Ranked Retrieval-Based Search Engine Along With Implicit Relevance feedback

Surya Teja A,Pawan Kalyan D,Krishna Karthik G
Computer Science Engineering
Indian Institute of Information Technology
Sri city, Chittoor, India

***Abstract* – Ranking of query results is one of the fundamental problems in information retrieval (IR), the scientific/engineering discipline behind search engines. The majority of search engines today provide users with results in a ranked order. This order reflects the underlying algorithm's best guess at what the users' needs might be and the documents that are most likely to be relevant to the needs. If there is a document that the algorithm thinks, based on a variety of factors, might be the answer, it has to come up on position 1, followed by another document with the next highest likelihood of being relevant, and so on.**

Keywords – Term-Frequency, Document-Frequency, Inverse DocumentFrequency,ClickStreamMining,stemming,lemmatization, implicit relevance feedback.

I. INTRODUCTION

The motto of this project is to implement a rank based retrieval system using term weights whose rank is based on the tf-idf score and updating relevance based on number of clicks(click stream mining).

To begin with, each document undergoes processes like tokenization, stemming and stop word removal. The term frequency and inverse document frequency are calculated for each term. Then the term weights are calculated for
each term using the tf-idf weighting method. Now the score between each document and the given query is calculated by summing tf-idf weights of terms in the query. The document with the highest score will get highest priority. And the priority is not static which will increase with the number of clicks on that particular document (Implicit Relevance).

II. MOTIVATION

Ranking helps to make things easy and Click stream mining(Implicit Relevance Feedback) helps in improving the chance of relevance of a document that user expect. Whenever a user types a query, he/she might expect the most important and relevant documents. If the search system is not based on rank then it may be difficult for the user to search among the retrieved documents. And if the system is build based on tf-idf value, the tf–idf value increases proportionally to the number of times a word appears in the document. So, we use tf-idf weights which can eradicate the problem faced by the other method. Sometimes, the top ranked document may not be the document which user expects. So, on using click stream mining which will add relevance based on number of times the particular document has been visited.

Through this project we have highlighted the importance of having clear motivation behind each factor and clarity on the role it plays in achieving the ranking that retrieves the documents which the users expect.

III. SYSTEM ARCHITECTURE

Figure 1 shows the architecture of our platform. Our platform is composed of two different retrieval methods, one is based on Tf-Idf weights and other on Cosine Similarity. Tf-Idf based system will present the user with documents based on tf-idf weighting method and the Cosine similarity based ranking system will present the user with the documents based on cosine similarities between query and documents.

Search allows users to enter queries and retrieve lists of documents. And the select button is to select the method of retrieval. The platform server, as shown in the Figure 1, is responsible for displaying the document assessment interfaces to the user, configuring the system, and communicating requests and responses to and from each component.

The types of user interactions that the server allows are:

- Search for specific documents using Search;
- Assess documents retrieved by either tf-idf scores or Cosine similarity

All components in the architecture are stand-alone and interact with each other via HTTP API. For example, any search engine can be added to our system with minimal effort. The platform server is built using Django, a Python web frame-work. Tf-idf rank and Cosine similarity based retrieval systems are written in Python3.

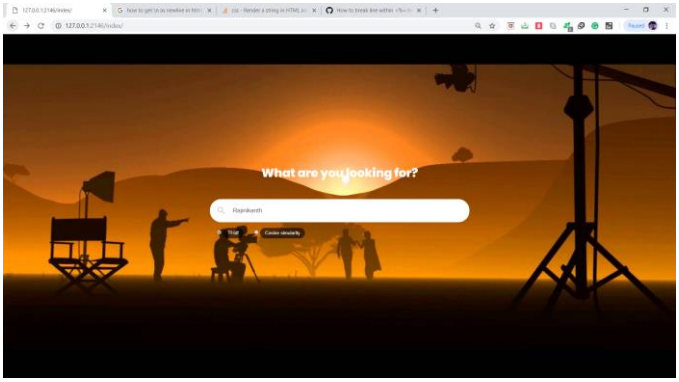


Figure-1

Figure-1 shows the user interface where the user need to enter the query and select the method in which the user want to retrieve.

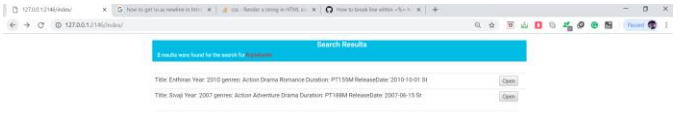


Figure-2

Figure-2 shows the UI which consists all the retrieved documents.

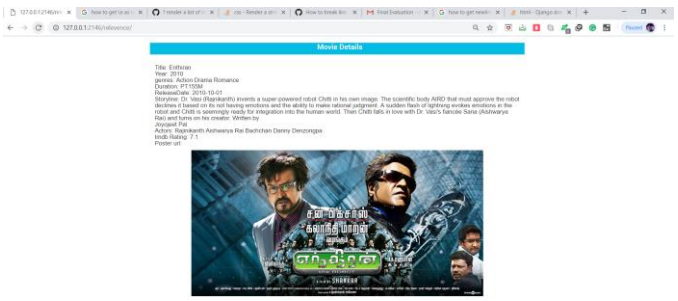


Figure-3

Figure-3 shows the UI where you can see the details of that particular document which is opened.

IV. PLATFORM COMPONENTS & FEATURES

A. Dataset

We used the dataset which contains the top rated IMDB movies. It contains 500 movie documents which have title, storyline, actors, year of release, etc.,

B. TF-IDF .

The tf–idf is the product of two statistics, term frequency weights and inverse document frequency. In the case of the term frequency Tf(t,d), the simplest choice is to use the raw count of a term in a document, i.e., the number of times that term t occurs in document d.If we denote the raw count by ft,d, then the simplest tf scheme is Tf(t,d) = ft,d. But using the term frequency, the tf–idf value increases proportionally to the number of times a word appears in the document. So, we use term frequency weights which would be calculated using the following mathematical expression:

$$\mathbf{tf(t,d) = (1 + log(Tf(t,d)))}$$

The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

With

N: total number of documents in the corpus
|{1+ (d € D : t € d) }|: number of documents where the term t appears. If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to **|{1+ (d € D : t € d) }|.**

Then tf–idf is calculated as

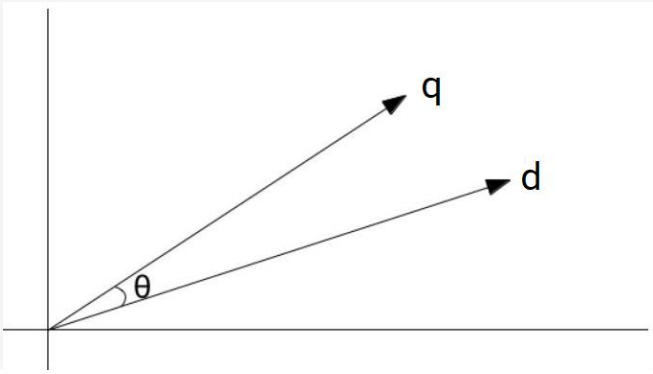
$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

C. Cosine Similarity

In Cosine Similarity from each document we derive a vector. The set of documents in a collection then is viewed as a set of vectors in a vector space and the query is also a vector. Each

term will have its own axis. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any angle in the interval (0,π] radians. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90° relative to each other have a similarity of 0. Using the formula given below we can find out the similarity between any two documents.

```
Cosine Similarity(d,q)=Dot product(d,q)/||d||*||q||
```



Where d is the document and q is the query. So using cosine similarity we can find similarity between documents and query. If the cosine similarity value is near to 1 it implies that the document is closely related to the given query.

Even though the cosine similarity has been prevalent in the majority of web clustering approaches, it may fail to explicitly seek profiles that achieve high coverage and high precision simultaneously and also the cosine similarity looks at “directional similarity” rather than magnitude differences. Suppose if there are two documents d1,d2 each containing all the query terms but document d2 has more no of repetitions of the query terms, in this case a perfect system should return document d2 since it has more no of repetitions of the query term but the system using cosine similarity gives same score to both the documents d1 and d2.We integrated the cosine similarity system with implicit relevance feedback allowing the most popular documents to come first in the search results which makes the search system a bit efficient than the traditional cosine similarity system.

V. CONCLUSION

In this paper, we described the design of an information retrieval system. We tokenized, lemmatized and normalized the given documents and then we used term frequency and inverse document frequency to calculate the TF-IDF score for a given query and further improved the ranking system by using the implicit relevance feedback from the users. We also developed the ranking system using cosine similarity and compared it with the TF-IDF scoring. The system allows for both Implicit relevance feedback and search, and these components can be easily replaced with different implementations

References

[1]. Cosine similarity by Venkatesh Vinayakarao.

[2].Tf-Idf scoring by Venkatesh Vinayakarao.

[3]. Tf-Idf ranking wikipedia

[4]. Cosine similarity wikipedia

[5]. Click stream mining for nlp