Text Based Information Retrieval Search Engine

Prakrati Gupta
IIIT Delhi
MT20014
prakrati20014@iiitd.ac.in

Vandana
IIIT Delhi
MT20106
vandana20106@iiitd.ac.in

Nidhi Allwani
IIIT Delhi
MT20034
nidhi20034@iiitd.ac.in

Chetan Sharaf
IIIT Delhi
MT20109
chetan20109@iiitd.ac.in

Avaneesh Kumar Patel
IIIT Delhi
MT20005
avaneesh20005@iiitd.ac.in

ABSTRACT

This document is regarding the project presentation on Text based search engine. In this we are analysing the challenge faced in retrieval of document, Methods used to make document retrieval more efficient and has create a hybrid model which has been implemented using TF-IDF technique.Here we have implemented a search engine on the wikipedia dump of size 81.8 GB. which can search query generally and field specific both.We are analysing our model by running a sample dataset(100MB) for baseline model and compare the time taken to create a index and to search a query.

1 INTRODUCTION

According to the Pew Internet American Life Project, Web search continues its explosive growth there are over 107 million Web-search users in the United. States alone, in June 2004, and they did over 3.9 billion queries. A study at the beginning of 2005 says indexable web size is at least 11.5 billion pages. Thus search algorithmic efficiency is as important as ever: although processor speeds are increasing and hardware is getting less expensive every day, the size of the corpus and the number of searches is growing at an even faster pace.

Since 2016, IBM Marketing Cloud study 90 % of the data on the internet has been created. All People, businesses, and devices have become data factories that are pumping out incredible amounts of information to the web each day.

Over 3.5 Billion Google researches are attended worldwide every breath of each day. That is 2 trillion explorations per year worldwide. That is above 40,000 search questions per second With an increase in data available on the internet, getting difficult to retrieve a result from such large data that is becoming a challenge.

Here we will discuss Some features used to make retrieval of data more efficient from Big data.

Much of research has already be done on the topic considering that the area has already been explored long back by not only the researchers but also big tech giants like google which itself was not the first economical text based search engine, this paper tries a new approach. This model is a hybrid model by the combination of various techniques (Such as tokenization, stopwords removal, stemming, inverted indexing) to overcome the limitations and create an optimized search engine.

2 LITERATURE REVIEW

Search Engine Optimization is very wide topic and has been approach in several ways :

Sampling search-engine results[1]. In this sampling is done on small part of data instead of full data. Analysis of tf-idf model and its variant for document retrieval authors worked with numerical as well as textual data and tried to find out which word, phrase or set of numbers can be considered more important relating to a particular query using different variation of vectorization techniques focusing more on TF-IDF. Improving precision in information retrieval for swedish using stemming. Stemming make the words in its root form,defines how stemming improves both precision and recall for the swedish text. Inverted index compression and query processing with optimized document Ordering discussed how using previously optimised ordering, they developed a method in the compression technique that gives a higher throughput than most of the algorithms already present elsewhere.

Information retrieval on the world wide web[7] discussed navigation on web, how documents on web is represented, and models for retrieving information.

Intelligent Search Engine algorithms on indexing and searching of text documents using text representation[10] describes and compares various indexing and searching technique with their complexities and which algorithm can be used for increasing indexing and searching speed in various circumstances.

The authors of the paper,Document clustering: TF-IDF approach[2] provide a new overview of clustering the document where the similar documents can be clustered together, which makes the process of retrieving data much faster and efficient.

CiteSeerX: AI in a Digital Library Search Engine,[13] article mentions how various machine learning and artificial intelligence techniques can be used for fast information retrieval, by using metadata , tables , subsections as features in the training model.

The authors in the paper,[6] describe a basic procedure to use nificantly boost the page ranking process. The paper uses distributed format to compute the page rank.

3 CHALLENGES

It discusses the unique problems at high level that appeared to search engines[8]

- **Storage in Computing power**: As many indices need to be stored, collecting a lot of data from the web via crawling will require a lot of storage.

Resolved: We can resolve the storage by using big storage like cloud and computation power can be resolved by referring for computer which has high performance

- **Spam** Search Engine optimization and various techniques can be used for page rank manipulation, which might not give the best results.
  Resolved: Spam can be resolved by using different ranking algorithm
- **Quality Evaluation**:Determining the status of various ranking algorithms is a specially challenging query. Popular search engines benefit from massive quantities of user-behavior data they can employ to help estimate ranking. Users normally will not attempt to give exact feedback but give implicit feedback, such as the outcomes they agreed.
  Resolved: Joachims [9] suggested experimental technique which merge the result of two ranking algorithm in a single set of result. So comparisons between two algorithm is done in this way.Joachims uses number of clicks as quality metric and show under which condition which algorithm retrieve more relevant links than others.
- **Content Quality**: Quality of the content depends on the person; different people have a different perspective when searching for a query. E.g., "I saw a girl with Binoculars." has two different interpretations.
  Resolved: we can Resolve the content quality by link analysis and text-based judgement

4 DATASET

The smaller data(100 Mb approximately) set that is used for comparison between baseline and new model can be found here.

The data set used (The original data set used can be found here) is a subset of wikidump, which consists of various XML files. There has a XML file format of input data for this wiki search. It contains two major part in this data.
1. Title
2. Body
Title represents title of the document, which will be final result for any search and body part contains multiple things such as category, reference link, content of documents etc.

5 METHODOLOGY

5.1 PreProcessing

Preprocessing(Fig 1) is needed to be done in almost any kind of dataset, as most of it always consists some noise.The preprocessing steps that were used in the project are:

(1) **XML parsing:** XML parsing is a procedure to provide an interface to the user for accessing the XML document and to read an XML file. XML parser mainly verify the formation of an XML file and can also validate the document w.r.t XML schema. These are of two types:
- **SAX(Simple API for XML) :** It parse data node by node and doesn't store the XML in memory. In SAX parser we can't insert or delete a node. It perform top to bottom traversing.

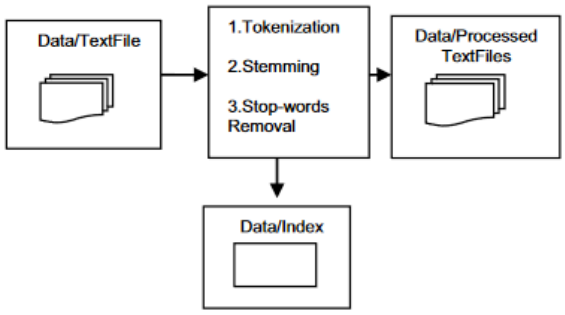

Figure 1: PreProcessing

- **DOM(Document Object Model) :** It first store the entire XML file in memory before processing here we can insert and delete the nodes. It can traverse in both top to bottom and bottom to top both direction.

(2) **Tokenization:** It is the process of converting the statements into list of tokens(single words), which can be comparatively easier for maintaining and storing whether a particular word is present in the document or not, which is created in posting list.
(3) **Case Folding:** Converting the entire data to either lowercase or uppercase.
(4) **Stemming:** This process removes suffixes and affixes from the word and only the root word is considered, which helps removing redundant data and gives more information about the document.
(5) **Creating posting list:** List that records which documents the term occurs in. Each item in the list - which records that a term appeared in a document is conventionally called a posting. It is later used for searching query.
(6) **Removing stop words:** Various words like, in, a, the, is, are; are basically used to represent grammar and proper formation of the statement becomes worthless when doing natural language processing by computer, hence words like these were removed.

5.2 Creating posting list/indexing

The posting list was created on the basis of Term Frequency- Inverse Document frequency (TF-IDF) methodology [12]. The indexing was made on five different fields:

- General (Considering all the texts)
- Title
- Body
- Infobox
- Category

5.3 Query searching

Before running the query on the posting list, the following preprocessing was done on the query:

- Tokenization
- Case Folding
- Stemming

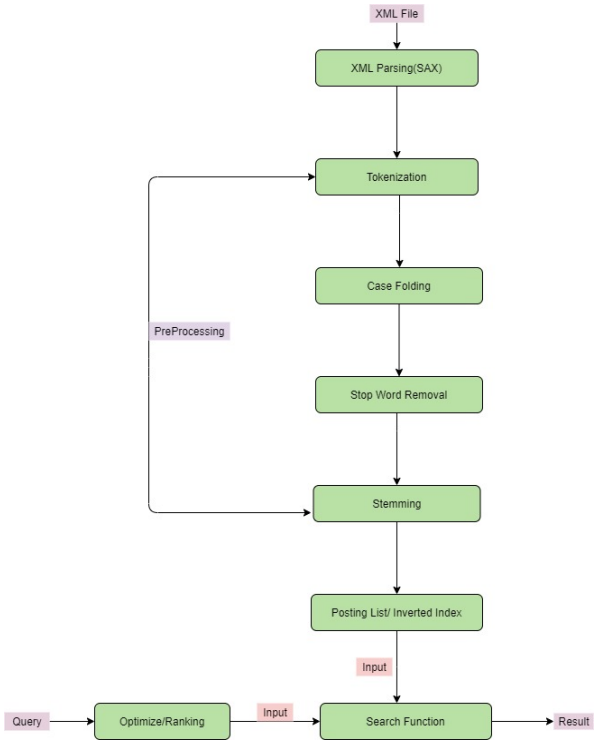

Figure 2: Flowchart

- Removing stop words
- Spelling Correction(for the time when the query written by the user has wrong spelling)

Two types of query searching is implemented:

- **General Query** - In this method the entire document is searched using the posting list created created using all the text in the data set eg new york , gandhi , 1981 world cup.
- **Field Based Query** -This is a more focused format of searching in the XML files where if entered, different sections or fields of the files are considered like Title, Body, Infobox and Category.e.g. title:gandhi body:arjun infobox:gandhi category:gandhi ref:gandhi

Things that our model returns are:

- **Time Taken** Time taken to retrieve the information needed from the posting list.
- **Number of document retrieved** Top 10 matched documents are retrieved by the model, if the number of documents retrieved is less than 10, than that will e specified.
- **Document list** Titles of all the document Retrieved are shown
- **Query after Auto-correction** The corrected spelling that we have used

6 EVALUATION

6.1 COMPARISON

Our model performs better than the baseline model in aspect of time taking. We have used term document incidence matrices for the baseline model. The comparison of time taking for index construction and query output for the query term **sherlock holmes** is given below-

Model Name	Index Time (in ms)	Query Time (in ms)
Our Model	62.83	3.80
Baseline Model	1950.89	7.10

For query term **telephone paved**, the comparison is given below-

Model Name	Index Time (in ms)	Query Time (in ms)
Our Model	62.83	2.48
Baseline Model	1950.89	4.90

For query term **telephone**, the comparison is given below-

Model Name	Index Time (in ms)	Query Time (in ms)
Our Model	62.83	1.36
Baseline Model	1950.89	3.25

6.2 ANALYSIS OF RESULTS

The results obtained from model was top 10 best matching documents based on similarity. Our model outperforms the baseline model in aspect of time taking. The baseline was not covering spelling errors. Our model covers the spelling error in query term, it will give the output with possible correct spelling term. But our model has certain limitations because it does not cover semantic matching. Our model does not cover any other form of data. So we can improve our model using link analysis and semantic matching.

7 CONCLUSION

Our model is taking very less time(less than a second) in searching a query on the Wikipedia dump of 81.8 GB. Indexing may take time to create but later searching is easy and less time taking. we can develop this model more to make indexing dynamic which can update itself periodically which can help in new file management. Overall system is very efficient which can perform more better on adding more feature .

REFERENCES

[1] Aris Anagnostopoulos, Andrei Z. Broder, and David Carmel. 2006. Sampling Search-Engine Results. *World Wide Web* 9, 4 (Dec. 2006), 397–429. https://doi.org/10.1007/s11280-006-0222-z

[2] P. Bafna, D. Pramod, and A. Vaidya. 2016. Document clustering: TF-IDF approach. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. 61–66. https://doi.org/10.1109/ICEEOT.2016.7754750

[3] Jeffrey Beall. 2008. The weaknesses of full-text searching. *The Journal of Academic Librarianship* 34, 5 (2008), 438–444.

[4] Johan Carlberger, Hercules Dalianis, Martin Hassel, and Ola Knutsson. 2001. Improving Precision in Information Retrieval for Swedish Using Stemming. (2001), 5.

[5] J. Shane Culpepper, Gonzalo Navarro, Simon J. Puglisi, and Andrew Turpin. 2010. Top-k Ranked Document Search in General Text Databases. In *Algorithms – ESA 2010*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann

Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Mark de Berg, and Ulrich Meyer (Eds.). Vol. 6347. Springer Berlin Heidelberg, Berlin, Heidelberg, 194–205. https://doi.org/10.1007/978-3-642-15781-3_17

[6] Atish Das Sarma, Anisur Rahaman Molla, Gopal Pandurangan, and Eli Upfal. 2015. Fast distributed PageRank computation. *Theoretical Computer Science* 561 (2015), 113–121. https://doi.org/10.1016/j.tcs.2014.04.003 Special Issue on Distributed Computing and Networking.

[7] Venkat N Gudivada, Vijay V Raghavan, William I Grosky, and Rajesh Kasanagottu. [n.d.]. RETRIEVAL ON THE WORLD WIDE WEB. *IEEE INTERNET COMPUTING* ([n. d.]), 11.

[8] Monika R Henzinger, Rajeev Motwani, and Craig Silverstein. 2002. Challenges in web search engines. In *ACM SIGIR Forum*, Vol. 36. ACM New York, NY, USA, 11–22.

[9] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. Association for Computing Machinery, New York, NY, USA, 133–142. https://doi.org/10.1145/775047.775067

[10] D. Minnie and S. Srinivasan. 2011. Intelligent Search Engine algorithms on indexing and searching of text documents using text representation. In *2011 International Conference on Recent Trends in Information Systems*. 121–125. https://doi.org/10.1109/ReTIS.2011.6146852

[11] Apra Mishra and Santosh Vishwakarma. 2015. Analysis of tf-idf model and its variant for document retrieval. In *2015 international conference on computational intelligence and communication networks (cicn)*. IEEE, 772–776.

[12] Takenobu Tokunaga and Iwayama Makoto. 1994. Text categorization based on weighted inverse document frequency. In *Special Interest Groups and Information Process Society of Japan (SIG-IPSJ*. Citeseer.

[13] Jian Wu, Kyle Mark Williams, Hung-Hsuan Chen, Madian Khabsa, Cornelia Caragea, Suppawong Tuarob, Alexander G. Ororbia, Douglas Jordan, Prasenjit Mitra, and C. Lee Giles. 2015. CiteSeerX: AI in a Digital Library Search Engine. *AI Magazine* 36, 3 (Sep. 2015), 35–48. https://doi.org/10.1609/aimag.v36i3.2601

[14] Hao Yan, Shuai Ding, and Torsten Suel. 2009. Inverted Index Compression and Query Processing with Optimized Document Ordering. (2009), 10.