International Journal of Electrical Electronics & Computer Science Engineering
Volume 4, Issue 3 (June, 2017) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

Auto Suggestion Based on Lucene

Dr. M. Nagaratna[1], Bhavani Lakkarusu[2]
[1]Associate Professor, [2]Student, CSE Department, JNTUH College of Engineering, Hyderabad, Telangana, India
[1]mratnajntu@gmail.com, [2]bhavanilakkarusu@gmail.com

Abstract: This paper introduces us the Auto Suggest search engine based on Lucene. Auto Suggest is searching a dynamic list of related phrases, keywords and items. Now a days, many online websites provide autosuggest feature. Auto suggest predicts the buyers train of thought in suggesting new ideas, and also suggests items in addition to the buyers query .The full text search of Lucene provides faster retrieval speed compared to other technologies.

Keywords: Full Text Search; Lucene; Keyword Search; Analyzing infixsuggestor; Drupal;

I. INTRODUCTION

With the rapid development of Internet and with the explosive growth of online websites how to remove the impurities and obtain the information easily they need from the massive information with fast retrieval speed is a hot topic.

One disadvantage of database search technology is low efficiency .full text search of apache lucene resolves that problem. The core of auto suggestion is the full-text retrieval technology. Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It creates all the possible terms in the index which are searched by network users frequently, allows ordering massive information and enables user to quickly and easily retrieve any information they need. Lucene is a pure Java software project which is free and open source. In recent years Lucene has become most praised and most popular retrieval java library.

II. RELATED WORK

A. Introduction of Lucene:

Lucene is fully featured text search engine written in java. It can be placed within any application to realize auto suggestion feature [2]. It provides ranked searching i.e, best results returned first. Scope of its application is wide. Some of its applications are Eclipse help system [7] and Jive [4].

B. Systematic Structure of Lucene:

Lucene has systematic and clear package structure .Initially it scans each and every word of paper and then creates indexes, later uses them for searching. The packages used for indexing and searching are as follows:

1) *org.apache.lucene.analysis*: Its main function is to convert text into indexable tokens. It segments document and removes stop words. Stop words are words which are of no help for retrieval and they occur very frequently such as “The”,”a” etc. Lucene also provides analyzers such as SimpleAnalyzer, StandardAnalyzer, FrenchAnalyzer etc.

2) *org.apache.lucene.document:* Its main function is document management. Document is same as record in relational database. Analyzer can understand document only. Document consists of text field and data field.

3) *org.apache.lucene.index:* Its main function is index management. This includes index creation and index deletion. Here tokens are used to create index and full-text search can only be achieved by index traversal. This greatly improves efficiency of information retrieval.

4) *org.apache.lucene.queryParser*: Its main function is to operate on keywords. It parsers the user query and then pass the searcher.

C. Functionality of Lucene:

Functionality of Lucene includes index creation and index search module. Index creation module includes creating index after word segmentation and then adds indexed record to repository. Index search module includes querying index based on parameters and then returns result to user.

Index creation module uses following classes: *IndexWriter*, *Directory*, *Analyzer*, *Document* and *Field*. It includes three steps[5]: extraction of text , creation of document, analysis and creation of indexes.

Index search module uses following classes: *QueryParser*, *IndexSearcher* and *Hits*. It includes query creation, query parsing and return result [6].

III. PROPOSED ALGORITHM

A. Description of Algorithm

Aim of the proposed algorithm is to provide an auto suggestion for a website user. The proposed algorithm is consists of following steps.

1) *Building a Database*: Initially we need to build a database to store all information needed by user. We need to choose best database management system. Some of them are hibernate, my ibatis, drupal. The model should have low degree of redundancy. Out of all drupal provides more flexibility. It allows non technical users to edit, manage content without any prior knowledge.

2) *Creating Indexes:* Here we create indexes using the

data from database. We make use of StandardAnalyzer which analyzes document. Standard Analyzer internally uses Tokenizer and filter. Tokenzier reads data from document and converts it into tokens. Then Tokenfilter takes those tokens, removes stop words from it. Stop words are words which frequently occur and they do not help in indexing. For Example in sentence "The apple is red" here *the* and *is* are stop words. Then from those filtered tokens we create indexes using AnalyzingInfixSuggestor. It takes tokens and Directory where to store indexes as input. Lucene provides many indexing types. We are using RAMDirectory which uses physical memory to store index.SimpleFSDirectory uses JAVA IO API to store index on hard disk, NIOFSDirectory uses JAVA NIO API to store index on hard disk. RAMDirectory gives best performance for small indexes. Before indexing, it even checks if there any synonyms if so it makes use of Synonymfilter to map those words.

3) *Searching and sorting results:* Whenever user types a word, every keystroke generates a new query. Then QueryParser parses query and then based on the query we perform lookup on AnalyzingInfixSuggestor. It returns result in the form of Lookup.LookupResult. Then we perform readobject() method on it which returns hits as output .we can even filter the results by specifying amount of data we want. Then the result is given to the user.

IV. EXPERIMENTAL RESULTS

Lucene can also be used to perform searching with multiple indexes. Suppose whenever a user types of "A" we want to list all the cities, states etc. Here we create different AnalyzingInfixsuggestor for cities, states and country etc. Depending upon whether we want to show only cities or states or both we perform lookup on that corresponding stored AnalyzingInfixSuggestor.

The following are response times for different scenarios:

Table I. (Table for Time Taken to Fetch Results)

Response Time (Ms)	Lookup on Indexes	Sample Results
179	Town	Amaravathi, Aluva etc.
283	City, Town	Amaravathi, Aluva, Ambaji City, Amalapuram City etc.
313	City, Town, State	Amaravathi, Aluva, Ambaji City, Amalapuram City, Andhra Pradesh, Arunachal Pradesh etc.
622	City, Town, State, Country	Amaravathi, Aluva, Ambaji City, Amalapuram City, Andhra Pradesh, Arunachal Pradesh, Argentina, Austria etc.
907	City, Town, State, Country, Continent	Amaravathi, Aluva, Ambaji City, Amalapuram City, Andhra Pradesh, Arunachal Pradesh, Argentina, Austria, Asia, Africa, Antartica etc.

V. CONCLUSION

In this paper we mainly focus on designing Auto suggest service based on Lucene. The above experimental results shows proposed algorithm has faster retrieval speed. The proposed algorithm performs better than other auto suggestion methods. Lucene is a high-performance, full-featured text search engine library written entirely in Java. *Drupal* is a content management system which provides both user interface and rest services. In addition, implementation of *SynonymFilter* provides a good experience to user. There are other search libraries which are being developed based on Apache Lucene such as ElasticSearch etc.

VI. REFERENCES

[1] Lang-XaoWei, Wang-ShenKang, Full-text Search System Application Research and Implement [J]. Computer Engineering, 2006.2, Volume32 (4) : 94-96, 99.

[2] Guan-JianHe, Gan-JianFeng, Full-text Search Engine Application Research and Implement Based on Lucene[J].Computer Engineering & Design, 007.1, Volume28 (2) : 489-491

[3] Otis Gospodnetic;Erik Hatcher. Lucene in Action. 2007.4.

[4] Su-TanYing, Guo-XianYong, Jin-Xin. One Chinese Full-Text Search System Based on Lucene[J]. Computer Engineering, 2007.11, Volume3(23):94-96

[5] Che-Dong, Full-Text Search Engine Introduction Based on Java, http://www.chedong.com/tech/lucene.html, 2002.8.

[6] Steven J. Owens, Lucene Tutorial, http://darksleep.com/lucene/.2008.7

[7] http://www.eclipse.org/

[8] https://www.toptal.com/database/full-text-search-

of-dialogues-with-apache-lucene

[9] B. Y. Lin, R. Zhao and L. C. Chen, "Research and Application of Full-Text Search Engine Based on Lucene," Computer Technology Development,Vol. 17, No. 5, 2007, pp. 184-187.

[10] X. Zhang and Y. Zhou, "Improvement of an Algorithm for Ranking Pages Based on Lucene," Computer System Application, Vol. 18, No. 2, 2009, pp. 155-158.

[11] W. He, S. J. Xue, M. R. Kong, et al., "Design and Implementation of Full-Text Search Engine Based on Lucene," Information Science, Vol. 3, No. 9, 2006, pp. 88-89.

[12] http://lucene.apache.org/core/

[13] Y. C. Li and H. F. Ding, "Research and Application of Full-Text Search Engine Based on Lucene," Computer Technology and Development, Vol. 20, No. 2, 2010, pp. 4-56.

[14] Q. Zhao, "Information Retrieval," China Machine Press, Beijing, 2008.