INFORMATION RETRIEVAL

PROJECT REPORT

ENTERTAINMENT

Explanation : Basically,this project is mainly about searching the related information and retrieving relevant documents based on the user given query.this project is related to entertainment news based search.Most of the data crawled from some entertainment news based websites.here,documents are retrieved based on the tf-idf scores of each document.

Task 1 :

<u>INDEXING :</u>

Website scraping

Index construction of all the terms in title and contents of all the documents ..

I scraped a total of 49975 documents which consists of some sort of entertainment news in each document. This entire dataset is stored as .csv named as entertainment.csv.

★ For all the terms which we are extracted are preprocessed using different preprocessing steps like tokenization,case folding,stemming ..

★ I have used trie data structure to store the inverted index because retrieval of documents is faster.

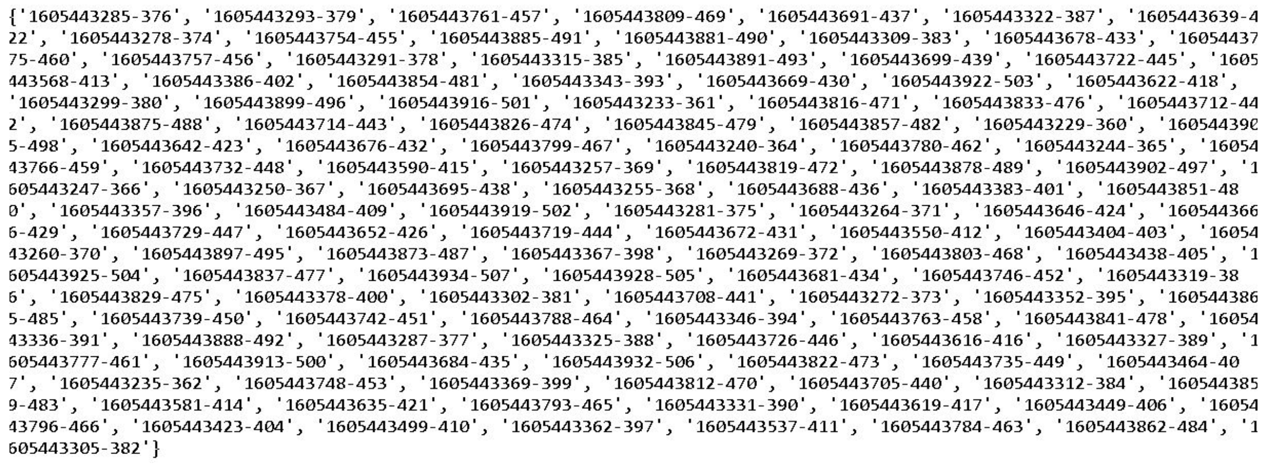
★ We will follow the ranked retrieval model for retrieving relevant documents.

Task 2 : Relevant Based Search .

<u>SEARCHING :</u>

Ranked retrieval for retrieving the relevant documents :

- ★ Firstly,the query undergoes some preprocessing techniques like tokenization,case folding,stemming.
- ★ we will calculate the scores and stores the dot product of the tf-idf weights of(q,d) score vectors of the query and document pairs .here idf is inverse document frequency calculated by using the following formula idf = log(N/dft),where N is the total number of documents in the dataset we collected
- ★ We arrange wordsInDocument is a sorted list of (word, score) tuples where score is the tf-idf score for the (word, <ith document>) pair.
- ★ We will get the terms present in the query and retrieve all the documents which contain the query terms.for the retrieved documents we calculate tfidf_document = 1 + math.log10(tf_document).
- ★ for all documents we retrieved there is document id and we will calculate the score of doc_id using scores[docID] = scores[docID]/ math.sqrt(documentLength[docID]).
- ★ DocId is as shown below:



```
{'1605443285-376', '1605443293-379', '1605443761-457', '1605443809-469', '1605443691-437', '1605443322-387', '1605443639-4
22', '1605443278-374', '1605443754-455', '1605443885-491', '1605443881-490', '1605443309-383', '1605443678-433', '16054437
75-460', '1605443757-456', '1605443291-378', '1605443315-385', '1605443891-493', '1605443699-439', '1605443722-445', '1605
443568-413', '1605443386-402', '1605443854-481', '1605443343-393', '1605443669-430', '1605443922-503', '1605443622-418',
'1605443299-380', '1605443899-496', '1605443916-501', '1605443233-361', '1605443816-471', '1605443833-476', '1605443712-44
2', '1605443875-488', '1605443714-443', '1605443826-474', '1605443845-479', '1605443857-482', '1605443229-360', '160544390
5-498', '1605443642-423', '1605443676-432', '1605443799-467', '1605443240-364', '1605443780-462', '1605443244-365', '16054
43766-459', '1605443732-448', '1605443590-415', '1605443257-369', '1605443819-472', '1605443878-489', '1605443902-497', '1
605443247-366', '1605443250-367', '1605443695-438', '1605443255-368', '1605443688-436', '1605443383-401', '1605443851-48
0', '1605443357-396', '1605443484-409', '1605443919-502', '1605443281-375', '1605443264-371', '1605443646-424', '160544366
6-429', '1605443729-447', '1605443652-426', '1605443719-444', '1605443672-431', '1605443550-412', '1605443404-403', '16054
43260-370', '1605443897-495', '1605443873-487', '1605443367-398', '1605443269-372', '1605443803-468', '1605443438-405', '1
605443925-504', '1605443837-477', '1605443934-507', '1605443928-505', '1605443681-434', '1605443746-452', '1605443319-38
6', '1605443829-475', '1605443378-400', '1605443302-381', '1605443708-441', '1605443272-373', '1605443352-395', '160544386
5-485', '1605443739-450', '1605443742-451', '1605443788-464', '1605443346-394', '1605443763-458', '1605443841-478', '16054
43336-391', '1605443888-492', '1605443287-377', '1605443325-388', '1605443726-446', '1605443616-416', '1605443327-389', '1
605443777-461', '1605443913-500', '1605443684-435', '1605443932-506', '1605443822-473', '1605443735-449', '1605443464-40
7', '1605443235-362', '1605443748-453', '1605443369-399', '1605443812-470', '1605443705-440', '1605443312-384', '160544385
9-483', '1605443581-414', '1605443635-421', '1605443793-465', '1605443331-390', '1605443619-417', '1605443449-406', '16054
43796-466', '1605443423-404', '1605443499-410', '1605443362-397', '1605443537-411', '1605443784-463', '1605443862-484', '1
605443305-382'}
```

★ Dictionary will store the score of each document with respect to the query.

Like for example:

doc ID = 1605443742-451

tf-idf score= 0.0013485645557762816

And some content related to the query in that document..

doc ID = 1605443383-401

tf-idf score= 0.0012923743659522698

doc ID = 1607869432-15682

tf-idf score= 39.31373777802908

doc ID = 1607874717-35007

tf-idf score= 39.31373777802908

We will get all scores of each document

Now by using score it will return relevant documents related to the query given by the user.

PUPPALA SUDHEER KUMAR
S20180010140,UG3
INFORMATION RETRIEVAL.