Movie Search Engine Based on Description

Dongjian Chen, Ren Wang, Yumeng Shao

December 16, 2019

Abstract

In this report, we describe our work on description-based movie retrieval. Our search engine is motivated by the idea of helping people retrieve movies when they could only remember fragments of the movies. We adopted a dataset with movie titles, descriptions and keywords as our search database. BM25+ function is adapted after we preprocess the dataset with with tokenizing, stemming ,stopword and punctuation removal, and keyword addition. By calculating the harmonic mean of the ranks of the desired answers in our test cases, we are able to evaluate our system and conclude that our search engine has a fairly good performance.

Keywords

Movie, BM25+, Information Retrieval, Search

1 Introuction

Watching movies has become one of the most popular ways of entertainment, and there are hundreds and thousands of new movies being released every year. The information we recieve is explosing and the challenge of remembering the right title for movie plots is becoming larger and larger, since an average person could have watched dozens or hundreds of movies. It is not rare for someone to bump into an annoying situation when they could only remember a rough story or some segments of the movie, but just could not recall its name. In such cases without a specialized movie search engine, existing search websites will return various results that include not only movies and the search results would not be very ideal with queries composed of common terms but has specialized meanings and indications when it comes to movies.

As a result, we we aim at helping people retrieve the film they want when they could only remember a part of the story plot or segments like character names and special props. The search engine we designed and implemented helps with finding the corresponding movies with description or keywords provided by the user that specialize in movies.

2 Design Description

The whole process of our design process including getting data, processing data, selecting model and building interface. And the working process of our final design is shown as following figure:

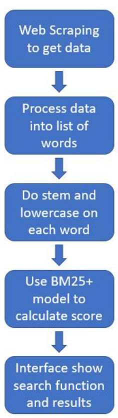

Figure 1: Working process of our searching system

2.1 Dataset

We obtained our data from two sources:

Kaggle movie dataset: (`https://www.kaggle.com/rounakbanik/the-movies-dataset#movies_metadata.csv`)

IMDB website (`https://www.imdb.com/?ref_=nv_home`)

We first download a csv file from Kaggle with data about overviews and imdb link for each movie. We first try to use these overviews as documents for queries. Then we find that the performance of searching can't fulfill our satisfication so we try to find more documents. Finally, after tring the summary on wikipedia and total scene description on IMDB, we choose keywords on IMDB and use python webscraping to get these data.

2.2 Basic Data processing

In this project, our retrieval system is built to retrieve movie by its description and our data document for each movie contains its overview and keywords related to it.

We first scrape keyword data from a website and combined it with the overview of each movie from the Kaggle dataset into one string. Then, we tokenize the string into words and do stem and lowercase on each word and get lists of words for each document. Finally we remove stopwords and punctuations, and the document is then ready for query. The above mentioned steps are also applied to query sentences.

2.3 Advanced Data processing

We tried two advanced methods on data to see if the performance of search can be enhanced.

First, we thought of adding the synonyms of each words in query so that those words with similar mean as query words in documents would not be missed. For each query we used nltk wordnet to get synonyms for each word in the query and added them into the query. However, the performance become worse instead of bettern. We thought that may be adding synouyms made the query vaguer so that more unrelated results are searched.

Also, we thought of adding weight to different kind of words in query. But that will led to another question that how should we judge the importance of different words. We thought of using part-of-speech tagger to tag each words in query and give more weight to adjectives and verbs. However, there is no distince improvement in the performance of our system. We can do further improvement on our system by trying to find more proper weighting methods on queries.

2.4 Model

We first use BM25 score function which is use to give score to each query-document pair by the term frequency(TF) of each word in the query and inverse document frequency (IDF) as following:

$$S = \sum_{t \in q} \cdot \left[\frac{(k_1 + 1) \cdot c_t^d}{c_t^d + k_1 \cdot \left(1 - b + b \cdot \frac{l_d}{L}\right)} \right] \cdot \ln\left(\frac{N+1}{N_t}\right)$$

However, the performance of BM25 score function is not so good so we want to find a more proper score function. We tried BM25+ function. The differences between BM25+ and BM25 are that BM25+ add an additional parameter and consider the frequency of each words in query. Since the performance of BM25+ is better and can relatively meet our needs, we finally choose it as our final score function (Inspiration from `https://www.eecis.udel.edu/~ypeilin/pub/ictir2016_long.pdf`) which is :

$$S = \sum_{t \in q} \frac{(k_3 + 1) \cdot c_t^q}{k_3 + c_t^q} \cdot \left[\frac{(k_1 + 1) \cdot c_t^d}{c_t^d + k_1 \cdot \left(1 - b + b \cdot \frac{l_d}{L}\right)} + \delta \right] \cdot \ln\left(\frac{N+1}{N_t}\right)$$

This BM25+ function add two more parameters to change the weight of TF and IDF in the function and it make our performance better in our test query set. The performance comparison and analysis will be detailedly discussed in Result and evaluation part.

2.5 Interface

We have also built an interface for users to input the description they remember and will return the result movie name and simple overview. The movie with best score will be shown on the page and user can see more results with lower scores if he slide down the page as following graphs:

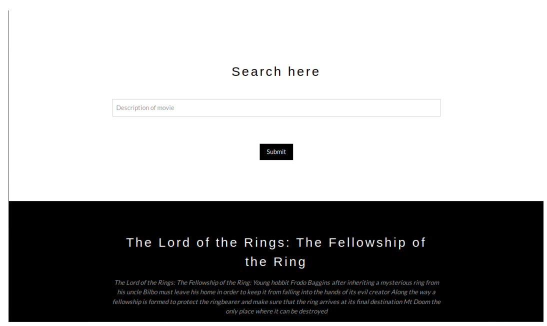

Figure 2: Interface

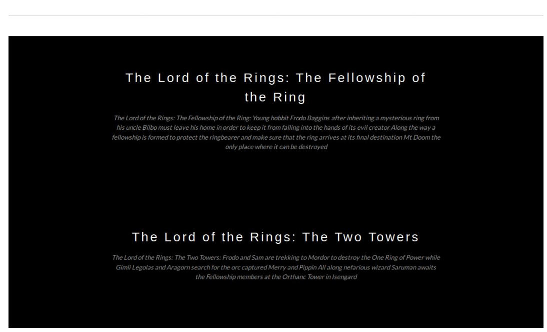

Figure 3: Interface slide down

3 Results and evaluation

We formulate 35 short descriptions of movies and use our search engine to fetch the result. Some of the queries come from us and some of them are from our user during the testing phase of our development. For each description, we check if the intended result is returned by the search engine.

Then we check the rank R of the expected movie returned by the search engine. We define $\frac{1}{R}$ as the precision of our search engine. We calculate the harmonic mean of R (if the expected movie isn't retrieved within the first 10 movies, R is infinity.) and use it as the evaluation for the search engine.

This is part of the testing result from our search engine, the third coloum is the rank of the movie title retrieved based on BM25 BM25 and the fourth coloum is for BM25+.

Query	Answer	$\frac{R}{\text{BM25}}$	$\frac{R}{\text{BM25+}}$
Jack and rose	Titanic	2	1
Dinosaur	Jurassic World	4	1
France revolution	Les Misérables	4	2
Dream thief	Inception	6	1
Boy magic school	Harry Potter	9	1
Cowboy astronaut	Toy story	5	1
Robot kill human	Terminator	4	1

Figure 4: Part of Our Queries Test Result

The harmonic mean of rank of the expected movies can evaluate the average results the user need to skim through to find the movie they intend to have. Thus, the lower harmonic mean of rank, the better engine we have. The following is the harmonic mean of rank result of BM25 and BM25+.

	BM25	BM25+
$\frac{n}{\sum_{i=1}^{i=n} \frac{1}{R_i}}$	2.944	1.449

Figure 5: Comparison of Harmonic Mean of Rank

The following is the figure showing the sorted distribution of expected movie retrieved ranks. For the figure for BM25, there are a lot of queries that returns failure (greater than 10), while for BM25+, there is no failure and the mean of ranks is obviously lower.

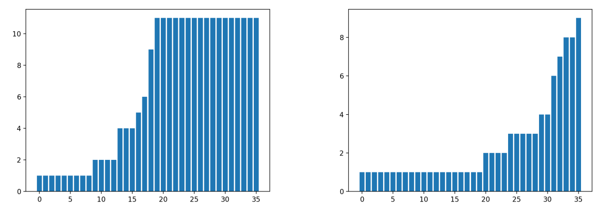

Figure 6: Expected Movie Retreived Ranks Distribution

Thus we can evaluate our search engine as useful, for on average the user can expect to find the moive they want from the first and second result retrieved.

4 Conclusions

The result of this project is generally satisfying with our experiments with it. We succeeded in returning the related results with the queries and has improved its performance with data processing and parameter tuning. Thus, with our search engine, people can mostly get the movie they referred to. However, the search engine might not work well with subjective feelings on movies since the data we used for query are neutral descriptions.

5 Future Improvements

During data processing, we have tried part of speech tagging on the sentence and give more weights to certain types. However, it doesn't turn out well as we expected. We believe that digging further into this by analyzing the parse tree to evaluate the part of speeches more precisely can help with giving better weights. Moreover, we believe that keeping a search and view log can also help with improving accuracy. Within a series of similar searches, we can analyze the movies the user has clicked into and find the similarities to feed back into the algorithm and adjust the weights accordingly.

6 Acknowledgement

Prof. Qiaozhu Mei, Mr. Wei Ai from University of Michigan; Teaching Assistant [Shiyan Yan, Xinyan Zhao] for SI 650 at University of Michigan.

References