Search Engine Based On Spark

Rabin Banjade**
rbnjade1@memphis.edu
University of memphis

Md Azim Ullah†
mullah@memphis.edu
University of Memphis

ABSTRACT

The major purpose of information retrieval(IR) systems is to find out relevant documents from pool of large collection of documents that resemble a query.The process of finding relevant documents possess different challenges given the huge collection of documents. In this paper we explore the use of distributed computing framework spark to solve the challenge of efficient query and document processing to speed up information retrieval process.

CCS CONCEPTS

• **Information Retrieval** → **Search**; *Vector Space Model*; • **Distributed System** → Spark.

KEYWORDS

Spark, Information Retrieval, Vector Space Model, Search Engine

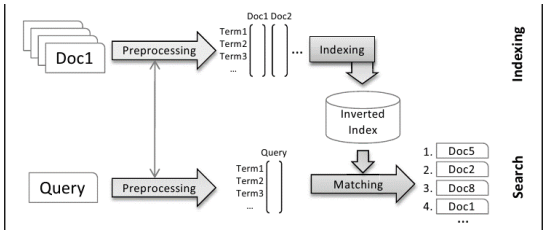

Figure 1: Information retrieval system

1 INTRODUCTION

Information retrieval system begins when user enters search query into the system.A basic overview of an information retrieval system in shown in fig 1.Given a query, relevant document is fetched by the system.For this, each information retrieval system consists of document processing, query processing, indexing, ranking etc. Given use of huge amount of documents, time spent for each of the processes increases to a huge extent.Time spent for a system to return relevant documents to the user depends on several factors like model used, ranking mechanisms and most importantly computing capacity.Major portion of time is spent on indexing and pre-processing of large set of documents.In this project we use spark to speed up pre-processing and indexing of documents.We analyze speed gain using spark for information system we build.We have used vector space model [1] which is an algebric document representation model.Using spark we carry out necessary preprocessing steps like stop words removal,stemming of each word and building an inverted index of documents.We also use spark for query processing and relevancy ranking.

*Both authors contributed equally to this project.

1.1 Vector Space Model

Vector space model[1] is popularly used to find relevant documents for a given query.Each document is represented as a N-dimensional vector where N is the number of distinct terms in the documents or queries.Each i-th index in the vector represents the score of the i-th term of the document. For scoring term-frequency and Inverse-Document-Frequency(idf) is used.

Term-frequency: It represents the number of terms contained by a document.The Term-Fequency (tf_{ij}) is computed with respect to the i-th term and j-th document :

$$\mathrm{t}f_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Inverse document frequency: Inverse document frequency gives a measure of how important a word is, how much information is provided by the word i.e. if a word is common in all the documents than it provides least information about relevancy of the document to the query.It gives measure of how rare a word is across all the documents. It is logarithmically scaled inverse fraction of the documents that contain the word.

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

with

N: total number of documents in the corpus

$|\{d \in D : t \in d\}|$: number of documents where the term t appears.

For each of the vectors each i-th index representing the index is $tf * idf$.

Cosine similarity: For calculation of relevancy between two documents or documents and query we use cosine similarity.

$$\cos\theta = \frac{\mathbf{d_2} \cdot \mathbf{q}}{\|\mathbf{d_2}\| \, \|\mathbf{q}\|}$$

For relevancy measure we calculate the cosine of the angle described by the vectors. If the angle is small relevance is high and if the angle is large, relevance is low.

Spark: Spark [2] is a distributed computing framework.It is basically a data processing engine that also contains libraries for SQL, machine learning, graph computation and stream processing.Apache Spark uses resilient distributed dataset(RDD). Spark facilitates implementations of iterative algorithms very quickly than map-reduce frameworks.

2 SYSTEM DESIGN

2.1 System Configuration

Our system is implemented using Python 3.6, Spark 2.4.0 The environment is configure to use one executor with 16 cores, and 30GB memory. For GUI we have used flask 1.0.2.

2.2 Dataset Description

We scraped more than 12,000 documents crawling from university of memphis homepage.These documents only contain text and numeric characters.

3 IMPLEMENTATION DETAILS

3.1 Data Collection

For data collection, we used Breadth First Search(BFS) to parse webpages that were linked to University of memphis homepage. We parsed these web pages into text files

3.2 Data pre-processing

For data preprocessing was done in two steps mainly:

- Stop words removal: We removed stop words from the text documents. Stop words are such words that commonly occur in english language like:'the','of','he','she' etc.
- Stemming: For finding root words for each of the words we used Porter Stemmer. For example root word for quickly would be quick, eaten would be eat etc.

For each of the steps we used spark to speed up the process.

3.3 TfIdf Computation

We use hashing trick to determine the index of each word in our vocabulary. Apache Spark provides a function to hash the texts. After computing the term frequency We use the tfidf model to compute the tfidf sparse vector for each document. The L2 norm of each sparse vector is also computed to reduce the query time.

3.4 Query Algorithm:

(1) For each query tokenize, remove stop words, stem each word.
(2) Find the vectorized representation of the query through murmur3 Hashing transform with the predefined vocabulary size.
(3) Broadcast the query vector and L2 norm of the query vector to each row of the tfidf matrix.
(4) Compute the dot product between query and each document vector and normalize with the precomputed document L2 norm and query norm to obtain the cosine simillarities.
(5) Sort the cosine similarities to get the top 5 highest values and corresponding indices.
(6) Show the results.

4 RESULTS AND ANALYSIS

A snapshot of our search engine is shown in fig 2

4.1 Time analysis:

Preparation of $tf - idf$ vector was the major time consuming part of our system.For each of the query.

total Memory : 64 GB total number of documents : 12000 Time taken varying number of cores for preprocessing step is shown in 1

For each query average time taken : 1.05 secs time taken to return search result doesn't depend upon the length of the query

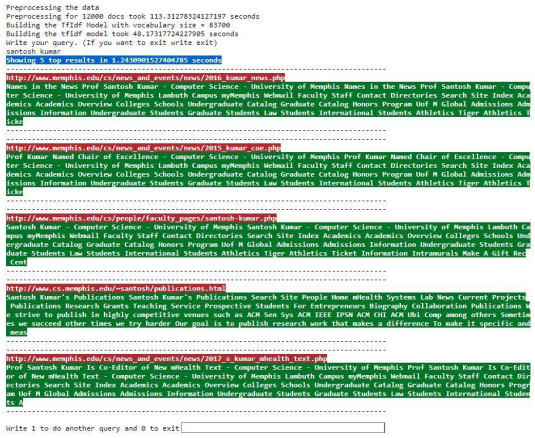

Figure 2: Information retrieval system

N	time (secs)
sequential	14400
4	50
6	48
8	42

Table 1: Time taken varying cores

4.2 Precision-Recall

For measuring the performance of our search engine we checked with few queries to see how our search engine performs.We have posted analysis for one of the queries 'International Office' for our search engine,expressed in table 2 below.

For performance of the search engine, precision and recall

N	Doc_id	Relevant	Recall	Precision
1	1318	yes	0.090909	1
4	5996	yes	0.181818	0.5
7	6496	yes	0.272727	0.428571
8	5995	yes	0.363636	0.5
10	6007	yes	0.454545	0.5
11	4767	yes	0.545455	0.545455
12	2907	yes	0.636364	0.583333
13	353	yes	0.727273	0.615385
19	1320	yes	0.818182	0.473684
24	4871	yes	0.909091	0.416667
81	6010	yes	1	0.135802

Table 2: Precision and recall

5 CONCLUSION AND FUTURE WORK

Using vector space model is simple model to implement.However, it assumes that terms are statistically independent. For future improvements can be made on the search engine algorithm itself, for example incorporating semantic notion to the relevancy. Also, scaling the project with large corpus of documents can be another improvement.

ACKNOWLEDGMENTS

To wikipedia for imparting knowledge to mankind.

REFERENCES

[1] Salton, G. and Wong, A. and Yang, C. S.,A Vector Space Model for Automatic Indexing, Commun. ACM,Nov. 1975, volume 18, number 11,nov,publisher,ACM,New York, NY, USA,

[2] Zhang, Qi and Cheng, Lu and Boutaba, Raouf", Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications,2010, May,01,vol 1,pages 7–18,issn 1869-0238,doi 10.1007/s13174-010-0007-6, url "https://doi.org/10.1007/s13174-010-0007-6"