AMSE JOURNALS –2014-Series: ADVANCES D; Vol. 19; N° 1; pp 1-14
Submitted Nov. 2013; Revised Jan. 26, 2014; Accepted Feb. 10, 2014

Performance Analysis on Graph Based Information Retrieval Approaches

[*]P.Janarthanan, [**]N.Rajkumar, [*]G.Padmanaban, [*]S.Yamini

*Dept. of Computer Applications, Sri Venkateswara College of Engg.,
Pennalur - 602 117, Tamil Nadu,India
[**] Dept. of CSE, PG, Sri Ramakrishna Engineering College,
Coimbatore-641022, India
(janap@svce.ac.in, nrk29@rediffmail.com, padmanabang.bca@gmail.com,
yaminisandy@gmail.com)

Abstract

Information Retrieval system (IRS) is very popular research topic in the world. Now a days, to retrieve a particular text unit either word or document from large text repository is a challenging task. In an information retrieval process, the information retrieved based on user query by matching user query to document repository consumes more time. Instead of exact query match, the set of keywords will be used to find the relevant documents from document repository. Before searching document repository, the documents details are also maintain in the form of keywords. Most of the research scholar and the search engines are used the advanced technique called Indexing. Indexing is a technique is used to store and retrieve the keywords and their details in efficient manner. To reduce the index size, we have to apply stemming technique to keywords. Stemming is the process of reducing a word to its stem and a stemmer or a stemming algorithm is a computer program that automates the task of stemming. This analysis work is very helpful to know the techniques and how to improve the various indexing technique and stemming algorithms. This paper discuss and analysis the performance of some indexing techniques and stemming algorithms.

Key words Edge Index Graph, Document Index Graph and Inverted Index.

1. Introduction

Information Retrieval System is the fundamental requirement of documents in a collection that must be retrieved in order to satisfy a user's need for information [1]. In an information retrieval, stemming and indexing are important tasks. Before preceding the indexing technique, first apply the stemming process to the document repository. Stemming is a process of converting the words having morphological similarity into one common form. A stemming algorithm is applied to minimize a word to its stem or root form and it reduces the size of index [11][12]. There are several stemming algorithms available such as Porter's stemming algorithm, Peak and Plateau method, Table lookup approach, Lovin's stemming algorithm and Paice/Husk stemming algorithm [2][5].

In an information retrieval, the relationship between a query and a document is determined primarily by the number and frequency of terms which they have in common [13]. Searching user query on non standardized format of large document is highly difficult where in indexing reduces the complexity of search process [14]. Indexing is a process of identifying keywords to represent a document based on their contents. Indexing is very important phase of Information Retrieval System to create a searchable words or documents for the given query. Basically, indexing is maintained to all document details with the respective keywords or descriptive terms representing the document [3][15].

Indices can be constructed in three ways. They are Manual Indexing, Automatic Indexing, and Semi-Automatic Indexing. Manual indexing is a time taking process and it requires huge manual hours to index a repository which grows day by day [9]. The computer system is used to record or store the user generated indexing terms and their document details. Automatic text indexing which is faster and less error-prone has become a common practice on big corpus [10]. Human only contributes by setting parameters or thresholds or implementing the algorithm. Most of the search engines using this indexing technique. It shows the retrieval effectiveness of automatic indexing. Semi-Automatic indexing is including some properties of automatic indexing system and including some properties of related system references [4].

In this paper we have been analyzed the faster indexing technique called Automatic indexing techniques and their types. Automatic Indexing is done by a machine according to the rules framed in the program. It is a better indexing approach as it takes away the time, cost, exhaustively, specificity, vocabulary, searching and browsing limit and allows the entire document to be analyzed. But it has the option to be directed to particular parts of the document. Some of popular automatic indexing techniques are Edge Index Graph (EIG), Document Index Graph (DIG) and Inverted Index [6][8].

2. Stemming algorithms

Stemming is a process of obtaining unique root word from given documents. There are several stemming algorithms available. Here we have been analyzed some efficient stemming algorithms called Porter's stemming algorithm, Lovin's stemming algorithm and Paice/Husk stemming algorithm[7].

2.1 Porter's Stemming Algorithm

The Porter stemmer algorithm is systematic and stepwise process. Its main target is removing the endings from the words in English. This algorithm is common to all the words in English. It takes the original word as the input and gives the stemmed word as the result. The stemmed word is called the root word, which may not have any meaning and the stemmed words are inserted into indexes. The Porter's stemming algorithm consists of five step process and a common word is in the following form:

[C] (VC) m [V]

Where **C** - List of Consonant, **V** - List of Vowel and **m** – Measure of any word or word part.

The rules for eliminating a suffix will be given in the form

(Condition) S1 ◊ S2 S2 replaces S1 if condition satisfied.

Step 1 It works with plurals and past participles.

a) SSES → SS caresses → caress
   S → cats → cat

b) (m>0) EED → EE feed → feed
   (*v*) ED → plastered → plaster

Step 2 Deals with model matching on some common suffixes.

(m>0) ATIONAL → ATE relational → relate
(m>0) TIONAL → TION conditional → condition.

Step 3 It processes special word endings.

(m>0) ALIZE → AL formalize → formal

Step 4 Examines the stripped word against more suffixes in case the word is compounded.

(m>1) ANCE → allowance → allow

Step 5 Checks if the above word ends in a vowel and fixes it appropriately

(m>1) E → probate → probat

Rate → rate

Example 1 Input Word = GENERALIZATIONS

Step 1 Compares with the plural list

S → generalizations → generalization

Step 2 Compares with the pattern matching

(m>0) IZATION → IZE generalization → generalize

Step 3 Compares with the special word endings

(m>0) ALIZE → AL generalize → general

Step 4 Compares with suffix words list

(m>1) AL →general → gener

Step 5 The Stemmed word of "generalizations" is "gener"

2.2 Lovin's Stemming Algorithm

Lovin's stemming algorithm is a single pass stemming algorithm. It is a context sensitive stemmer. This algorithm removes endings based on the longest-match principle. This approach removes a maximum of one suffix from a word due to its nature as single pass algorithm. It uses list of 250 distinct suffixes and removes the longest suffix attached to the word and ensuring that the stem after the suffix has been removed is always at least 3 characters size. Then the end of the stem may be reformed by referring to a list of recoding transformations. The Lovin's stemmer consists 294 word endings, 29 conditions and 35 conversion rules. Each ending is associated with one of the conditions. It consists of two steps as follows:

Step1 If longest ending is found which satisfies one of the associated conditions, then it is removed.

Step2 The 35 rules are applied to transform the ending.

The second step is done irrespective of an ending is removed in the previous step.
The few lists of endings are represented in table 1. They are grouped by length from 11 characters down to 1 character. Each ending is followed by its condition code.

The few lists of conditions are represented in table 2**.** Lovin’s has 29 conditions called A to Z, AA, BB and CC (* stands for any letter). These are the codes for context-sensitive rules associated with certain endings.

Table 1. List of Endings

11 Characters Endings	**Condition Name**	**03 Characters Endings**	**Condition Name**
Alistically	B	Acy	A
Arizability	A	Age	B
izationally	B	Als	BB
10 Character Endings	**Condition Name**	.	.
anitialness	A	**02 Character Endings**	**Condition Name**
arisations	A	Ae	A
arizations	A	Al	BB
.	.	Ar	X
09 Character Endings	**Condition Name**	.	.
Allically	C	**01 Character Endings**	**Condition Name**
antaneous	A	A	A
eableness	E	E	A
.	.	S	W

Table 2. Codes for context sensitive rules

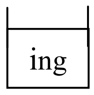
Condition Name	**Rule Description**
A	No restriction on stem
B	Minimum stem length = 3
C	Minimum stem length = 4
D	Minimum stem length = 5
E	Do not remove after ending ‘e’
F	Minimum stem length = 3 and do not remove after ending ‘e’
.	.
.	.
Z	Do not remove after ending ‘f’
AA	Remove ending only after ‘d’, ’f’, ‘ph’, ‘l’, ‘er’, ‘or’, ‘es’ or ‘t’
BB	Minimum stem length = 3 and do not remove ending after ‘me’, ‘t’ or ‘ryst’

The algorithm has 35 transformation rules few of them listed in table 3 which are applied to recoding stem terminations. This step is done whether or not an ending is removed in the first step.

Table 3. List of transformation rules

R.No.	Transformation Rule	R.No.	Transformation Rule
1		.	.
2	iev → ief	30	ent → ens except following m
3	uct → uc	31	ert → ers
4	umpt → um	32	et → es except following n
5	rpt → rb	33	yt → ys
.	.	34	yz → ys

Put in stack



Rule N : Minimum stem length = 4 after s,**

elsewhere = 3

training → train

Apply Step 2: Scan the transformation rules. No rule found.

Final Output : training → train

2.3 Paice/Husk Stemming Algorithm

Paice/ Husk is a simple affix removing and replacement algorithm. It is an iterative algorithm and comprises 120 rules for deletion and replacement. This stemming technique obtains the root word in the document with high accuracy. There are two important steps associated with this approach if we apply this algorithm. The above said steps are:

Step1 Reverse the word

Step2 Map characters to its 120 rules.

Table 4 List of Paice / Husk Rules

Rule	Deletion	Replacement	Rule	Deletion	Replacement
ai*2	-ia	---	j1s	-j	-s
a*1	-a	---	lbaifi6	-ifiabl	---
bb1	-bb	-b	lbai4y	-iabl	-y
city3s	-ytic	-ys	lba3	-abl	---
ci2	-ic	---	lbi3	-lbi	---
cn1t	-nc	-nt	lib2l	-bil	-bl
dd1.	-dd	-d	lc1	-cl	-c
dei3y	-ied	-y	lufi4y	-iful	-y
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
ygo1	-ogy	-og	ytl2	-lty	-l
yhp1	-phy	-ph	yrtsi5	-istry	---
ymo1	-omy	-om	yra3	-ary	---

If any one of the rule matches, perform corresponding removal and replacement and perform step 2 until the minimum character length is 3.The complete 120 rules and its removal and replacements are given in the above table 4.

Example 3

Input Word = "player"

Step 1 Reverse the word ("reyalp")

Step 2 Compare last character 'r' with the 120 rules until a rule is satisfied.

If the last character of word doesn't satisfy rule, then compare last 2 characters "re" again with 120 rules until a rule is satisfied. In our example, the rule is

Rule: *re2* $\rightarrow$ *-re* Replacement: —

re2 satisfies our word *re*. *-er* denotes the deletion part, "—" denotes replacement part. Here deletion is "er" replacement is null ('—"). After stemming, the root word *play* is formed from the word "player". Stemming is used to improve retrieval effectiveness and to reduce the size of indexing files.

3. Classification of indexing techniques

The indexing process is performed by assigning each document with keywords or descriptive terms representing the document. The terms are indexed in such a way that each term reflects the content of the document to allow effective keyword searching. Indexing is an important aspect in the perspective of information retrieval as it decreases the time of searching and is source for ranking the retrieved documents. Here we have been performed detailed analysis on some popular and advanced indexing techniques called Edge Index Graph (EIG), Document Index Graph (DIG) and Inverted Index.

3.1 Edge Index Graph (EIG) Technique

The edge index graph is graph theory approach. It is a directed graph G = (V, E) where V- is a group of nodes $\{v_1, v_2 \ldots v_n\}$, each node v_i represents a content or root word in the index table. E- is a set of edges $\{e_1, e_2, \ldots, e_m\}$ such that each edge e_i has an sequence pair of nodes (v_i, v_j). There is an edge from v_i to v_j if and only if, the word v_j appears successive to the word v_i in any document as shown in fig.1 and 2. Edge index graph consist set of nodes that embody root words available in the document [1].

V1
V2
Wind
Energy

Fig. 1. Root Words

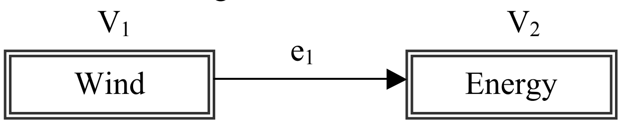


Fig. 2. Edge Index Graph

Edge index graph maintains an edge index. Edge index is very important to find the way to identify subsequent word to get a hold of the appropriate information. It contains distinct word pair appearing consecutively. They are called as the first word and next word pair. For example, the phrase "*Wind Energy*" is composed of two words:

v_1=Wind v_2=Energy

We call the root word v_1 as a first word and v_2 as successor of v_1.The edge index connects the first words and next words. Each first word is associated with a set of edge index for the next words. Hence, finding the first word gives the complete information about accessing remaining words in a particular phrase including document id, phrase of the word and position it exists in that phrase.

Edge Index of a word v is $\quad EI = \{\mathbf{d, fd, v, ed, v\ [o_i, o_i \ldots o_d], v}\}$ **(1)**

Where d is the identifier of document containing the content word v, fd, v is the number times appearance of v in d, ed, v is the identifier of root word v occurring in document d. and o_i is the location in d at which v is observed.

Now Assume that a phrase is collected from n words appears in a document consisting of the following word sequence $\{v_1, v_2, \ . \ . \ . \ , v_n\}$. The phrase is represented in the graph by drawing a path from v_1 to v_n, such that $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{n\_1}, v_n)\}$ are the edges in the graph. Each edge in the graph maintains the edge index. It stores complete detail about edge index graph and it helps in easy retrieval of information.Edge index graph construction is illustrated by constructing graph to three documents. Each document contains a mixture of phrases with more number of words and sentences which may overlap in the documents. Graph is formed by searching one document and one sentence at a time is shown in fig. 3. When a first document is considered to build a graph, it is scanned one after another phrase wise.

Node is formed to each new word and link is created if it is a successive node to preceding node. Edge index information is updated to preserves link detail between neighbor words and the root index is updated for content words. When finds next root word to attach in to the graph, it is compared with existing nodes to check whether it is already available in the graph.

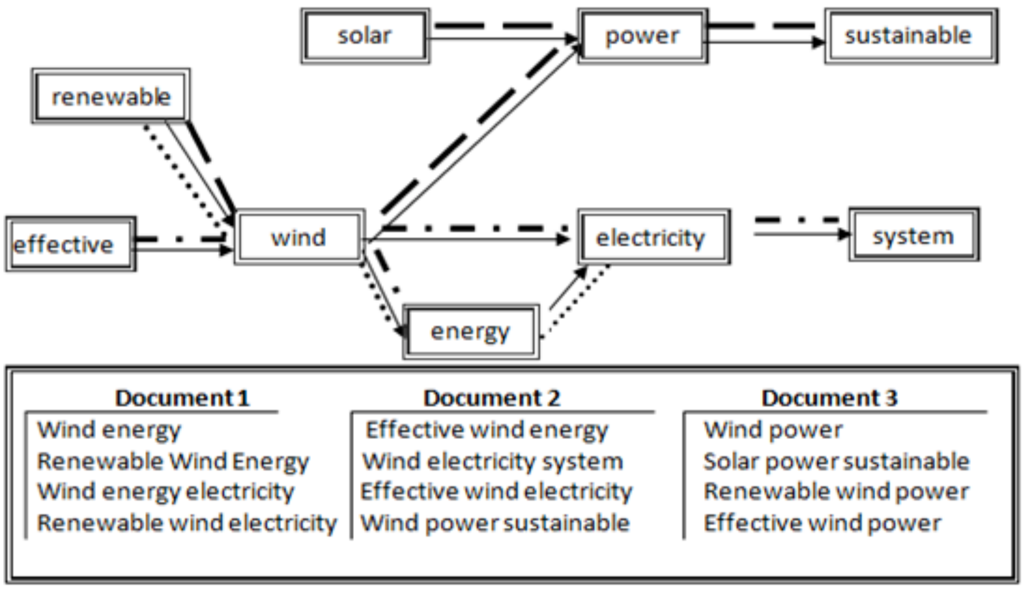


Fig. 3. Edge Index graph for Documents

3.2 Document Index Graph

The Document Index Graph (DIG) indexes the documents while maintaining the sentence structure. The DIG is a directed graph (digraph) G= (V, E) Where, V is a set of nodes $\{v_1,v_2,...,v_n\}$, where each node v represents a unique word in the entire document set; and E is a set of edges$\{e_1,e_2,... ,e_m\}$such that each edge e is an ordered pair of nodes(v_i,v_j). Edge (v_i,v_j) is from v_i to v_j, and v_j is adjacent to v_j. There will be an edge from v_i to v_j if and only if the word v_j appears subsequent to the word v_i in any document. The above definition of the graph suggests that the number of nodes in the graph is the number of unique words in the document set. Assume that a sentence of m words appearing in one document consists of the following word sequence: $\{ v_1, v_2,\ldots, v_m\}$. The sentence is represented in the graph by a path from v_1 to v_m such that $(v_1, v_2), (v_2, v_3),\ldots, (v_{m-1},v_m)$ are edges in the graph.
The Document Index Graph is mathematically shown as follows:

Document Index $\mathbf{DI = \{t_{id}, t_n, tf, s_{id}, s_n, d_{id}, sen_n, w_n, e_{id}, p\}}$ **(2)**

Where $\mathbf{t_{id}}$ is the unique id of the term in the document, $\mathbf{t_n}$ is the name of the term **tf** is the frequency of each term occurring in all the documents,$\mathbf{s_{id}}$ is the unique stem id of the respective term t_{id},$\mathbf{s_n}$ is the name of the respective stem id s_{id},$\mathbf{d_{id}}$ is the document number in which the term t_{id} occurs, $\mathbf{se_{nn}}$ is the sentence number in which the term t_{id} occurs in the document $\mathbf{w_n}$ is the word number in which the s_{id} resides in the document, $\mathbf{_{eid}}$ is the unique edge id which maintains the sentence structure,**p** is the previous term.

The representation of a term *river* in document index graph is shown in fig. 4 and document index graph is in fig.5.

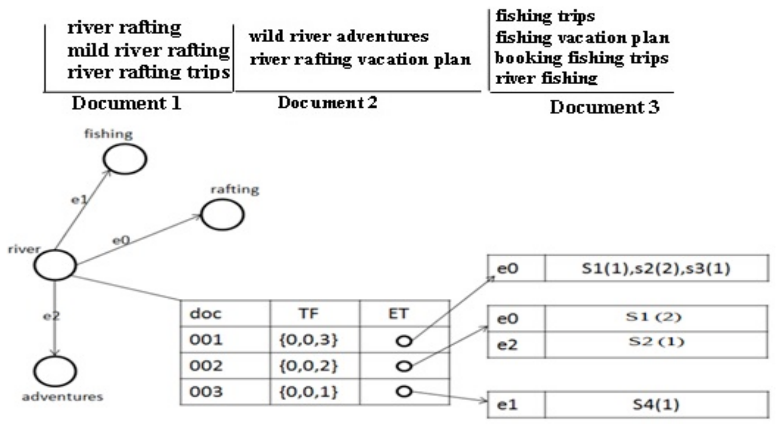


Fig. 4. DIG Structure

The DIG structure maintains two types of tables namely Document table and Edge table. Each document entry in the document table records the term frequency of the word in that document. Since the graph is directed, each node maintains a list of an outgoing edge per document entry. This list of edges indicates us which sentence proceeds along which edge. Each entry in edge table records the edge number, sentence number and word number thus maintaining the sentence structure of the document. The edge table is a list of an outgoing set of edges E_d, subset of Ev, where E_v is the list of outgoing edges that belong to node v.

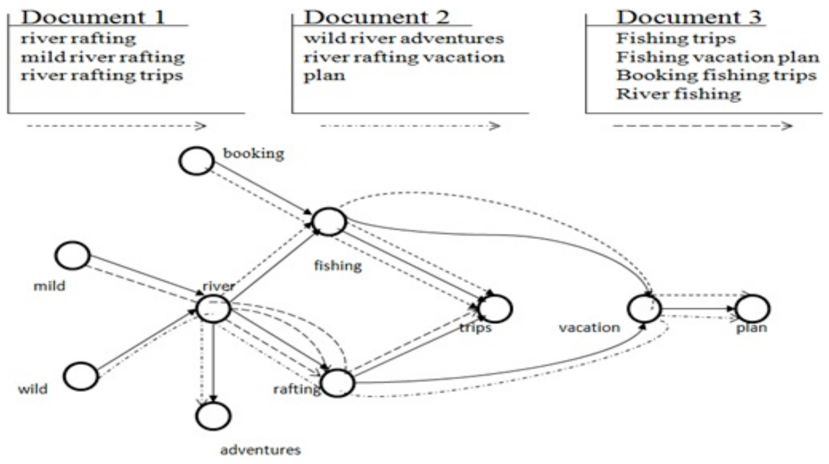


Fig. 5. Example of Document Index Graph

3.3 Inverted Index Technique

Inverted indexing technique is used to index all the root terms of the documents. The structure of inverted index contains entire detail about the root term such as root word name, the number of documents that contains the root term, document number or name that contains the root term, number of times the word occurs in a particular document, sentence number where the word occurs in a corresponding document as in fig.6. Inverted index is easy to construct.

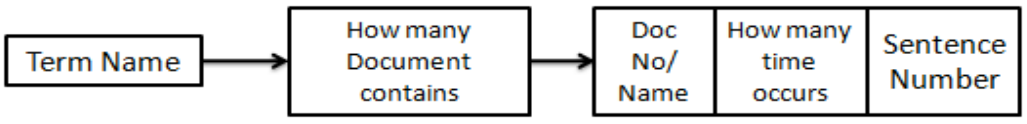


Fig. 6. Inverted Index Structure

The system process the documents one after another and also term by term. For each unique term while scanning, it creates inverted index structure and updates the whole detail about the term. If the same term appearing more than one time in the same document, its frequency will be incremented in the index structure. If the term appears more than one document, another inverted index structure will be created to the term for that document and linked with previously created inverted index structure. This process will be continued for all terms in the entire corpus.

Example 1 The indexing of two documents related to volleyball and football in fig. 7 is represented in fig. 8.

volleyball team sport teams players separated net kicking ball	foot ball refers number sports involve varying degrees kicking ball foot score

Fig. 7. Preprocessed Document

The above sample documents “volleyball.txt” and “football.txt” in figure 8 are inserted into inverted index using proposed system algorithm. The below fig. 8 shows result of inverted index.

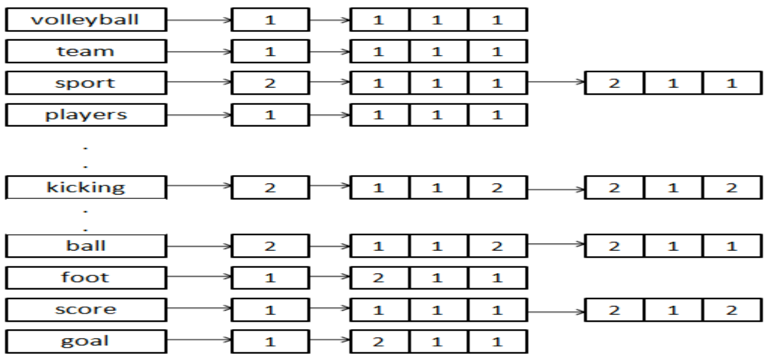


Fig. 8. Inverted Index Construction

4. Performance and evaluation

The model of the system has been developed using Net Beans as a front end tool and SQL Server and Ms Access as a backend data sources to study the working procedure and performance of all techniques. In order to support the experiment, the benchmark documents are collected from www.Wikipedia.com source and also some collections from www.uc.dataset.org to create benchmark corpus.

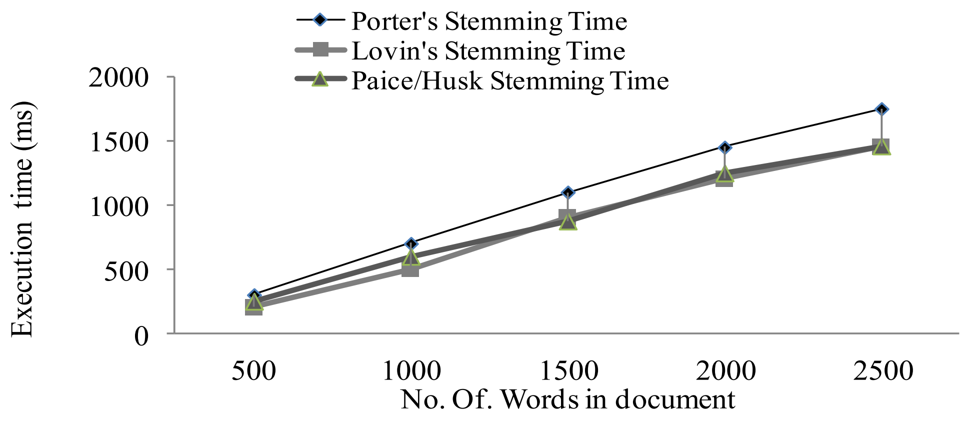


Fig. 9. Porter's, Lovin's and Paice /Husk Timing Efficiency

The performance of stemming algorithms: Porter’s stemming algorithm, Lovin’s stemming algorithm and Paice/Husk stemming algorithms are compared and shown in fig.9. The plotted graph shows the comparison between no. of words in document and execution time in milliseconds.The performance of automatic indexing approaches: Edge Index Graph (EIG), Document Index Graph (DIG) and Inverted Index techniques have been analyzed and shown in fig. 10. The plotted graph shows the comparison between number of words in document and time it takes to index.

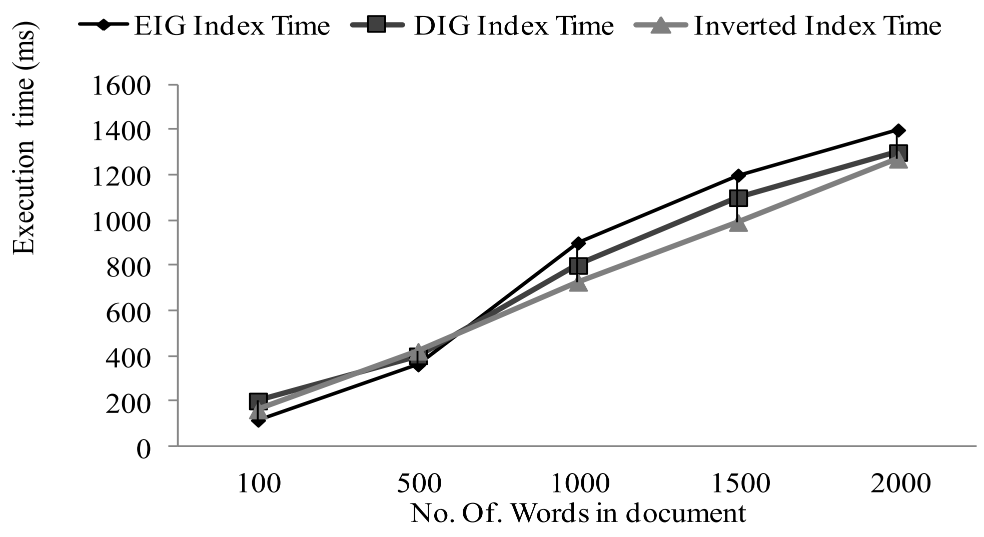


Fig. 10. EIG, DIG and Inverted Index ConstructionTime

5. Conclusion and Future work

In this paper, we studied the efficient stemming algorithms such as Porter's stemming algorithm, Lovin's stemming algorithm and Paice/Husk stemming algorithm. We have understood that how to implement the stemming algorithms and how to improve the stemming process in future. Then, we discussed the indexing types and their advances. Then we have been analyzed the automatic indexing techniques called as Edge Index Graph (EIG), Document Index Graph (DIG) and Inverted Index and their performance with suitable examples. This analysis work is very helpful to improve the existing techniques and to create new techniques in future.

In future work mainly focus on stemming process, either to create a new stemming algorithm or to improve the existing one with effective manner. Also try to improve the searching process or mapping time for the user query with indexed term and to reduce the indexing time.

References

1. R.Dhanapal, "IntelligentInformationRetrievalAgent", *Knowledge Based Systems*, Vol. 21, pp.466-470, 2008.
2. Ms. Anjali Ganesh Jivani (2011), "A Comparative Study of Stemming Algorithms", *International Journal of Computer Technology and Applications*, Vol. 2 , pp.1930-1938.

3. A N K Zaman and Charles Grant Brown, **"**Latent Semantic Indexing and Large Dataset: Study of Term-Weighting Schemes**",** *ICDIM Conference,* Lakehead University, Canada, pp.1-4, 2010.
4. Kolikipogu Ramakrishna and B. Padmaja Rani, "Study of Indexing Techniques to Improve the Performance of Information Retrieval in Telugu Language", *International Journal of Emerging Technology and Advanced Engineering*, Vol. 3(1), pp.1-9, 2013.
5. M. F. Porter , "An algorithm for suffix stripping", *Proceed. Computer Laboratory*, Corn exchange Street, Cambridge, Vol.1, pp.11-19, 1980.
6. Khaled M. Hammouda and Mohamed S. Kamel (2002),"Phrase-based Document Similarity based on an Index Graph Model", *International conference on Data Mining*, Ontario, Canada, pp.203-210, 2002.
7. Julie Beth Lovins , "Development of a Stemming Algorithm", *Mechanical Translation and Computational Linguistics*, Vol.11, Nos.1 and 2, pp.22-31, 1968.
8. Khaled M.Hammouda and Mohamed S. Kamel , "Efficient Phrase-Based Document Indexing for Web Document Clustering ",*Knowledge and Data Engineering*, Vol. 16, no. 10, pp.1279-1296, 2004.
9. Daniel Osuna-Ontiveros, Ivan Lopez-Arevalo and Victor Sosa-Sosa , "A topic based indexing approach for searching in documents", *Electrical Engineering Computing Science and Automatic Control,* International Conference Proceedings, Merida City,pp.1-6, 2011.
10. Carlo Abi Chahine et al , "Conceptual Indexing of Documents Using Wikipedia", *Web Intelligence and Intelligent Agent Technology, ACM International Conference*, Lyon, France, Vol. 1., pp.195-202, 2011.
11. Zhisheng Li et al (2011), "An Efficient Index for Geographic Document Search", *Knowledge and Data Engineering*, Vol. 23, No. 4, pp.585-599.
12. Wee Keong Ng, Yonggang Wen and Huafei Zhu, "Private Query Processing on Inverted Index", *Knowledge and Data Engineering*, Vol. 2, pp.1-5, 2011.
13. Syed Raza Ali Rizvi and Shawn Xiong Wang (2011), "Using Semantic and Structural Similarities for Indexing and Searching Scientific Papers", *Computer Science and Automation Engineering,* Vol. 26, pp.1-5, 2011.
14. B.Barla Cambazoglu and Cevdet Aykanat," Performance of query processing implementations in ranking-based text retrieval systems using inverted indices", *Information Processing and Management*, Vol. 42, pp.875–898, 2006.
15. Jia-Ling Koh Nonhlanhla Shongwe and Chung-Wen Cho, "A Multi-level Hierarchical Index Structure for Supporting Efficient Similarity Search on Tag Sets " , *Research Challenges in Information Science*, Vol. 12, pp.1-12, 2011.