GSM - Engine

Sathwik Ch and Harisaipravin Sv
AMCS Department
PSG College of Technology

Abstract

The Search Engine project will be able to provide users required information at one particular place by using the words and patterns entered by the user during their search operation.

1. Introduction

This paper gives an overview of Mobile Phone Information Retrieval systems for content-based mobile searching, preceded by a brief overview of the methods commonly used by these systems.

In this paper, we examine the results of applying Term Frequency Inverse Document Frequency(TF-IDF) to determine what words in a corpus of documents might be more favorable to use in a query. As the term implies, TF-IDF calculates values for each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of documents the word appears in. Words with high TF-IDF numbers imply a strong relationship with the document they appear in, suggesting that if that word were to appear in a query, the document could be of interest to the user.

The other goal of this study is to enhance the Okapi BM25 document retrieval model. In doing so, this research hypothesizes that the structure of text inside documents and queries hold valuable semantic information that can be incorporated into the Okapi BM25 model to increase its performance.

2. Background

GSMArena is an online website that offers comprehensive and up-to-date mobile phone information.
● The website offers information about various mobile brands, including Nokia, Samsung, Motorola, Sony, LG, and more.
● It enables its users to get to know about latest and real-time mobile phone trends and other related information.
● GSMArena blog offers
● The previous Search Engine was not able to work under real time situations and able to fetch information which is only saved under their database.

● So there was no mechanism for building the index file and keeping information as per the user search query.
● Most of the time, users were not able to get their desired information and this section kept users at their limit.

3. Results

The proposed system would make use of information retrieval mechanisms to give proper sorted and a well structured search result to the user based upon their search keyword. With this system in place, users should be able to search with the camera specs, hardware, internal specs etc. The UI developed using NodeJs for backend and React for frontend allowed the user to use the search option by providing info to the retrieval models.

The comparison between Okapi and tf-idf yielded mixed results where Kendall tau coefficient showed Okapi performs slightly better than tf-idf.

4. Format

Data fetched from scraping from the gsm website was stored as comma separated values for easier access. For documents we used a “bag of words” which is a representation of text that describes the occurrence of words within a document. Pre-processed data was stored in files for faster retrieval and to improve the speed of our application. Hash tables were used to store the tf-idf results which were then saved to the datafile for faster access during successive iterations. This helped in boosting the search query time as the entire database was not queried.

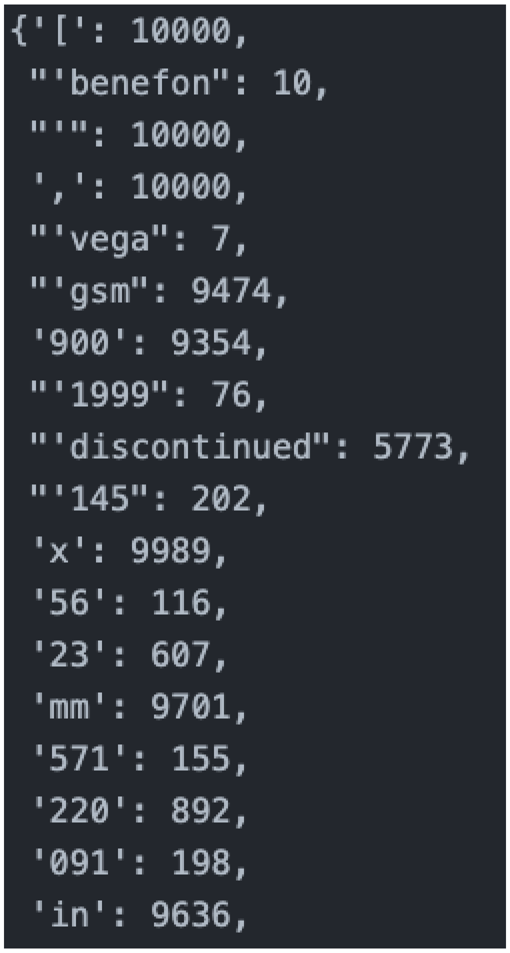
4.1 Choice of Word Processor

The data crawled from the website consisted of html tags and other noises which were taken care of during the data preprocessing process. The Photos and the Yes/No answers extracted from the website had to be linked to the proper column to avoid confusions, but since tf/idf just calculates the output considering all as a bag of words, it made sense to remove the yes/no options from the dataset. Finally the stop words were removed and the processed words were lemmatized. The few false positives found were checked manually and the errors were removed with utmost care with some help from the domain expert.

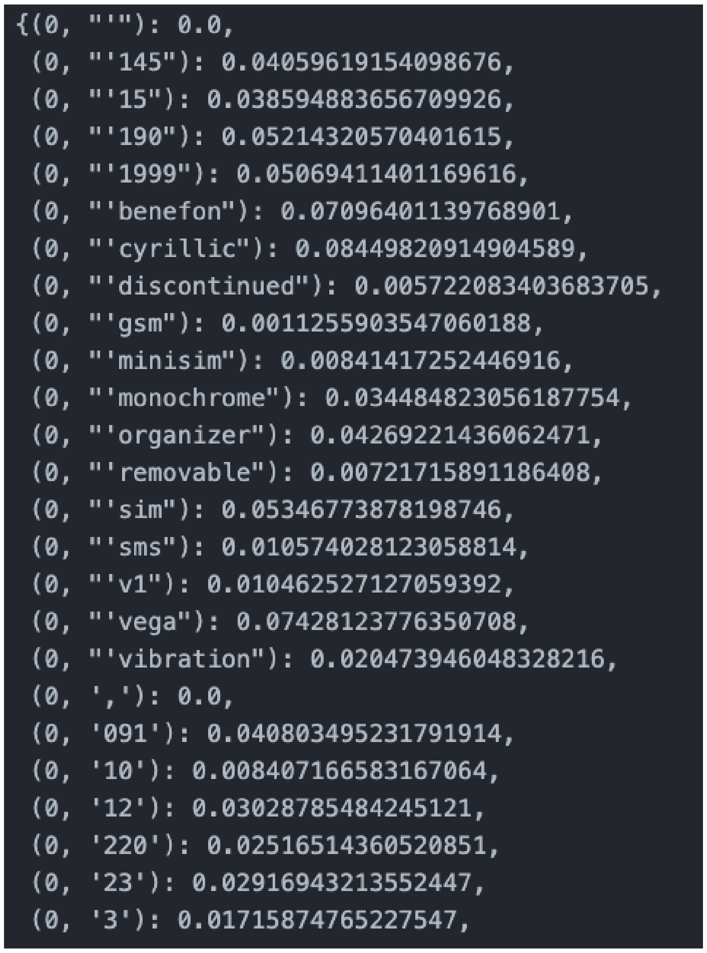
4.2 Layout

A simple frontend UI was made using React with python flask supporting the data flow between python and ReactJS. The data were stored as csv files to be extracted later.

4.3 Data Presentation



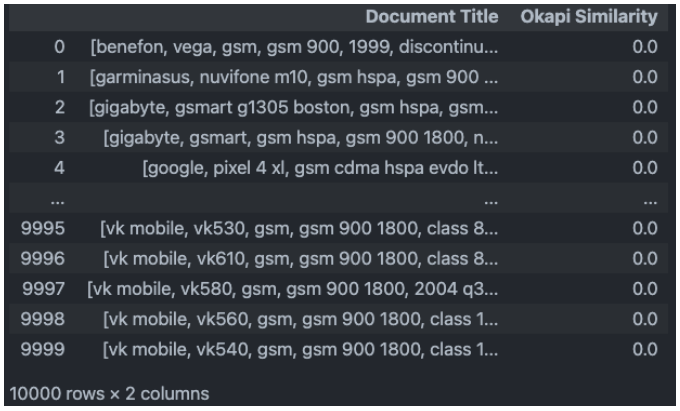
```
{'[': 10000,
 "'benefon": 10,
 "'": 10000,
 ',': 10000,
 "'vega": 7,
 "'gsm": 9474,
 '900': 9354,
 "'1999": 76,
 "'discontinued": 5773,
 "'145": 202,
 'x': 9989,
 '56': 116,
 '23': 607,
 'mm': 9701,
 '571': 155,
 '220': 892,
 '091': 198,
 'in': 9636,
```



```
{(0, "'"): 0.0,
 (0, "'145"): 0.04059619154098676,
 (0, "'15"): 0.038594883656709926,
 (0, "'190"): 0.05214320570401615,
 (0, "'1999"): 0.05069411401169616,
 (0, "'benefon"): 0.07096401139768901,
 (0, "'cyrillic"): 0.08449820914904589,
 (0, "'discontinued"): 0.005722083403683705,
 (0, "'gsm"): 0.0011255903547060188,
 (0, "'minisim"): 0.00841417252446916,
 (0, "'monochrome"): 0.034484823056187754,
 (0, "'organizer"): 0.04269221436062471,
 (0, "'removable"): 0.00721715891186408,
 (0, "'sim"): 0.05346773878198746,
 (0, "'sms"): 0.010574028123058814,
 (0, "'v1"): 0.010462527127059392,
 (0, "'vega"): 0.07428123776350708,
 (0, "'vibration"): 0.020473946048328216,
 (0, ','): 0.0,
 (0, '091'): 0.040803495231791914,
 (0, '10'): 0.008407166583167064,
 (0, '12'): 0.03028785484245121,
 (0, '220'): 0.02516514360520851,
 (0, '23'): 0.02916943213552447,
 (0, '3'): 0.01715874765227547,
```

The tf/idf representation shown is the figure above. The left image shows the tf count of the keywords in the document. Whereas the right image shows the idf value of the word in the document, thus giving us an idea of how important the word is in the document. Greater the idf value, greater is its importance. tf/idf states that the term occurring frequently within a document but rarely in the rest of the collection is of high importance.

Okapi is designed to be sensitive to term frequency and document length while not adding too many parameters. It differs from vector space only by the term weights.Okapi has the ability to adapt itself to user clicks and evolve over time with relevance feedback. It ranks a set of documents based on the query terms appearing in each document, regardless of their proximity within the document.



	Document Title	Okapi Similarity
0	[benefon, vega, gsm, gsm 900, 1999, discontinu...	0.0
1	[garminasus, nuvifone m10, gsm hspa, gsm 900 ...	0.0
2	[gigabyte, gsmart g1305 boston, gsm hspa, gsm...	0.0
3	[gigabyte, gsmart, gsm hspa, gsm 900 1800, n...	0.0
4	[google, pixel 4 xl, gsm cdma hspa evdo lt...	0.0
...	...	...
9995	[vk mobile, vk530, gsm, gsm 900 1800, class 8...	0.0
9996	[vk mobile, vk610, gsm, gsm 900 1800, class 8...	0.0
9997	[vk mobile, vk580, gsm, gsm 900 1800, 2004 q3...	0.0
9998	[vk mobile, vk560, gsm, gsm 900 1800, class 1...	0.0
9999	[vk mobile, vk540, gsm, gsm 900 1800, class 1...	0.0

10000 rows × 2 columns

4.4 Algorithms

Algorithms used here include tf/idf and Okapi model. The code for TF-IDF is elegant in its simplicity. Given a query q composed of a set of words wi, we calculate wi, d for each wi for every document d ϵ D. In the simplest way, this can be done by running through the document collection and keeping a running sum of fw, d and fw, D. Once done, we can easily calculate wi d according to the mathematical framework presented before. Once all wi, dís are found, we return a set D* containing documents d such that we maximize the following equation:

$$\Sigma i \ wi, d$$

Either the user or the system can arbitrarily determine the size of D* prior to initiating the query. Also, documents are returned in a decreasing order according to the equation.
This is the traditional method of implementing TF-IDF.

okapi ranks a set of documents based on the query terms appearing in each document, regardless of their proximity within the document. It is a family of scoring functions with slightly different components and parameters. One of the most prominent instantiations of the function is as follows.

Given a query Q, containing keywords {\displaystyle q_{1},...,q_{n}}q_1, ..., q_n, the BM25 score of a document D is:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

Addition Algorithm - todo

```
Input:    query
Output:   ranked documents
Method:   tf-idf and Okapi

input(crawled document);

query = 'samsung';        /* query is a string */

Document Preprocessing with stop words and lemmatization

Word tokenization using inbuilt libraries

for i in document
      update pooling table   /* tf-idf modelling*/
```

```
end;

Ranking docs using the tf-idf calculated values

Calculate Similarity measures for effective retrieval of
the search query results

for i in document
      okapi estimation  /* probability modelling*/
end;

/* At this point, both okapi and tf-idf is modelled */

Compare how effectively Algorithm works using the spearman
coefficient.

print(top k relevant documents);

end;

function search(integer k, string query) return list

     output = []
     for i in outputlist:
          output.append(outputlist[i])
     return output;

end search;
```

Algorithm 1. Computes n * n

5. Conclusions

We have seen that TF-IDF is an efficient and simple algorithm for matching words in a query to documents that are relevant to that query. From the data collected, we see that TF-IDF returns documents that are highly relevant to a particular query. If a user were to input a query for a particular topic, TF-IDF can find documents that contain relevant information on the query. Furthermore, encoding TF-IDF is straightforward, making it ideal for forming the basis for more complicated algorithms and query retrieval systems (Berger et al,2000).

Despite its strength, TF-IDF has its limitations. For large document collections, this could present an escalating problem.

References

1. GSMArena Website (https://www.gsmarena.com/)

2.Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents (https://www.researchgate.net/publication/326425709_Text_Mining_Use_of_TF-IDF_to_Examine_the_Relevance_of_Words_to_Documents)

3.OkapiBM25,(https://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html)

Appendix

Raw data: There are 108 unique phone brands with 39 variables: network*technology , 2G*bands , 3G*bands , 4G*bands, network*speed, GPRS, EDGE ,announced ,status ,dimentions ,weight*g ,weight*oz ,SIM ,display*type ,display*resolution ,display*size ,OS ,CPU ,Chipset , GPU ,memory*card ,internal*memory ,RAM ,primary*camera ,secondary*camera ,loud*speaker ,audio*jack ,WLAN ,bluetooth ,GPS ,NFC ,radio ,USB ,sensors ,battery ,colors ,approx*price*EUR ,img_url