COMPUSOFT, An international journal of advanced computer technology, 3 (7), July-2014 (Volume-III, Issue-VII)

ISSN:2320-0790

COMPUSOFT
An International Journal of Advanced Computer Technology

An Approach Paper-Compressed Indexing of Tweets for Information Retrieval

Ritu Godhani[1], Devishree Naidu[2]

[1]PG Student, RCOEM, Nagpur, India.
[2]Assistant Professor, RCOEM, Nagpur, India.

Abstract: In this paper, we present an approach for a compressed indexing of tweets for information retrieval in which, we take short 140-character text messages called tweets, preprocess tweets, and create the index, compress the index, and find percentage accuracy of the compressed index and uncompressed index. The paper also outlines literature review of some of the approaches used in optimizing inverted index compression.

Keywords: Information Retrieval; Inverted Index; Compression; Tweets; Inverted index Compression; Decompression;

I. INTRODUCTION

Information retrieval in computer science is the main activity for finding the related information on a particular topic out of collection or a corpus. Generally, information to be retrieved is text or documents but retrieval of multimedia on web is also a research discipline of Information retrieval. For information to be retrieved, Queries are used to find documents containing the information that is relevant to an area of interest. For example, search query in web search engines. Result is evaluated on the basis of most suitable results given by the query. Results are weighted and ranked according to ranking criteria. Information can be retrieved from the collection by sequentially searching through all stored document collection to get specific data. By applying an index, linear search is avoided and speed and efficiency of searches of the document collection is also increased. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. For eg. without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power. The idea of Indexing is important in the information retrieval. Idea of inverted indexing is standard in search engine indexing and information retrieval.

Inverted index is a data structure that is most commonly used in search engines. When evaluating a search query to quickly locate documents containing the words in a query and then rank these documents by relevance, Inverted index is used. Because the inverted index stores a list of the documents containing each word, the search engine can use direct access to find the documents associated with each word in the query in order to retrieve the matching documents quickly. We can build simple inverted index as below-Inverted index take and stores words or numbers from every document, creates a dictionary of unique terms, maps these terms to documents that contain the term. For Example, if 'Hello' word appears in Document 1, Document 2, Document 3. 'World' appears in document 2, Document 3, Document 4, Then Inverted index looks like-

Table I. Simplified Example of Inverted Index

Word	Document No
Hello	1,2,3
World	2,3,4

Inverted index provides full text searches, i.e. each word of the document collection will be in the index. Thus, upon searching, we get all the relevant documents related to the word. If the collections are large, then the entries in the inverted index also gets huge , because of which it needs to be compressed for efficient storage representation, as maintaining inverted index for large scale data requires considerable storage. A form of compression is required to reduce the size of indices on the disk. Various data compression schemes can be applied for inverted index.

II. LITERATURE REVIEW

Hussein Al-Bahadili and Saif Al-Saab [1] proposed a new web search engine model called the compressed index-

query (CIQ) Web search engine model in which different components of the new Web search engine model are implemented in a prototype CIQ test tool (CIQTT), which is used as a test bench to validate the accuracy and integrity of the retrieved data from the search engine model. This model incorporates two bit-level compression layers implemented at the back-end processor (server) side, one layer resides after the indexer acting as a second compression layer to generate a double compressed index (index compressor), and the second layer resides after the query parser for query compression (query compressor) to enable bit-level compressed index-query search. The data structure used for indexing is Inverted indexing method. The data compression algorithm used in this model is the Hamming codes based compression (HCDC) algorithm. They show that their new web search engine model proved accuracy with 100% agreement between the results retrieved by the CIQ and the uncompressed model.
The new model demands less disk space and providing a reduction in processing time of over 24%.

Diego Arroyuelo, Senén Gonzalez et.al [2] proposed a method for reassigning the document-id numbers to given subset of inverted lists in inverted index using IBDA (Intersection Based Document Identifier Assignment). This method uses the inter-list dependencies (i.e., the list intersections) to assign docIDs. As a result, these intersections become runs when encoded as DGaps. For instance, Consider The following example:I_1 (10, 30, 65, 66, 67, 70, 98), I_2 (20, 30, 66, 70, 99, 101) 10 → 1, 30 → 2, 65 → 3, 66 → 4, 67 → 5, 70 → 6, 98 → 7. Now, docIDs 30, 66 and 70 are fixed. Hence, we assign 20 → 8,99 → 9,101 → 10. After the process, the result is I_1 (1, 2, 3, 4, 5, 6, 7) and I_2 (2, 4, 6, 8, 9, 10).Now, Start remunerating the intersection $I_1 \cap I_2$ as IBDA Scheme, so first take intersection of I_1 and I_2 then assign DocId numbers 30 → 1, 66 → 2, 70 → 3. Then remunerate the remaining elements in I_1 as 10 → 4, 65 →5, 67 → 6, 98 → 7. Finally, remunerate the remaining elements in I_2 as 20 → 8, 99 → 9, 101 → 10. The resulting lists are I_1(1, 2, 3, 4, 5, 6, 7) & I_2(1, 2, 3, 8, 9, 10). Some of the most effective compression schemes are applied to run-length encode runs of 1s in inverted lists. They have shown that by using this approach, not only the performance of the particular subset of inverted lists is improved, but also that of the whole inverted index.

In this paper [3] Authors propose a novel class of encoders called VSEncoding from Vector of Splits. Encoding work by partitioning an list of integers into blocks which are efficiently compressed by using simple encoders. They carry out this important step via dynamic programming, which leads to produce the optimal solution. Experiments show that simple class of encoders outperform all the existing methods in literature by more than 10% still retaining very fast decompression.

Naiyong Ao, Fan Zhang,Di Wu [4] propose a schema for efficient parallel Inverted lists intersection and inverted index compression algorithms using Graphics Processing Units. Authors present several novel techniques to optimize lists intersection and decompression, particularly suited for parallel computing on the GPU. Motivated by the significant linear characteristics of real-world inverted lists, authors propose the Linear Regression (LR) and Hash Segmentation (HS) algorithms to contract the initial search range of binary search.

Jimmy Lin from twitter [5] presented and evaluated a full-text index to optimize selection operations on free-text fields for analytics applications within records of Hadoop-based processing. Full-text indexing is performed on Tweet dataset and created pseudo documents consisting of the text of all tweets contained in the HDFS file block. These pseudo documents are then indexed with Lucene. Data is LZO compressed format. Upon submission of a Hadoop job, the inverted index is consulted for blocks that meet the selection criterion; Inverted index informs the Hadoop execution engine which compressed data blocks contain query terms of interest, and only those data blocks are processed and decompressed.

II. PROPOSED METHOD

A. COMPRESSED INDEXING OF TWEETS FOR INFORMATION RETRIEVAL

The method proposed covers basically five main phases- Collection of Data, preprocessing the Data, indexing, Compression, searching and comparison of the results.

1) Collection of Data: We collect documents directly from the web. Our Documents are tweets only. Twitter is an online social networking service that enables users to send and read short 140-character text messages, called tweets. Twitter is a micro blogging site where people can share their views via tweets. Users of micro-blogging website post views on different topics on twitter. This leads to heavy user generated content. Tweets are different from other online social networks messages. The limit length of a Twitter message is 140 characters. User can't write a paragraph as he does while writing a movie review in some of the other forums. Twitter users make use of the "@" symbol to refer to other users on Twitter. Users use hash tags "#" to mark topics."RT" is used to indicate that this tweet is written again.
Following is the example of two sample tweets where Date, time when tweet was created, username, tweet text is given-

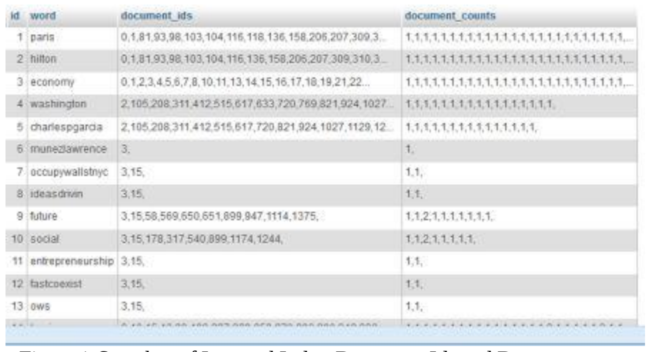
26/06/2014 08:48 ritugodhani1 I just saved my followers with@Tweet_Download (http://t.co/bLSurRGO0m)!

26/06/2014 07:49 ritugodhani1 this is my tweet.

2) Preprocessing the tweets: Tweets are fed as input to the preprocessing step. Tweets will be subjected to a number of preprocessing steps to make it useable for the next steps. The preprocessing phase aims to extract the relevant textual parts and prepares them for building the index.Pre-processing of tweets means applying a number of procedures for removing noise words.

We preprocess tweets as follow-
1) We start preprocessing tweets where actual tweet text begins from. We leave username, Date and Time and Other metadata information.
2) Remove all Stop-words and Stop Symbols from the text.
3) Remove Urls in tweet, for eg.http://t.co//ftr76u5h.
4) Tweets have some words RT (retweets), @username, #topic code.
RT, @, #, these symbols should be removed and words after #,@ should be preserved.
5) Using Stanford POS tagger, remaining text is parsed to get distinct terms and phrases assigned as most correct Parts of speech like noun, adjective, adverb, proper nouns.
After preprocessing the above sample tweets, we get following keywords:
Followers, Tweet_Download,tweet.

3)*Indexing:* We start numbering the documents i.e. tweets starting from number 0.Output of parser will be used as input to build index. To generate inverted index, we scan and list all keywords that occur in parsed tweets. For every word that occurs in tweets, we store its tweet number in which it occurs and frequency of the word in that tweet. We store document number and document count separately.



id	word	document_ids	document_counts
1	paris	0,1,81,93,98,103,104,116,118,136,158,206,207,309,3...	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,...
2	hilton	0,1,81,93,98,103,104,116,136,158,206,207,309,310,3...	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,...
3	economy	0,1,2,3,4,5,6,7,8,10,11,13,14,15,16,17,18,19,21,22...	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,...
4	washington	2,105,208,311,412,515,617,633,720,769,821,924,1027...	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
5	charlespgarcia	2,105,208,311,412,515,617,720,821,924,1027,1129,12...	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
6	munezlawrence	3,	1,
7	occupywallstnyc	3,15,	1,1,
8	ideasdrivin	3,15,	1,1,
9	future	3,15,58,569,650,651,899,947,1114,1375,	1,1,2,1,1,1,1,1,1,1,
10	social	3,15,178,317,540,899,1174,1244,	1,1,2,1,1,1,1,1,
11	entrepreneurship	3,15,	1,1,
12	fastcoexist	3,15,	1,1,
13	ows	3,15,	1,1,

Figure 1.Snapshot of Inverted Index-Document Ids and Document Counts Stored Separately.

4)*Compression:* As we are storing document numbers and document Counts separately, we can compress them individually, For Doc_Count, We use run length encoding, since many runs of consecutive 1's are generated. We will also optimize the compression of Doc_Ids.

5) Searching and Comparison: To carry out the searching, we need to read the query and examine the keywords or phrases to be searched.Then,After searching process, take out the document numbers that match all or some keywords/phrases. Decompress the data.
To confirm the accuracy, we check both the number of extracted documents in an uncompressed index and compressed index. In this procedure, the results of different search processes for the same keywords are compared.

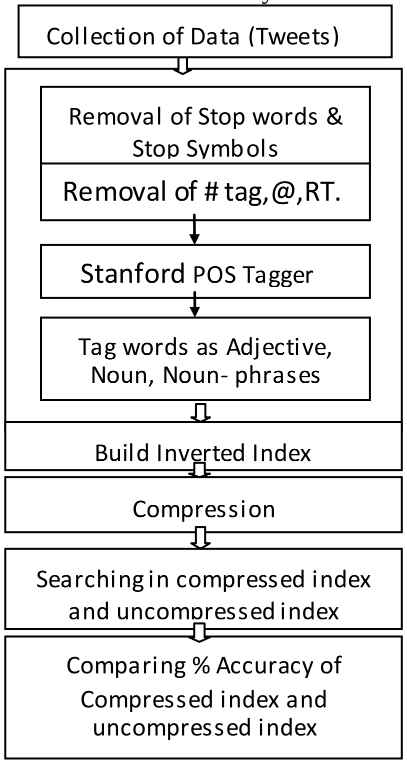

Collection of Data (Tweets)
Removal of Stop words & Stop Symbols
Removal of # tag,@,RT.
Stanford POS Tagger
Tag words as Adjective, Noun, Noun- phrases
Build Inverted Index
Compression
Searching in compressed index and uncompressed index
Comparing % Accuracy of Compressed index and uncompressed index

Figure 2.Main Components of Proposed Approach

IV. CONCLUSION

In our proposed method, we have preprocessed tweets and indexed keywords of the tweets using inverted index. We are storing Document ids and documents counts separately, We will optimize inverted index storage of Document counts by run length encoding. We will also optimize the storage of Document Ids.
From the above approach, our aim will be to obtain maximum accuracy of the retrieved data and to show that compressed index performs well as compared to uncompressed index.

REFERENCES

[1] Al-Bahadili, Al-Saab. "Compressed Index Query web Search Engine Model", International Journal of Computer Information Systems (IJCIS), Vol. 1, No.4, 73-79.

[2] Diego Arroyuelo, Senén Gonzalez."Document Identifier Reassignment and Run-Length Compressed Inverted Indexes for Improved Search performance", *SIGIR'13, Copyright* 2013 ACM.

[3] "VSEncoding: Efficient Coding and Decoding of Integer Lists Via Dynamic Programming", SIGIR'10, Copyrights ACM.

[4] Naiyong Ao, Fan Zhang, Douglas S. Stones. "Efficient Parallel Lists and List Intersection and Index Compression Using Graphics

Processing Units", Proceedings of VLDB Endowment, Vol.4, Copyright 2011 VLDB Endowment.

[5] Jimmy Lin, Twitter. "Full-Text Indexing for Optimizing Selection operations in Large-Scale data Analytics", ACM.

[6] Chun Chen, Feng Li."TI: An Efficient Indexing Mechanism for Real-Time Search on Tweets", *SIGMOD'11*, Copyright 2011 ACM.

[7] Akshi Kumar, Teeja Mary Sebastian."Sentiment analysis on twitter", IJCSI International Journal of Computer Science Issues, Vol.9, Issue 4, No. 3, July 2012.