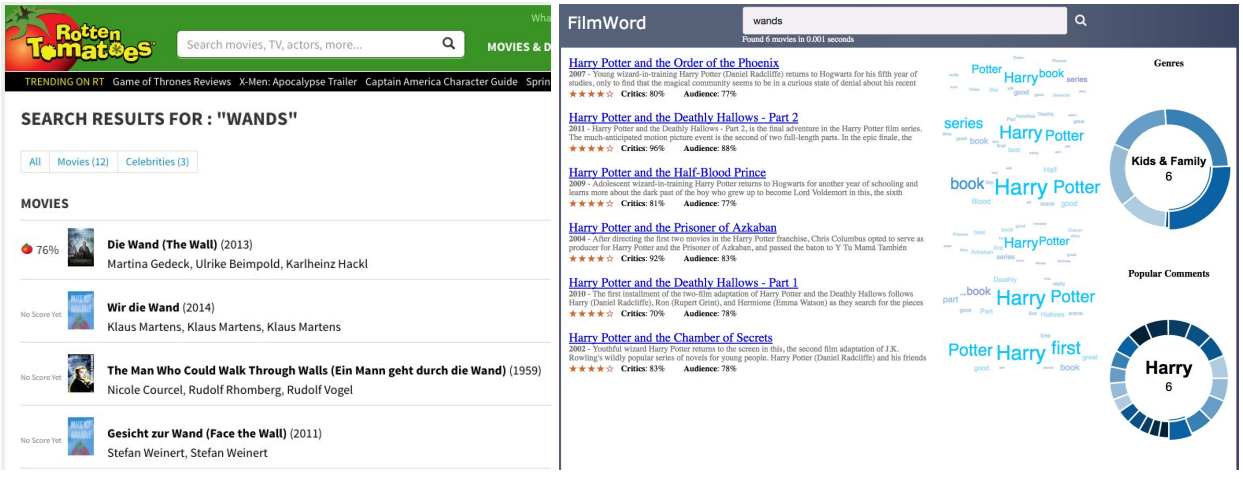

Figure 2: A comparison of the results returned by RottenTomatoes and FilmWord for the query "wands"

FilmWord can also make use of keywords to find relationships among films where none would otherwise be found. For instance, the director Tim Burton and actor Johnny Depp have collaborated on a number of movie projects, which could be useful information to a user trying to find a set of related films but not ones in a single series. If we input "burton and depp" in this

case, we can assume the user wants films directed by Tim Burton and acted by Johnny Depp. In this case, the search engine in IMDB can only provide an episode name from a small television series that happens to have both these terms. However, FilmWord successfully returns the films we want, as we see in the movie page of “Sleepy Hollow”.

In fact, the user doesn’t need to know about the frequent collaboration between Tim Burton and Johnny Depp to get relevant films in FilmWord. The “You May Also Like” section on any one of their films’ pages will have automatically found the similarity to other Burton and Depp films and recommend them to the user. As we mentioned before, this feature is also very good at finding other films in a series while still capturing the subtle differences among films.

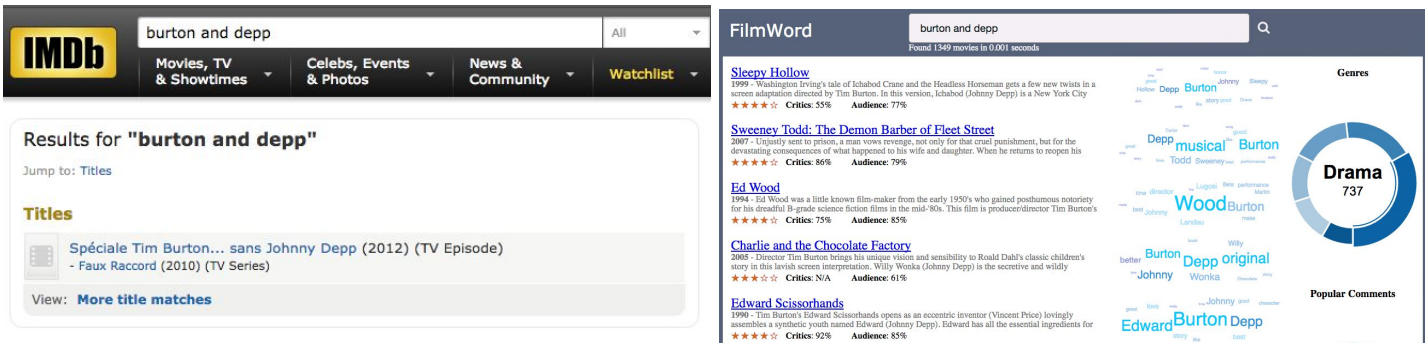

Figure: A comparison of iMDb and FilmWord when querying “burton and depp”

5. Conclusion

Given the size of the movie and entertainment industry today, it is perhaps a little surprising that movie search engines are still so naive that they can only perform matching against direct fields. However, semantic retrieval and recommendation systems are two fields that have only recently began to see major advances, at least to the comparatively ancient movie industry. In this work, we show that by using the richness of data like textual movie reviews, a system can learn at least some level of semantic understanding of user queries. FilmWord combines this understanding with the existing field-matching searching of systems today to give users a simple, streamlined movie searching and exploring experience.

References:

[1] Deerwester, Scott, et al. "Indexing by latent semantic analysis." *Journal of the American society for information science* 41.6 (1990): 391.

[2] Xu, Wei, Xin Liu, and Yihong Gong. "Document clustering based on non-negative matrix factorization." *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003.

[3] Doan, Anhai, Raghu Ramakrishnan, and Alon Y. Halevy. "Crowdsourcing systems on the world-wide web." *Communications of the ACM* 54.4 (2011): 86-96.

[4] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 8 (2009): 30-37.