TEXT SEARCH ENGINE

Sakshi Jain, Dr. Rupesh C. Jaiswal
E&TC Department, Pune Institute of Computer Technology, Pune, Maharashtra, India.

Abstract:

Text search engine is a prototype of search engine like Google which works on millions of Wikipedia pages (which are in original XML format) and retrieves the top 10 relevant Wikipedia documents that matches the input query. This search engine takes Wikipedia corpus in XML format which is available at wikipedia.org as input. So a data structure is build which will store all pages. And then the crucial part is how is top 10 pages are selected out of all pages. So logic and algorithm to sort those top 10 pages out of all available pages is applied. Using indexing it retrieves all the relevant ranked documents.

Introduction :

Text search engine is real time. Rather than doing exact search it uses full text search engine. Using full text search engine searches are shown having same context or is related to input. Text search engine is able to show the results in milliseconds. It uses the data structure which helps in showing the user the relevant information in real time. It uses stemming for making search efficient rather than finding exact match. By using machining learning algorithms search can be optimised as required.

Literature Survey :

To facilitate the full text search , first the text is analysed and result is used to build inverted index.
Say some millions of pages are available so text search engine breaks this pages into words(which is called as tokens) and then data structure is created which looks like matrix in which inverted index is used. So word is searched and all documents having particular word are used to build inverted index.

Words	Document 1	Document 2
This	present	present
is	Not present	present
text	present	Not present
search	present	present
engine	present	present

Fig 1: Data Structure

Fig 1 represents data structure which is created to store words from pages and then words are sorted to make search efficient. Then text pre-processing is done using white space tokenizer, Penn tree bank tokenization and case folding XML is used as communication medium between two computers to make communication clear and readability easy. XML is Extensible Markup Language. As referred in [1], an XML parser first groups a bit sequence into characters, then groups the characters into tokens, and finally verifies the tokens and organizes them into certain data representations for analysis at the access stage. XML tags identify the data and are used to store and organise data. Based on our requirements we can build any no. of tags. XML is used to create your own self descriptive tags, or language that suits your application. XML parsing is

expensive operation. It carries data and does not display it unlike HTML. There are many ways in which XML parsing can be improved as referenced in [8]. To execute XML, java has package named SAX parser. It parses entire XML document and based on tokens it will call appropriate token in handler. Code which is present in content handler, it can override based on need. Rather than creating parse trees SAX will read XML byte by byte. SAX is event based parser for XML documents. It receives event notification about the XML document being processed, an element and attribute at a time in sequential order starting at the top of the document. It reads the XML document from top to bottom, recognizing the tokens that make up a well formed XML document. Tokens are processed in the same order that they appear in the document.

High level design

Module 1:

1. Pages are separated using XML tags from Wikipedia dump. So each Wikipedia page in will have one java object. And each object have title, summary, external links, category, text-contents as attributes.

Title: title of the page
Summary: summary of the given page
External links: Links of the other pages present
Category: Page belongs to which particular category.
Text content: content of Wikipedia page

Module 2:

1. Each page object is processed and then text pre-processing is done using white space tokenizer, Penn tree bank tokenization and case folding and stemming.

2. In every Wikipedia page, depending on where a word has occurred, weightage of that page is going to vary. HashMap are used to implement this where key will be word and value will be the count of words.

Module 3:

1. Inverted indexing is done to find out in which all documents a particular word has occurred. So data structure is build having key as word and value as document having that value.

2. If data is more than RAM capacity than it is stored in hard disk and data is retrieved from hard disk part by part and then it is merged which is known as external sorting.

Logic Flow:

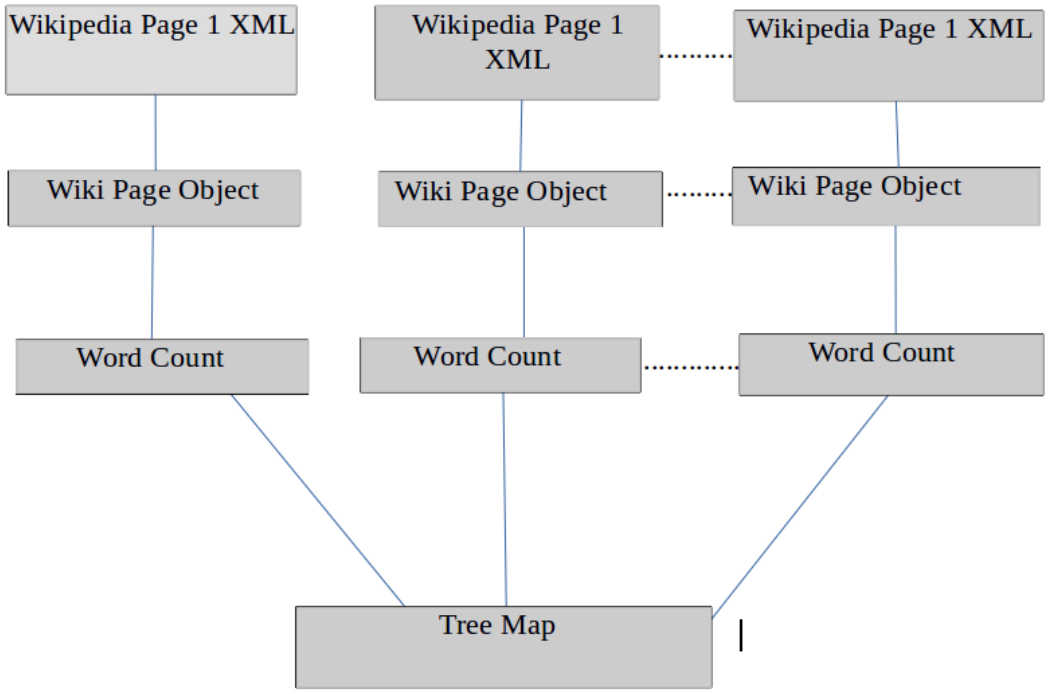

Fig 2: Logical Flow of project

Fig 2 shows tree map is new data structure. Whatever written in HashMap is written in tree. For every word there will be only one entry is created in tree map and that entry will contain how many times word has occurred in what documents. For every object HashMap are created individually.

Results:

Fig 3: Output

“adobe” keyword is searched in documents and Titles of documents having “adobe” is displayed. “adobe” is present in documents having title --

1. Adobe Director
2. Adobe Persuasion
3. 8BF
4. AS2
5. Aldikos

Conclusion:

Text search engine is implemented which takes XML as input file and than this files are parsed. To execute

XML, SAX parser is used to read XML byte by byte. HashMap data structure is used to store the word counts. Linear sorting takes lot of time and is inefficient so indexing is used to rank pages according to the weightage given. All things written in HashMap are inserted in tree map which is a new data structure. This text search engine helps in searching words efficiently. Machine learning algorithms can be used to make it work more efficiently.

Acknowledgement:

I am thankful to Dr. R C Jaiswal for motivating and guiding me through research. Because of his guidance and mentoring, this paper got insight which is needed to make it presentable one. His constructive feedback at every stage was of great help.

Reference:

[1] Tak Cheung Lam, Jianxun Jason, DingJyh-Charn Liu, "XML Document Parsing: Operational and Performance Characteristics", IEEE Computer Society, 2008.

[2] A. Slominski, "On Performance of Java XML Parsers"; www.cs.indiana.edu/~aslom/exxp.

[3] Adam Dukovich, Jimmy Hua, Jong Seo Lee, Michael Huffman,AlexDekhtyar, "JOXM: Java Object --XML Mapping" Published in EightInternational Conference IEEE onWeb Engineering, 2008, ICWE'08 ,1481 July 2008,Pages 332-335,ISBN 978-0-7695-3261-5.

[4] Diomidis Spinellis "Using and Abusing XML" in IEEE Computer Society 0740-7459/08/2008 IEEE

[5] http://www.w3schools.com/XML/XML-whatis.asp

[6] Toshiro Takase, Hisashi MIYASHITA ,Toyotaro Suzumura, Michiaki Tatsubori, " An Adaptive, Fast, and Safe XML Parser Based on Byte Sequences Memorization " published in ACM transactions 2005 May 10-14 Chiba Japan 1-59593-046-9/05/0005.

[7] Vaishali M. Deshmukh, G.R Bamnote, "An Empirical Study of XML Parsers across Applications" ,IEEE International Conference,2015

[8] Jie tang, Shaoshan Liu, Chen Liu, Zhimin Gu, Jean-Luc Gaudiot, "Acceleration of XML Parsing through Prefetching", *IEEE Transactions on Computers*, vol. 62, no. 8, August 2013.

[9] S. Karre, S. Elbaum, "An Empirical Assessment of XML Parsers", *6th Workshop on Web Engineering*, pp. 39-46, 2002.

[10] V. H. DINH. (2006, November), Hash Table [Online]. Available: http://libetpan.sourceforge.net/doc/API/API/x161.html

[11] Chouvalit Khancome, Veera Boonjing, "Character-Based Indexing Using Inverted Lists", IEEE International Conference, 28 December 2009

[12] Java and XML (O'Reilly Java Tools) 1st Edition by Brett McLaughlin (Author)

[13] Processing XML with Java™: A Guide to SAX, DOM, JDOM, JAXP, and TrAX by Elliotte Rusty Harold

[14] https://www.freejavaguide.com/xml-part1.pdf

[15] Introduction to Information Retrieval by Christopher D. Manning, Prabhakar Raghavan