Information and Knowledge Management
ISSN 2224-5758 (Paper) ISSN 2224-896X (Online)
Vol.4, No.4, 2014

The Super Pages Search Engine

Amelina Ahmeti
Agriculture University of Tirana ,Koder Kamez Tirana Albania
* E-mail of the corresponding author: amelina.ahmeti@gmail.com;

Abstract

With the huge growth of websites, the people find difficult to go to every page and waste of time reading every site. A query is formed by the user with a set of keywords that describe their item of interest, and then the search engine returns the pages that include that set of keywords. If a user wants to find information about a topic, the search engine Google will not help much. The user will have to read the pages that have been returned by the search engine, find related concepts and then issue new queries. For instance, we need to find information about the current economic problems in Europe. A query will return pages on that but not many on the Greek, Italian, Portuguese, and Irish issues specifically. The information returned is too generalized. The user has to form new specific queries to get that in- formation. So a search engine is needed which instead of returning pages in some simple flat list, returns the information in some more structured way. The pages that have relationships between elements that they contain or similarities of some kind should represent one group of links or one category. All these ideas are implemented in" The SUPER PAGES search engine" program .This report describes how the program is done, the experiments, the algorithm is used and the problems encountered during the work.
Keywords: Jaccard, Super search engine, index, stop words, similarity

1. Introduction

Internet is widely used nowadays by many people who search different information through it. The main search engine that the people use the most is Google (www.google.com).The number of links that it returns for a particular search is high, so there is a need for a program that can not only search the required information but also present it in a structured way. SUPER PAGE search engine takes your query and related with the query introduces some categories which can be selected by you and shows the group of documents belonging to each category. Each category is a cluster of documents which have similarities with each other. In order to create this program, Wikipedia and also bing.com was used. The number of documents that are to be compared is too many, so there is a need for an efficient and effective algorithm that compares the similarity of the documents. First, the user is introduced with a web page where he enters his query. Then the query is searched on Wikipedia. In the document that is returned by Wikipedia we remove all stop words like: if, then, else etc. which are stored in a table in our database as common words. After these words are removed then the remaining words (in our case lets distinguish them as an A set) which create a first set of words, are searched again in Wikipedia. The stop words should also be removed from all the documents that are returned from the second search. Every document that is returned from the second search is compared with that returned by the first search, to check if there is any similarity between them. In our case every document from the second search with its words creates a set (defined as B set the collection of remaining word for each document). In order to create structured information, the result documents should be compared with the first set of the user query. So there is a need of an algorithm which com- pares every element of set B with the set A. The algorithm that is used in our case is Jacquard similarity algorithm. After the program checks the comparison between the documents, it gives the results which are introduced in the website

2. Relevant Work

The delicate problem of similarity sites has attracted the attention of the researchers during the last years. In this section we will do a brief description of the current state of the topic we are discussing. The computer scientists have given a lot of efforts in creating a fast and usable web ser- vice. According to their research on this problem they have related this issue with three problems: text document matching, schema matching, and software component matching.

Text document matching: Document matching and classification is a long-standing problem in information retrieval (IR).The solutions are based on term frequency but this approach is not sufficient because text documentations for web-service operations are highly compact, and they ignore structure information that aids capturing the underlying semantics of the operations.

Schema matching: This is a discussion for database community who are trying to solve it. The work in this area has developed several methods that try to capture clues about the semantics of the schemas, and suggest

matches based on them. Such methods include linguistic analysis, structural analysis, the use of domain knowledge and previous matching experience. However, the search for similar web-service operations differs from schema matching in two significant ways. First, the granularity of the search is different: operation matching can be compared to finding a similar schema, while schema matching looks for similar components in two given schemas that are assumed to be related. Second, the operations in a web service are typically much more loosely related to each other than are tables in a schema, and each web service in isolation has much less information than a schema. Hence, the programmers are unable to adapt techniques for schema matching to this context.

Software component matching: Software component matching is considered important for software reuse. It defines the problem by examining signature (data type) matching and specification (program behavior) matching. The techniques employed there require analysis of data types and post-conditions, which are not available for web services. According to these problems different approaches have been evolved by using different similarities algorithm for example one of this algorithm is Ontology-Driven Similarity Algorithm .While we will introduce our approach based on Jacquard algorithm.

3. The details of Implementation

We have used Visual studio 2010 as a platform. The project is a web service on asp.net framework 4.0 whiles the database on SQL Server 2008 to implement our project.

3.1 The first Steps

There were technical and conceptual contradictions.

3.1.1 Problem Definition

Our problem is related with semantic similarity which is a concept whereby a set of documents or terms within term lists are assigned a metric based on the likeness of their meaning or semantic content. So we are interested in creating a web service which will analyze the similarity of pages and give the results based on their similarities .Each group of documents which have similarity semantic between each other will be represented in one group.

3.1.2 Technical problems

In the beginning ,we thought that after calculating the similarity index between the words of Wikipedia ,the results could be retrieved from xml custom search of Google by grouping them .There was a difficulty in implementing this because the free versions of xml custom Google need to have at least one domain in which adds by Google should be included .In another step we thought not to use xml but the searches would be included in a frame that had the target Google .This was also a problem because Google prevents the inclusion in a frame for security reasons .So in the end, we decided bing xml search (bing-service) to group the results . Another difficulty which is still not being solved is the database with stop words which contains around 500000 records .This effects on slow speed during the execution. We could not solve this problem as linguistics is needed to structure these words.

3.2 Algorithm

Jacquard similarity algorithm is used in this project. The Jacquard similarity coefficient is a measure of similarity between sample sets. For two sets, it is defined as the cardinality of their intersection divided by the cardinality of their union. Mathematically,

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In the case of Wikipedia data, a slightly different approach is required. For a pair of entities, the calculation is performed on sets of entities with which the elements of the pair occur. So in our case the set A is the set of words from Wikipedia without the stop words from the first search. While the set B is the set of remaining words again from Wikipedia from the second search for each document that is found. So we find the Jacquard index for B set and the first set A. The greater is the Jacquard Index the higher is the similarity. In our project we can define ourselves the Jacquard Index and it will give all the categories that have the Jacquard Index greater that what we wanted. Each category represents a group of documents which have the Jacquard index greater that we put in the text box. The acceptable values for Jacquard index

are greater than 0 and smaller than 1.

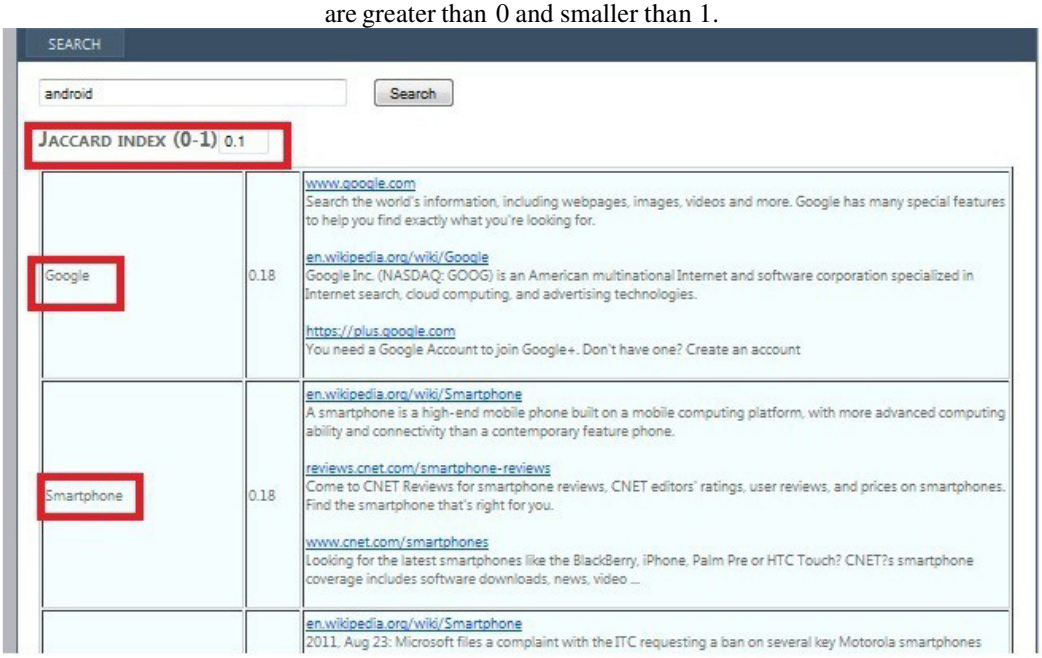


Figure 1: A result of execution.

In the upper red-lined box at Figure.1 we can put the Jacquard index and in the down red-lined boxes, All the categories are presented by Jacquard index greater than we put. So, every row or box represents a category in our case. In order to understand the calculation of Jacquard index better, let's suppose we have two sets:

A= (android, Smartphone) and B= (phone, touch screen, android)

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In our case for the actual A and B, the Jacquard Index is = 1/(3+2-1)= 0.25. Example 2:
Suppose we have two sets A=(7,3,2,4,1) and B=(4,1,9,7,5). Then the union:

$$|A \cup B| = (7, 3, 2, 4, 1, 9, 7, 5)$$

And the intersection is:

$$|A \cap B| = (7, 4, 1)$$

Jacquard's coefficient can be computed based on the number of elements in the intersection set divided by the number of elements in the union set:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = 3/7 = 0.429$$

We should have into considerations that the calculation of this formula is based on mathematical concepts:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$= \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

According to this formula of the algorithm, all the jacquard indexes are calculated. We describe in pseudo code this algorithm as follows in a simple way:

search the query in the web with user input parameter
put resulted words in the vector v1 removing stop words input minjaccardindex
for each word in the vector v1

search the query in the web with word from v1 parameter put resulted words in the vector v2 removing stop words
set union to v1 size + v2 size set intersection to 0
for each word in the vector v2
if word is in v1 then
set intersection incremented to 1 endif
end for
set
if then

$Jaccardindex = intersection/union-intersection$
$jaccardindex \geq minjaccardindex$

print result endif
endfor

3.2.1 The reasons of using Jacquard algorithm

1-Jaccard Similarity is very easy to calculate.
2-It also suits well in similarity situation as it is only practical to compare search sessions based on their contents and not all the possible keywords and service descriptions that exist in the entire search engine.
3-It gives the opportunity to compare two sets which we treat as vector sets and also we can adjust the coefficient in order to have similarity according to our desire.

3.3 Database

The database is consisted of five tables which are: common- words, results, finalresult, searchquery and finales.

1-The common words contains all stop words like: if, that, while etc which we have downloaded from ubuntu Linux
2-The table results contains all the words after the search. The word returned from the user search and all the success- save searches done after the first search.
3-Finalresult is used for the results of the first search. This table helps us to make the next searches after the first search removing duplicate words. Also has the methods for the intersection and union of words in the text that use the jacquard algorithm.
4-Searchquery retrieves the top 15 records for the next search after deleting stop words. (a limit used for speed up the results)
5-The finales contains results after applying the jacquard similarity algorithm.

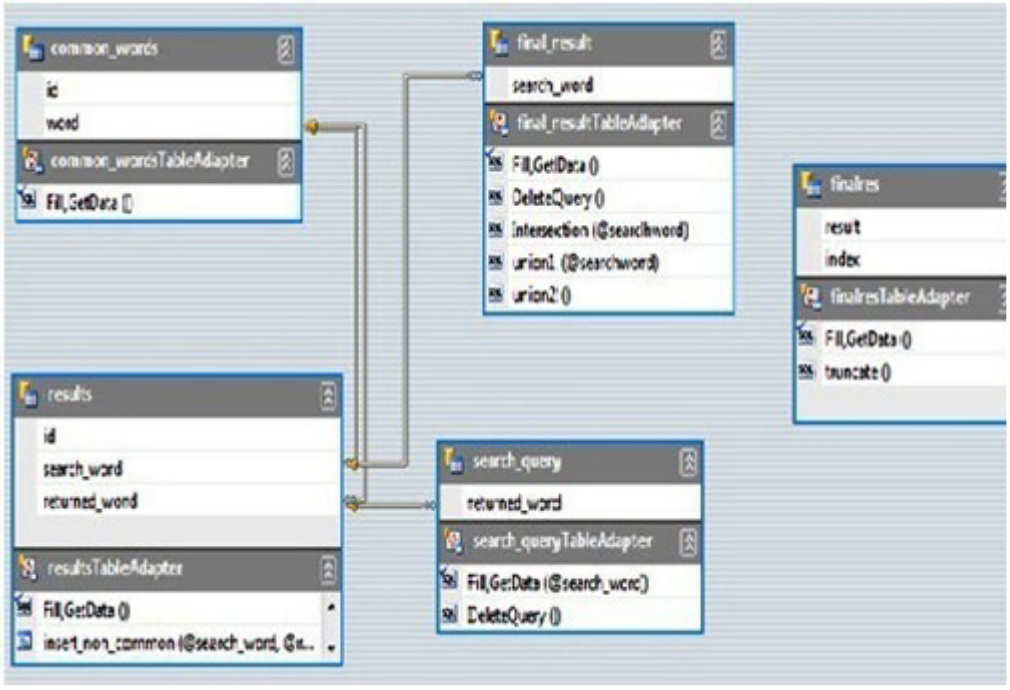


Figure 2: The database that is used in our project

So in this schema we summarize all the phases that we have performed for this project:

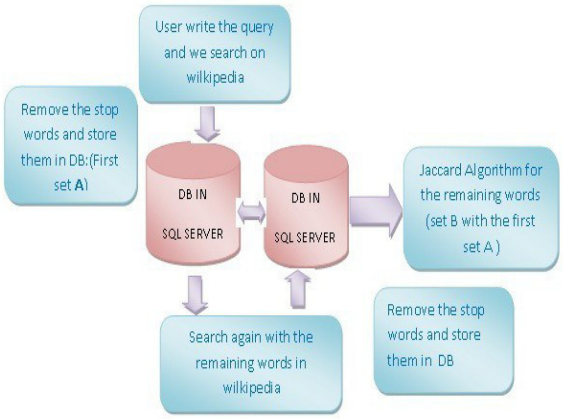


Figure 3: The phases of performance

3.3.1 Restrictions

There are some restrictions that we have used in order to have a faster web application:

1- searchquery retrieves the top 15 records for the next search after deleting stop words.

2- The other restriction is that there is a need of 30 seconds until 1 minute for the program to be loaded.

3- The speed is also depended on Network conditions.

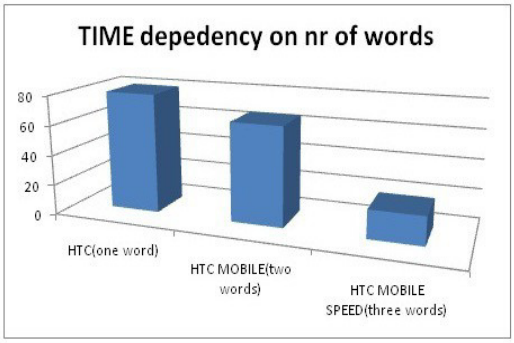
4- Also if the Jacquard index is high then the search needs more time.

3.4 Experiment Results

Although this project is implemented, there are problems with it. The first and obvious problem is the speed. The time of execution differs from 30 sec until to one minute. The speed of searching the categories for different words is different. There is a table of the speeds for different words while the Jacquard index that we put is 0.05:

Word	TIME
HTC	80 sec
TRENTO	50 sec
HTC MOBILE	66 sec
TRENTO UNIVERSITY	35 sec
HTC MOBILE SPEED	19 sec
TRENTO RESTAURANTS CENTER	15

Figure 4: The result of first experiment



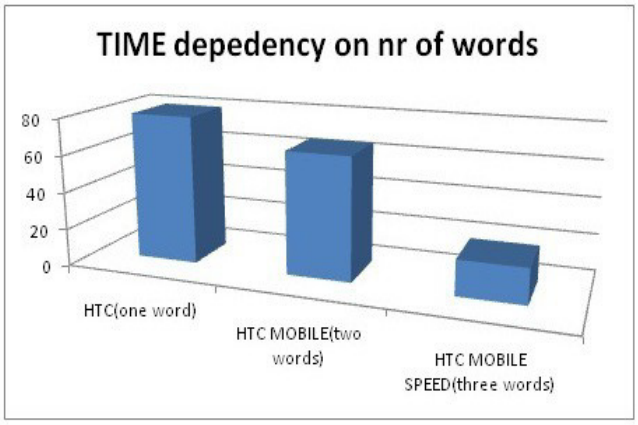


Figure 5: The Graphical presentation of HTC word search depending on time

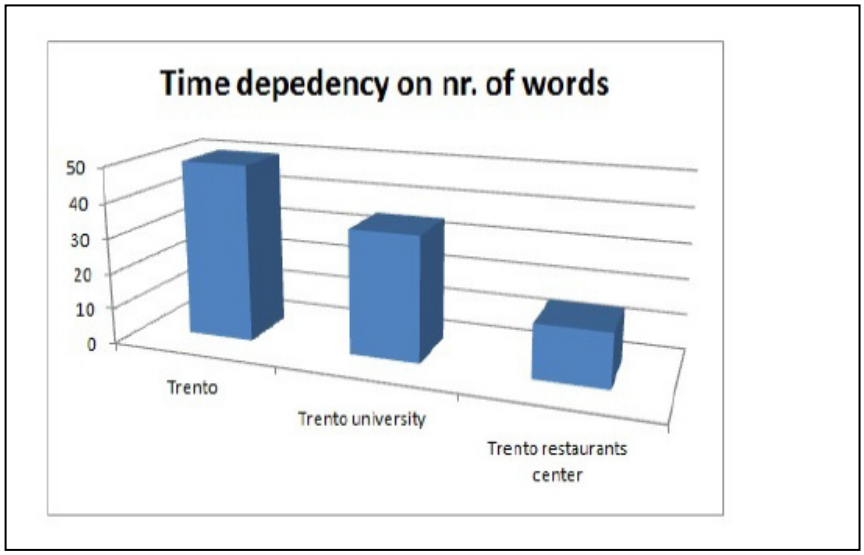


Figure 6: The Graphical presentation of Trento word search depending on time

If the Jacquard index is increased then the results change as it is shown in Figure nr.7.The results seem to be higher according to the graphical presentation.

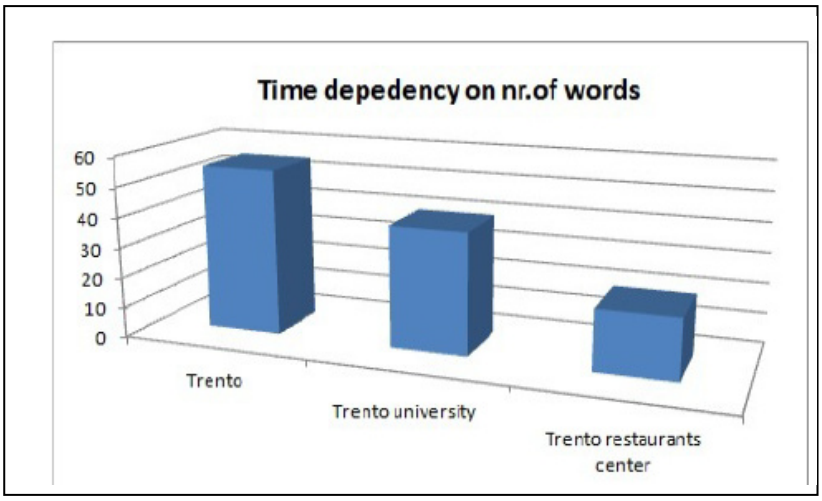


Figure 7: The result after changing the Jacquard index from user

Another factor that slows down the speed of search is the condition of internet .If network is in good conditions then the speed is higher .So there are a lot of factors that effect on the speed of the application.

3.4.1 The dependency on Internet conditions

One major factor that effects on speed application is the speed of internet .It is the major factor for having fast results and showing them after the execution .In the table below we show the results when the internet is fast with blue line and with red line the bad internet conditions.

Word	TIME on good internet	time on (bad internet)
HTC(one word)	80	88
HTC MOBILE(two words)	66	70
HTC MOBILE SPEED(three words)	19	25

Figure 8: The result depending on Internet conditions

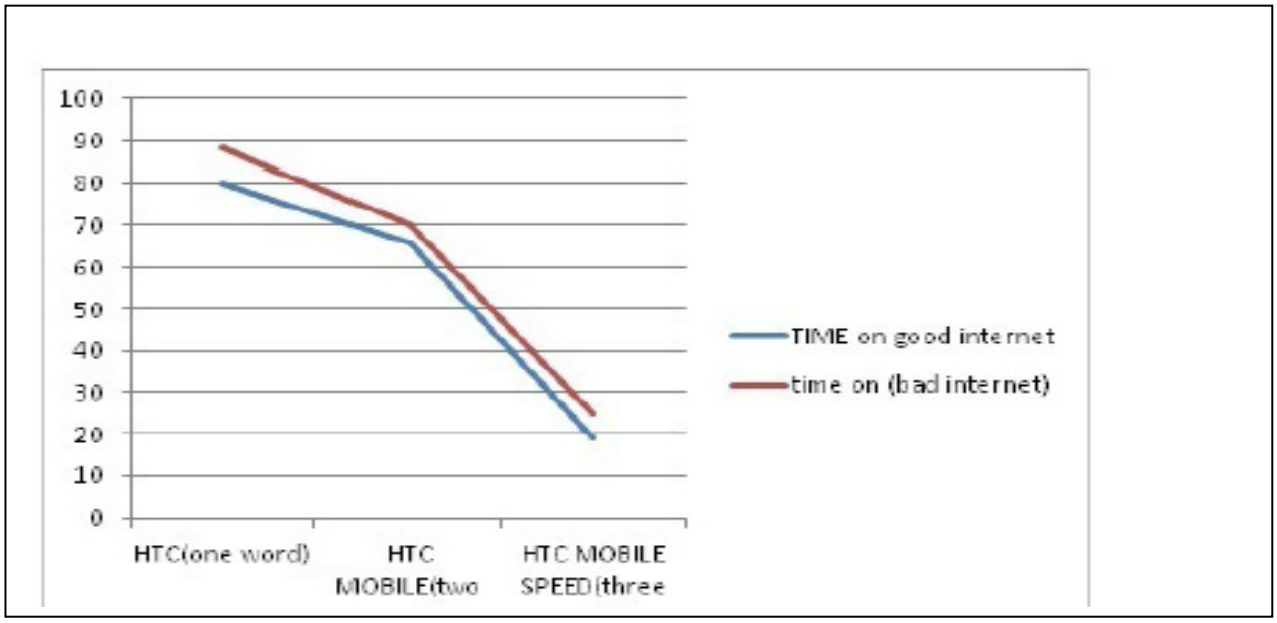


Figure 9: The dependency of time on words and internet

5. Conclusion

This work represents only a small first step in creating a web application of structured information according to user query .In this project we tried to approach the problem of structuring the similar pages .We tried to create a web pro- gram which doesn't give just the links related to query user but also grouping them based on their similarity .We used Jacquard algorithm for similarity during the program to show the related pages in one group which we called a category.This is a complicated problem in which a lot of people are working on .We created the program ,tested it and collected the results . During our work we discovered that many factors effect on the results which brings problems on the execution of our application .First is the high amount of data that we have to compare then the number of words, Jacquard Index and also the internet conditions. We estimate our results as relevant and satisfying in many cases, but there is still further work to do for refining them and considering other approaches that could improve the accuracy.

References

[1] http://www.vldb.org/conf/2004/RS10P1.PDF
[2] http://en.wikipedia.org/wiki/Jaccard index
[3] ww.bing.com
[4] www.wilkipedia.com
[5] The paper from http://weblab.infosci.cornell.edu/papers/Bank2008.pdf
[6]http://sujitpal.blogspot.com/2008/09/ir-

math-with-java- similarity-measures.html
[7] http://en.wikipedia.org/wiki/Semantic similarity
[8] http://wiki.commerce.net/images/9/9c/CN-TR-06-02.pdf
[9] http://kmi.open.ac.uk/publications/pdf/kmi-04-16.pdf
[10]http://people.revoledu.com/kardi/tutorial/Similarity/Jaccard.html
[11] http://kmi.open.ac.uk/publications/pdf/kmi-04-16.pdf
[12] http://www.vldb.org/conf/2004/RS10P1.PDF
[13] http://www.cse.unsw.edu.au/ weiw/_les/ICDE09-TopKSimJoin-Final.pdf

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Recent conferences: http://www.iiste.org/conference/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar