Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 3, March 2015, pg.613 – 617

RESEARCH ARTICLE

METICULOUS MINER

Ojas Sawant, Ninad Agashe, Sushant Chopda, Mr. Dilip Dalgade

Department of Information Technology, MCT's Rajiv Gandhi Institute of Technology, University of Mumbai, Mumbai, Maharashtra, India

ojasvs@gmail.com; itsninad93@gmail.com; chopdasushant@gmail.com; dilip8504@gmail.com

Abstract— Generally, any search engine retrieves the information using many algorithms like Page Rank, Distance vector algorithm, crawling, etc. by using the user's query. Sometimes it so happens that the links extracted by search engines may or may not be relevant to the user's query and user has to examine all the extracted links to know whether the relevant information is present in the particular page or not. It becomes a tedious and time consuming job for the user. Our focus is to fetch different web pages based on text similarities. Our proposed tool 'Meticulous Miner' which is based on the concept of sub search mining , which uses top-k pages extracted from Google search and mining the pages based on Term frequency & web document frequency for the results. When the user feeds a query our proposed application anticipates possible secondary queries, and allows the user to access the search response of the original query fed and these additional queries simultaneously. Hence our proposed application will help users to get optimal result without a hassle.

Keywords— Text Mining, Keyword Extraction, Sub Search, Users, TF-IDF

I. INTRODUCTION

Meticulous Miner is a tool which uses the concept of sub search mining. Search engine extracts various links relevant to the search query fed by the user. Sometimes it may happen that the user is not getting the satisfying results from the search engine. To get the satisfactory result it is required that user knows the effective querying techniques, which every user may not be aware of. Here the concept of sub search comes in the picture; it takes the query from the user and extracts result which is precise to the user's need. A user requesting data on a desired subject or item would otherwise have to go through hundreds of pages to find the most relevant information to his query. Hundreds of results through use of mining text are reduced by this application. Our application Meticulous Miner drags the focus to relevant web pages and gives refined results on them and hence, proves itself as a very useful tool for the user to get the desired result in the fastest possible means .

II. PROPOSED SYSTEM

The Components of system are:- Text Mining, Algorithm used, Keyword Extraction

Keyword Extraction

The Keyword are extracted Based on the Primary and Secondary Search Primary Search Filter outs the URL from the web and display in the form of Html format. The tool extracts the search results from either of the search engines [Google or yahoo]. It extracts the top-k pages from the web & extracts the contents of these pages & clusters them in the database.[4]

Text Mining

Text mining, roughly equivalent to text analysis, refers to the process of Extracting highly relevant information from text. This information is typically extracted by usage of patterns and trends through means such as statistical pattern analysis.[2]

Algorithm used

TF-IDF Algorithm is used on text documents to get some relevant information from them. In web Documents Text is not structured properly. It is not the only algorithm for the text documents, but it is usually considered as an optimal algorithm for clustering and classification methods. Our tool Meticulous Miner will find frequency of that keyword in each web page using 'term frequency inverse document frequency'. As the term implies, TF-IDF calculates values for each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of web page the word appears in. Words with high TF-IDF numbers imply a strong relationship with the web page they are included in, suggesting that if that word were to appear in a web page, that web page could be of interest to the user. So its proved that this simple algorithm efficiently categorizes relevant keywords that can enhance result retrieval[1][5]

Tf–idf is the product of two statistics, term frequency and inverse document frequency. Various ways for determining the exact values of both statistics exist. In the case of the **term frequency** tf(*t*,*d*), the simplest choice is to use the *raw frequency* of a term in a document, i.e. the number of times that term *t* occurs in document *d*. If we denote the raw frequency of *t* by f(*t*,*d*), then the simple tf scheme is tf(*t*,*d*) = f(*t*,*d*). Other possibilities include Boolean "frequencies": tf(*t*,*d*) = 1 if *t* occurs in *d* and 0 otherwise;
logarithmically scaled frequency: tf(*t*,*d*) = 1 + log f(*t*,*d*), or zero if f(*t*, *d*) is zero;
augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document:

$$\mathrm{tf}(t,d) = 0.5 + \frac{0.5 \times \mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

With
N: total number of documents in the corpus
$|\{d \in D : t \in d\}|$: Number of documents where the term t appears (i.e., $\mathrm{tf}(t,d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.
Mathematically the base of the log function does not matter and constitutes a constant multiplicative factor towards the overall result.
Then tf–idf is calculated as

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf-idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf-idf closer to 0.[1]

Psuedo Code

```
def tf(word, blob):
    return blob.words.count(word) / len(blob.words)
def n_containing(word, bloblist):
    return sum(1 for blob in bloblist if word in blob)
def idf(word, bloblist):
    return math.log(len(bloblist) / (1 + n_containing(word, bloblist)))
def tfidf(word, blob, bloblist):
    return tf(word, blob) * idf(word, bloblist)
```

1)tf(word, blob) computes "term frequency" which is the number of times a word appears in a document blob, normalized by dividing by the total number of words in blob. We use TextBlob for breaking up the text into words and getting the word counts.
2)n_containing(word, bloblist) returns the number of documents containing "word". A generator expression is passed to the sum() function.
3)idf(word, bloblist) computes "inverse document frequency" which measures how common a word is among all documents in bloblist. The more common a word is, the lower its idf. We take the ratio of the total number of documents to the number of documents containing word, then take the log of that. Add 1 to the divisor to prevent division by zero.
4)tfidf(word, blob, bloblist) computes the TF-IDF score. It is simply the product of tf and idf.

III. IMPLEMENTATION

System Architecture

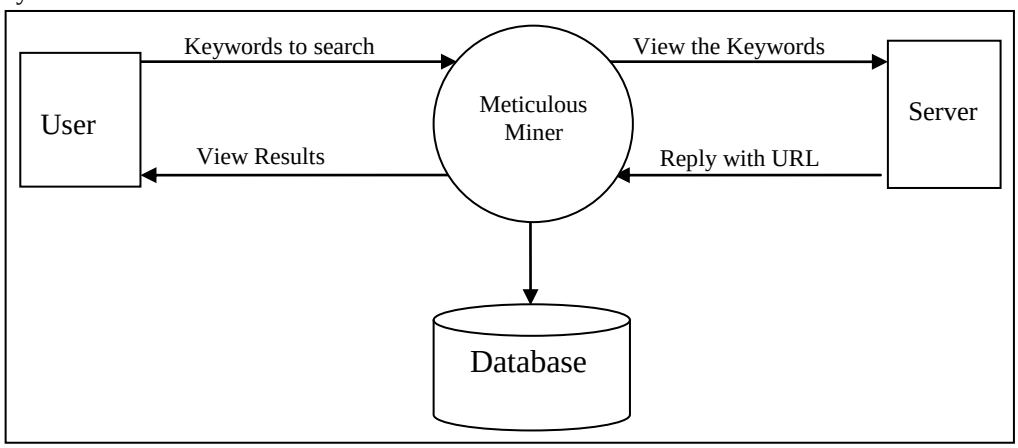

Fig. 1 Component

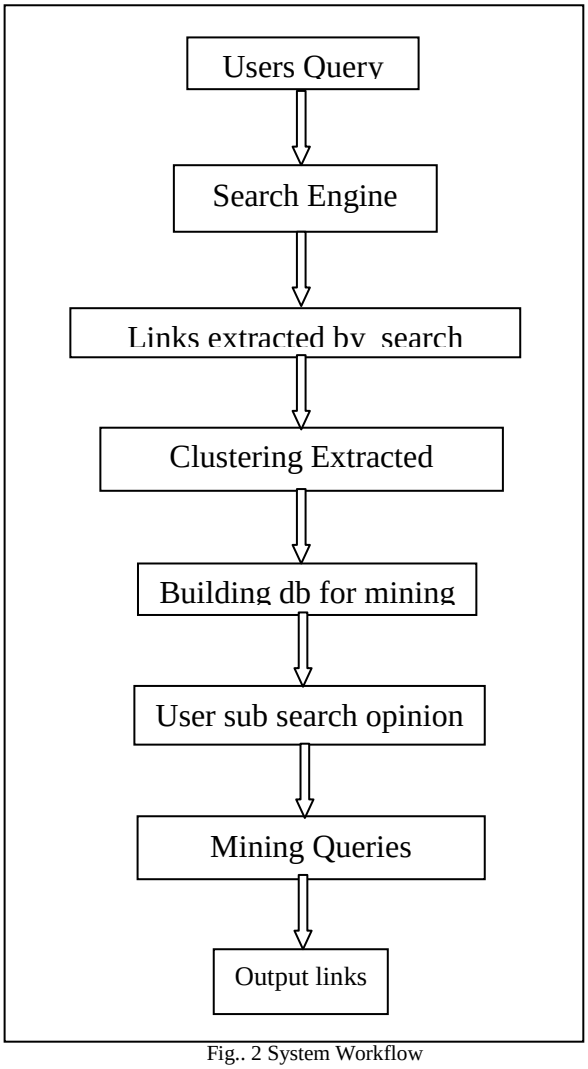

Fig.. 2 System Workflow

Application basically consists of one window. That window contains a listbox which is used for displaying the extracted relevant links. It also contains URL panel which is used for querying purposes. It also contains two extra search panel for implementing sub search technique.

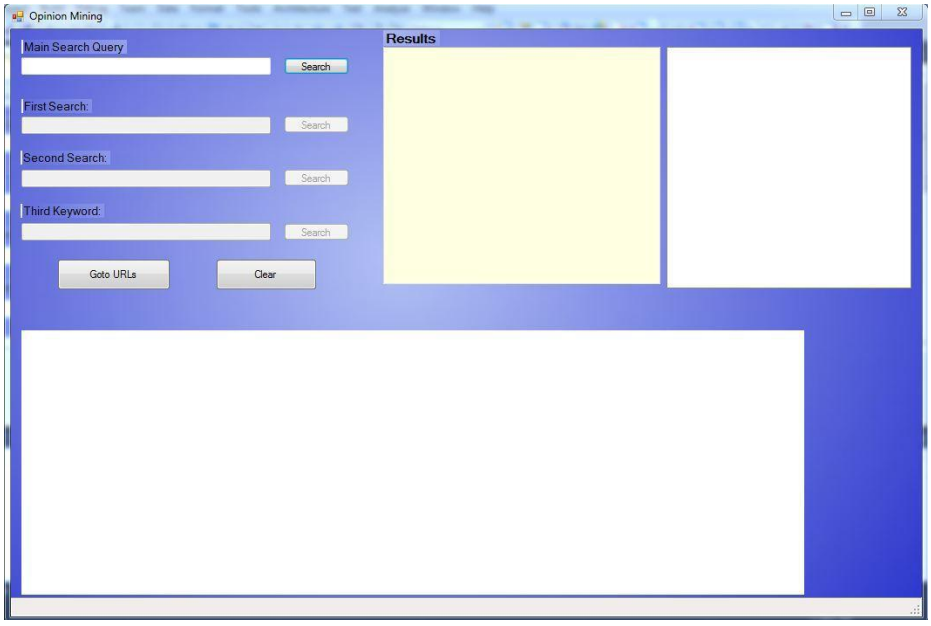

Fig. 3 System screenshot

IV. CONCLUSIONS

In this research paper, we have used TF-IDF Algorithm to efficiently produce optimal search results This will help users in obtaining the desired information in an efficient and fast manner. This application is suitable for the masses as it is search optimizing system that not only help improve efficiency and convenience it also enhance productivity and faster turnaround time

V. ACKNOWLEDGEMENT

We wish to express our sincere gratitude to Dr. U. V. Bhosle, Principal and Prof. D.M.Dalgade, H.O.D of Information Technology Department of RGIT for providing us an opportunity to do our project work on "**Meticulous Miner**". This project bears on imprint of many people. We sincerely thank our project guide Prof. Dilip Dalgade for his guidance and encouragement in successful completion of our project synopsis. We would also like to thank our staff members for their help in carrying out this project work. Finally, we would like to thank our colleagues and friends who helped us in completing the project synopsis successfully.

REFERENCES

[1] Juan Ramos ,Using TF-IDF to Determine Word Relevance in Document Queries
[2] http://en.wikipedia.org/wiki/Text_mining
[3] Reiner Kraft, Chi Chao Chang, Farzin Maghoul, Ravi Kumar , Searching with Context
[4] Francisco Corella , Karen Lewison Searching the Web More Effectively With Multiple Simultaneous Queries
[5] http://en.wikipedia.org/wiki/Tf%E2%80%93idf

