TOPIC SENSITIVE PAGERANK FOR SEARCH ENGINES

Report submitted to the SASTRA Deemed to be University as the requirement for the course

BCSCCS701R03: BIG DATA ANALYTICS

Submitted by

NIKHIL KUMAR REDDY B
(Reg. No.: 221003068, B.TECH-CSE)

February 2021

SRINIVASA RAMANUJAN CENTRE
KUMBAKONAM – 612 001

SRINIVASA RAMANUJAN CENTRE
KUMBAKONAM – 612 001

Bonafide Certificate

This is to certify that the report titled "**Topic Sensitive PageRank For Search Engines**" submitted as a requirement for the course, **BCSCCS701R03: BIG DATA ANALYTICS** for B.Tech. is a bonafide record of the work done by **Shri. Nikhil Kumar Reddy B (Reg. No.221003068, B.TECH-CSE**) during the academic year 2020-21, in the School of Computing

Project Based Work *Viva voc*e held on 17/02/2021

Examiner 1 **Examiner 2**

LIST OF FIGURES

Figure No.	**Title**	**Page No.**
1.1.1	Link structure of web	2
1.1.2	Inlinks and Outlinks	3
1.1.3	A loop with spider trap	3
1.4.1	Representation of TSPR	5
3.1	Data of Inlinks used	10
3.2	Example topics used	10
3.3	Output showing links with top ranks	11

LIST OF TABLES

Table No.	Table name	Page No.
1.1.1	Inlinks and Outlinks	3

ABBREVATIONS

ODP	Open Directory Path
SEO	Search Engine Optimization
TSPR	Topic Sensitive PageRank
URL	Uniform Resource Locator

NOTATIONS

Greek Symbols (in alphabetical order)

$\sum$	Summation
$\in$	Belong to

Miscellaneous Symbols (in alphabetical order)

$\mid x \mid$	Absolute value of x

ABSTRACT

Search engines are very useful tool now a days to fulfill the information need of a user. The performance of search engine mainly depends on ‘Page Ranking Algorithm’ which provides highly relevant web pages at the top of the search result. Hence for providing the efficient search we use ‘Topic Sensitive page Rank’, which will provide user with the most relevant URLs based on the topic that he is searching for.

The aim of this project is to improve the ranking of search-query results computes a single vector, using the link structure of the Web, to capture the relative importance of Web pages. Topic-sensitive PageRank uses a non-uniform personalization vector, not simply a post-processing step of the PageRank computation. Personalization vector introduces bias in all iterations of the iterative computation of the PageRank vector. Hence more accurate search results will be produced with this Topic Sensitive Page Rank algorithm.

KEY WORDS: URL, PageRank, Personalization vector, Link structure

Table of Contents

Title	Page No.
Bonafide Certificate	ii
List of Figures	iii
List of Tables	iv
Abbreviations	v
Notations	vi
Abstract	vii
1. Introduction, Merits and Demerits of the work	1
2. Source Code	8
3. Snapshots	11
4. Conclusion and Future Plans	13
5. References	14

1. INTRODUCTION, MERITS AND DEMERITS OF WORK

1.1 INTRODUCTION

We all know that the World Wide Web is growing rapidly. There are more than 100 million websites and more than 10 billion pages over there. The recent growth of the Internet and the World Wide Web makes it appear that the world is witnessing the arrival of a completely new technology. In fact, the Web is now considered to be a major driver of the way society accesses and views information.

To retrieve the information, one has to search it on the web, and it provides the pages that has the content you searched. For example, if we type the word “cricket” inside Google which is the most popular search engine, we will end up with around 456 million results. But the following questions are raised which are need to be answered.

- Other search engines will yield more or less different results. Why?
- What makes the foundation of the search engine?
- Why do we prefer one search engine over another?

Step back in time ten years that is to around 2010, and PageRank was the SEO (Search Engine Optimization) metric that everyone talked about. If you were working in the industry for more than a few years, you will definitely remember the excitement that came when you heard that there had been an update to the PageRank toolbar. With any luck, your efforts would have delivered an increase in your PageRank score, knowing that this meant that Google was now viewing your site as more authoritative than it previously was. An increase in your PageRank score was a great demonstrator that your SEO strategy was working. Fast forward to 2021, and PageRank is rarely mentioned. But that is not because it is no longer important, just that it is no longer a public-facing metric.

What is a PageRank?

PageRank is a system for ranking web pages that Google's founders Larry Page and Sergey Brin developed at Stanford University. And what it is important to understand is that PageRank is all about links. The higher the PageRank of a link, the more authoritative it is. We can simplify the PageRank algorithm to describe it as a way for the importance of a webpage to be measured by analysing the quantity and quality of the links that point to it.

PageRank can also be defined as a function that assigns a real number to each page in the web (or at least to that portion of the web that has been crawled and its links discovered). A method for rating the importantance of web pages objectively and mechanically using the link structure of web.

PageRank uses a random surfer model. It represents the likelihood that a person randomly clicking on links will arrive at any particular page. Probability distribution is evenly divided among all pages in the Web graph. PageRank value is computed for each page offline.

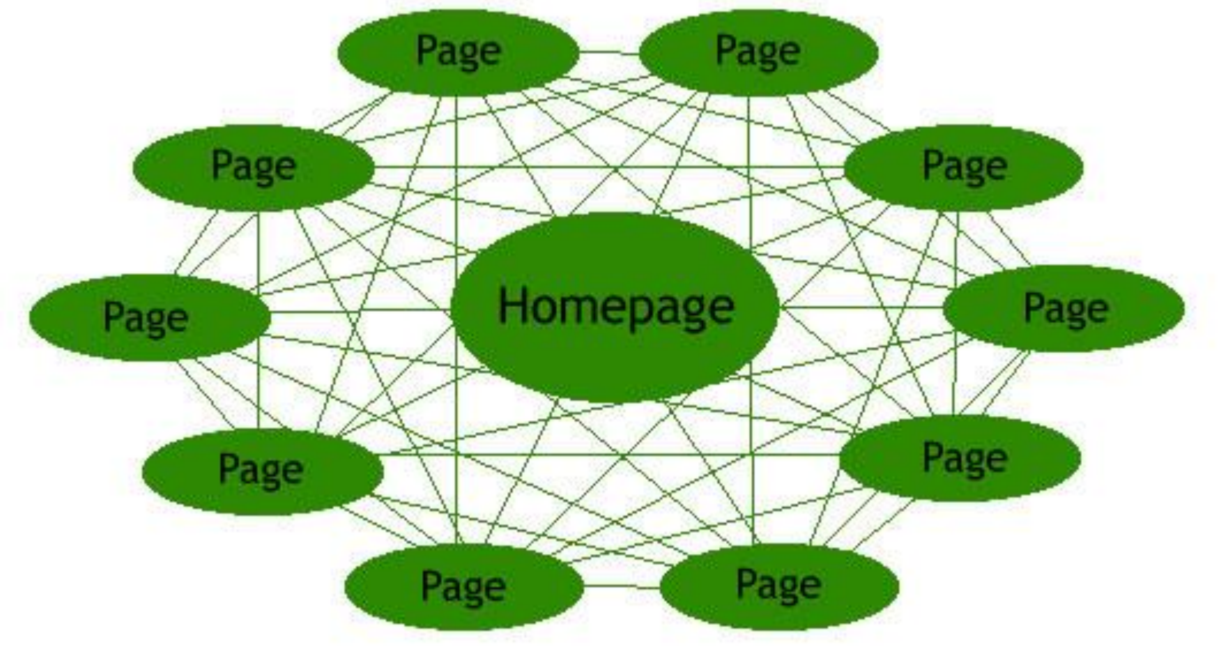


Figure 1.1.1 – Link Structure of Web

How does a PageRank Work?

"Page is important if many important pages point to it"

- Simplified PageRank formula :
  - r = PR(G)
- *Input* : Web graph G = (V,E)
- *Output* : Rank vector r
- Let G have n nodes (pages)
- *In-links* of page i :
  - Hyperlinks that point to page i from other pages
- *Out-links* of page i :
  - Hyperlinks that point out to other pages from page i

- **Original PageRank formula** :

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

- Damping factor d = 0.85
- More general formula : $\mathbf{P} = \mathbf{A}^{\mathrm{T}}\mathbf{P}.$
- Recursive definition :
- Equation of the *eigensystem*, where the solution to P is an *eigenvector* with the corresponding *eigenvalue* of 1
- Computation can be done using power iteration method.

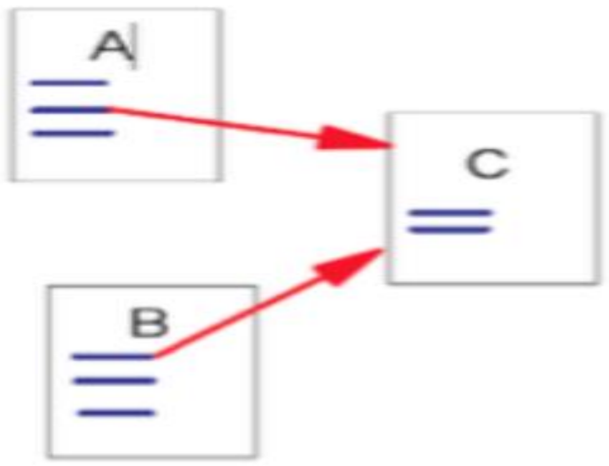


Figure 1.1.2 – In-links and Out-links

Link	In-links	Out-links
A	-	C
B	-	C
C	A, B	-

Table 1.1.1 – In-links and Out-links

In the above figure

- A and B are C's Backlinks or In-links
- C is A and B's Forward link or Out-link

Intuitively, "a page is important if it has a lot of backlinks to it".

Problems with Simplified PageRank

- **Spider Trap**

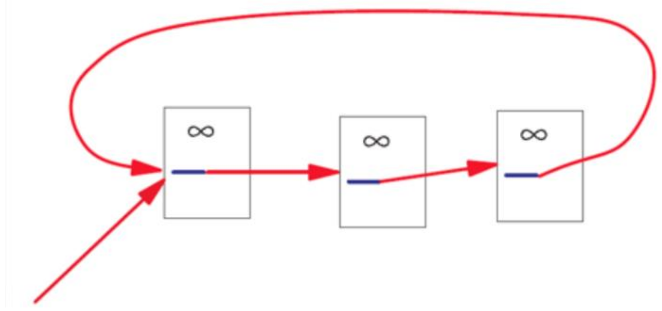


Figure 1.1.3 A loop with spider trap

During each iteration the loop accumulates rank but never distributes rank to other pages. This problem is known as spider traps. These are group of pages that all have out-links, but they never link to any other pages.

- **Dangling Links**

Links that point to any page with no outgoing links. This page with no outgoing link is known as a dead end. The probability of a surfer being anywhere goes to 0, as the number of steps increases.

There are two approaches to deal with dead ends:

1. Remove Dead Ends- Recursively drop dead ends from the graph along with their incoming arcs. Eventually, we will be having a strongly connected component with no dead ends.
2. Taxation- Overcomes the problem of dead end and spider traps, we add another term to the basic formula which consider the probability of moving from one page to another without following any link.

1.2 OBJECTIVE

The primary objective of the project is to implement the Topic Sensitive PageRank algorithm for the better search engine optimization. For providing the efficient search we use 'Topic Sensitive page Rank', which will provide user with the most relevant URLs based on the topic that he is searching for.

The aim of this project is to improve the ranking of search-query results computes a single vector, using the link structure of the Web. TSPR uses a non-uniform personalization vector, not simply a post-processing step of the PageRank computation. Personalization vector introduces bias in all iterations of the iterative computation of the PageRank vector. Hence more accurate search results will be produced with this TSPR algorithm.

1.3 EXISTING SYSTEM

Early Search Engines Without PageRank –

Worked by crawling the web and listing the terms found in each page in an 'Inverted Index'. An Inverted Index is a data structure that makes it easy, given a term to find all the places where that term occurs. Finally, the pages related to search query were extracted from the inverted index and ranked in a way that reflected the use of the terms within the page. There was no concept of PageRank earlier.

HITS Hyperlink-Induced Topic Search –

(also known as **hubs and authorities**) It is a link analysis algorithm that rates Web pages, developed by Jon Kleinberg. The idea behind Hubs and Authorities stemmed from a particular insight into the creation of web pages when the Internet was originally forming; that is, certain web pages, known as hubs, served as large directories that were not actually authoritative in the information that they held, but were used as compilations of a broad catalog of information that led users direct to other authoritative pages. In other words, a good hub represents a page that pointed to many other pages, while a good authority represents a page that is linked by many different hubs.

The scheme therefore assigns two scores for each page: its authority, which estimates the value of the content of the page, and its hub value, which estimates the value of its links to other pages.

The algorithm performs a series of iterations, each consisting of two basic steps:

- **Authority update**: Update each node's *authority score* to be equal to the sum of the *hub scores* of each node that points to it. That is, a node is given a high authority score by being linked from pages that are recognized as Hubs for information.
- **Hub update**: Update each node's *hub score* to be equal to the sum of the *authority scores* of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

Simple generalised PageRank –

As explained in the introduction 1.1

1.4 PROPOSED METHODOLOGY

In this project we implement a topic sensitive pagerank algorithm. There will be the following steps in the implementation.

1. Parsing the files
2. Calculating the topic sensitive pagerank
3. Providing the URLs with highest ranks

1.4.1 Topic Sensitive PageRank

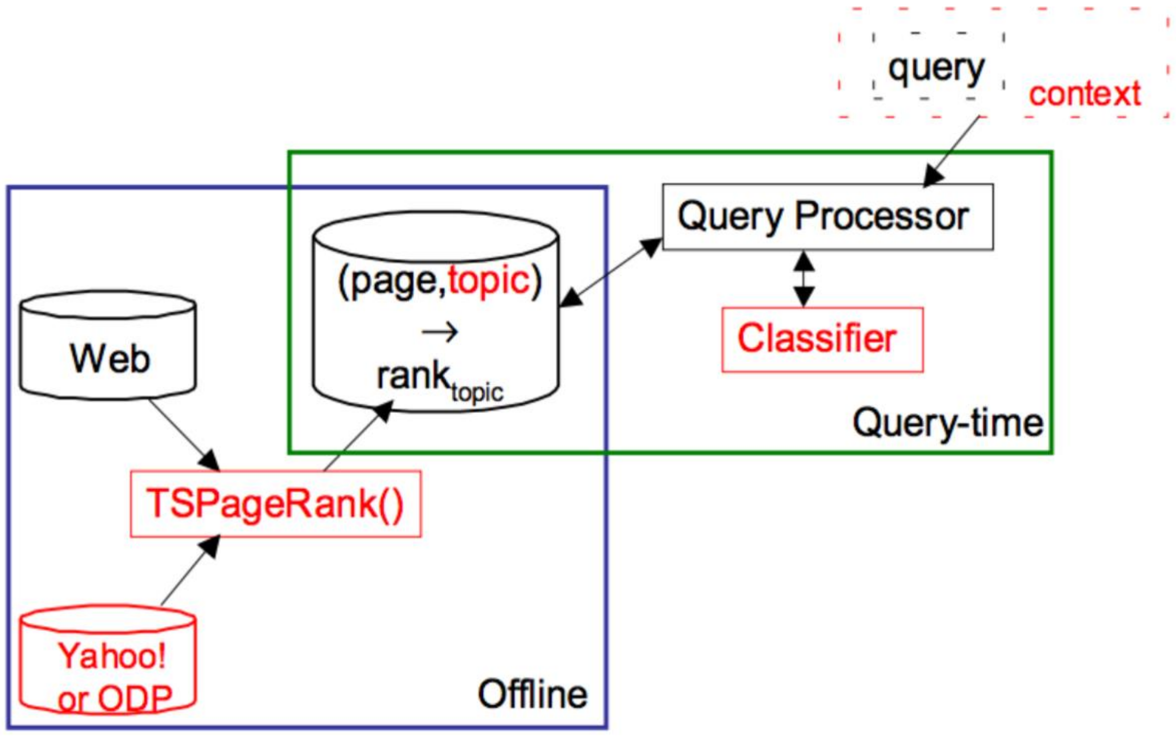


Figure 1.4.1 Representation of TSPR

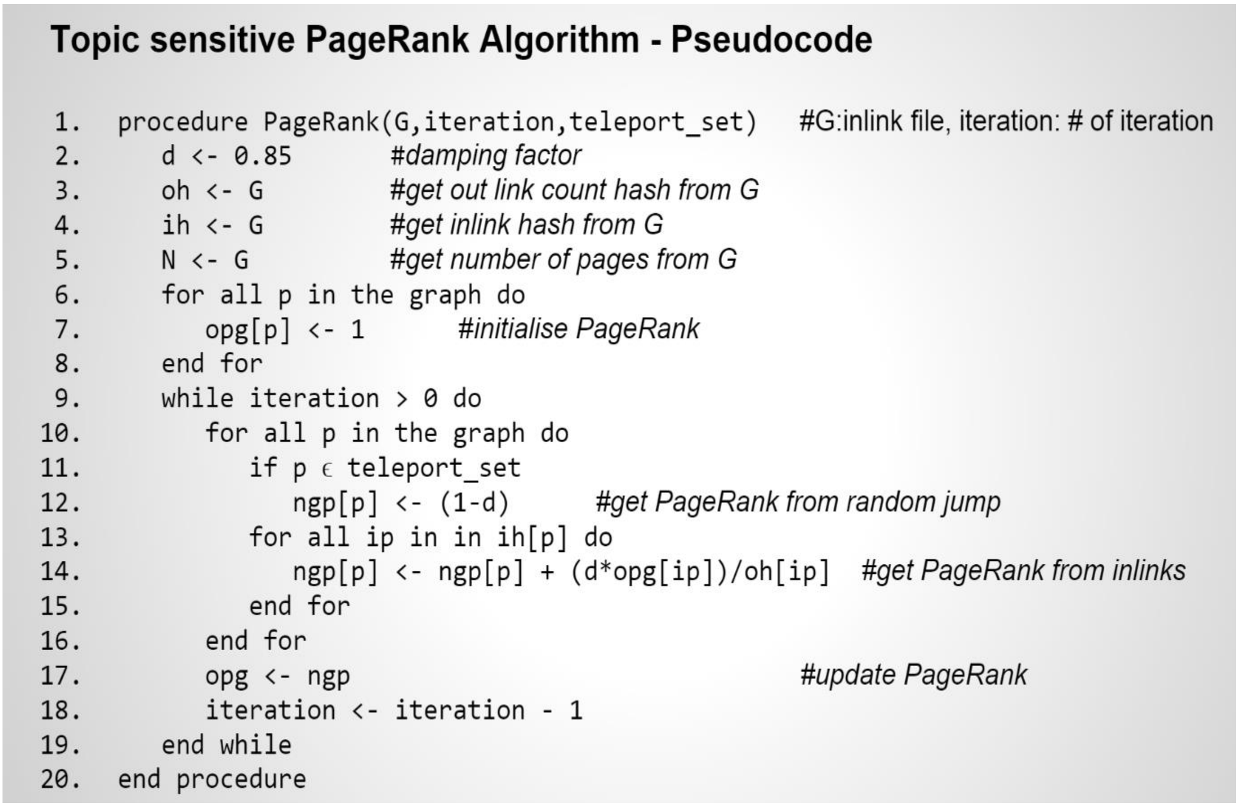
Different people interested in different topics, expressed using the same query. Topic sensitive page rank is Context sensitive ranking algorithm. This approach creates one vector for each of some small number of topics, biasing the PageRank to favor pages of that topic. Here random surfer surfing for let say “sports” will be directed to a random “sports” page only rather than to a page of any kind.

Teleport Set is the set of pages we have identified to belong to a certain topic.
Random walker has a small probability of teleporting at any step
• Teleport can go to:
– Standard PageRank: Any page with equal probability. To avoid dead-ends and spider-traps.
– Topic Specific PageRank: A topic-specific set of “relevant” pages (teleport set)
• Idea: Bias the random walk
– When walker teleports, she picks a page from a set S
– S contains only pages that are relevant to the topic
E.g., Open Directory (DMOZ) pages for a given topic/query
– For each teleport set S, we get a different vector r_S

Matrix formulation
To make this work all we need is to update the teleportation part of the PageRank formulation:

$$A_{ij} = \begin{cases} \beta M_{ij} + (1-\beta)/|S| & \text{if } i \in S \\ \beta M_{ij} + 0 & \text{otherwise} \end{cases}$$

Here, M is the transition matrix of the web, and |S| the size of set S.


Topic sensitive PageRank Algorithm - Pseudocode

```
procedure PageRank(G,iteration,teleport_set)    #G:inlink file, iteration: # of iteration
   d <- 0.85          #damping factor
   oh <- G            #get out link count hash from G
   ih <- G            #get inlink hash from G
   N <- G             #get number of pages from G
   for all p in the graph do
      opg[p] <- 1          #initialise PageRank
   end for
   while iteration > 0 do
      for all p in the graph do
         if p ϵ teleport_set
            ngp[p] <- (1-d)       #get PageRank from random jump
         for all ip in in ih[p] do
            ngp[p] <- ngp[p] + (d*opg[ip])/oh[ip]   #get PageRank from inlinks
         end for
      end for
      opg <- ngp                                    #update PageRank
      iteration <- iteration - 1
   end while
end procedure
```

1.5 MERITS OF THE PROPOSED METHOD

The following are the merits or advantages of the proposed methodology –
Moving from generalised pagerank to Topic sensitive pagerank the following are the merits.

1. Instead of generic popularity, can we measure popularity within a topic.
2. Allows search queries to be answered based on interests of the user
3. The problems like spider traps and the dangling links can be eliminated.
4. TPSR is not a Query independent rank score.
5. Random surfer model which is used in generalized pagerank is not appropriate in some situations.
6. Generalized pagerank is prone to manipulations like (Google bombs, link farms etc) where as topis sensitive page rank is not prone to any kind of manipulations.

1.6 DEMERITS OF THE PROPOSED METHOD

The following are some of the demerits of the proposed methodology –

1. Topis sensitive pagerank is expensive at the runtime, where as generalized pagerank is inexpensive at the runtime.
2. Here scores are not calculated using the entire web graph.
3. Using a small basis set is important for keeping the query-time costs low.

2. SOURCE CODE

2.1 Importing the libraries

```python
import sys, string, math
from math import log
from math import modf
import operator
```

2.2 Data Processing – Parsing the file

```python
outlink_dict = {}
inlink_dict = {}
PRank = {}
new_PRank = {}
Pages = []
SortedPR = []
topic_array = []

d = 0.85

def parse_file(inlink, topic):
    input = open(inlink, 'r')
    for line in input.readlines():
        words = []
        words = str.split(line)
        inlink_dict[words[0]] = words[1:]
        Pages.append(words[0])

    for word in str.split(open(topic, 'r').readline()):
        topic_array.append(word)
    print(topic_array)

    len_dic = float(len(Pages))
    for page in Pages:
        PRank[page] = float(1) / len_dic

    for page in inlink_dict.keys():
        for q in inlink_dict[page]:
            if q in outlink_dict:
                outlink_dict[q] += 1
            else:
                outlink_dict[q] = 1
```

2.3 Calculating the Topic Sensitive PageRank

```
def cal_topic_sensitive_pagerank():
    len_dic = float(len(inlink_dict.keys()))
    ite = 0

    while(ite < 2):
        print(ite)
        for page in PRank.keys():
            topic_related_page = 1 if page in topic_array else 0
            if topic_related_page == 1:
                print(page)
            new_PRank[page] = (float(1 - d)*topic_related_page) / float
(len(topic_array))
            for q in inlink_dict[page]:
                new_PRank[page] = new_PRank[page] + (d * float(PRank.ge
t(q)) / float(outlink_dict.get(q)))
        for page in new_PRank.keys():
            PRank[page] = new_PRank.get(page)
        ite = ite + 1
```

2.4 Getting the top ranked URLs

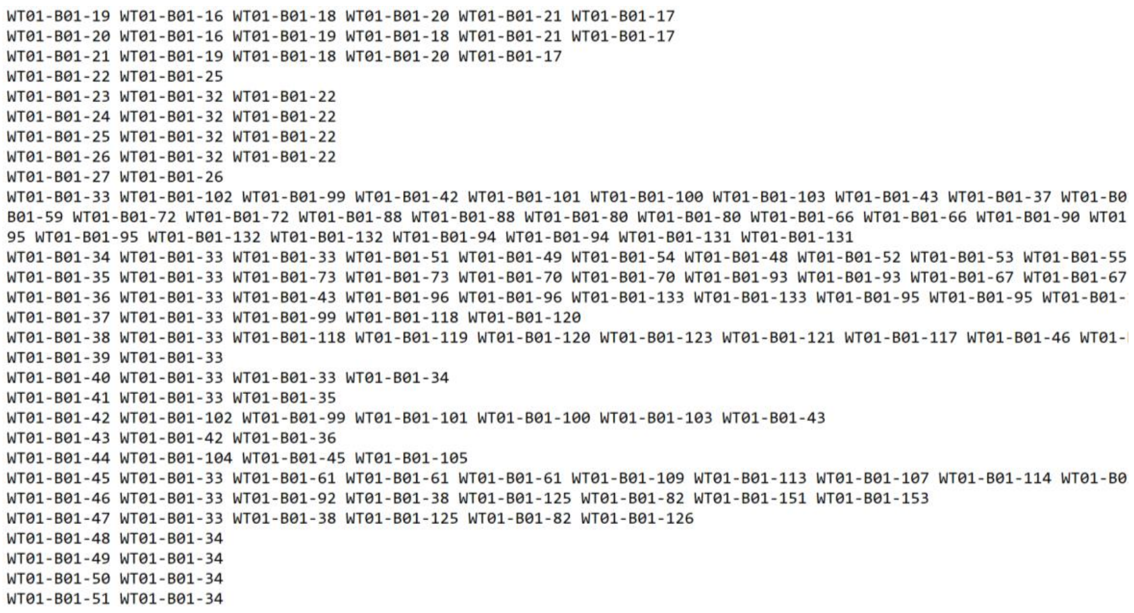
```
def top_rank():
    SortedPR = sorted(PRank.items(), key=operator.itemgetter(1), revers
e=True)
    for i in range(100):
        print(SortedPR[i])
```

2.5 Main function

```
if __name__ == "__main__":
        inlink = 'wt2g_inlinks.txt'
        topic = 'wt2g_example_topic.txt'
        parse_file(inlink, topic)
        cal_topic_sensitive_pagerank()
        top_rank()
```

3. SNAP SHOTS

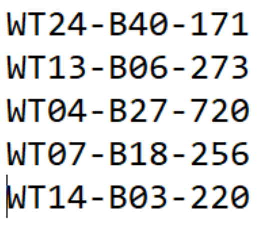
3.1 Data of all Inlinks that used for implementation



```
WT01-B01-19 WT01-B01-16 WT01-B01-18 WT01-B01-20 WT01-B01-21 WT01-B01-17
WT01-B01-20 WT01-B01-16 WT01-B01-19 WT01-B01-18 WT01-B01-21 WT01-B01-17
WT01-B01-21 WT01-B01-19 WT01-B01-18 WT01-B01-20 WT01-B01-17
WT01-B01-22 WT01-B01-25
WT01-B01-23 WT01-B01-32 WT01-B01-22
WT01-B01-24 WT01-B01-32 WT01-B01-22
WT01-B01-25 WT01-B01-32 WT01-B01-22
WT01-B01-26 WT01-B01-32 WT01-B01-22
WT01-B01-27 WT01-B01-26
WT01-B01-33 WT01-B01-102 WT01-B01-99 WT01-B01-42 WT01-B01-101 WT01-B01-100 WT01-B01-103 WT01-B01-43 WT01-B01-37 WT01-B0
B01-59 WT01-B01-72 WT01-B01-72 WT01-B01-88 WT01-B01-88 WT01-B01-80 WT01-B01-80 WT01-B01-66 WT01-B01-66 WT01-B01-90 WT01
95 WT01-B01-95 WT01-B01-132 WT01-B01-132 WT01-B01-94 WT01-B01-94 WT01-B01-131 WT01-B01-131
WT01-B01-34 WT01-B01-33 WT01-B01-33 WT01-B01-51 WT01-B01-49 WT01-B01-54 WT01-B01-48 WT01-B01-52 WT01-B01-53 WT01-B01-55
WT01-B01-35 WT01-B01-33 WT01-B01-73 WT01-B01-73 WT01-B01-70 WT01-B01-70 WT01-B01-93 WT01-B01-93 WT01-B01-67 WT01-B01-67
WT01-B01-36 WT01-B01-33 WT01-B01-43 WT01-B01-96 WT01-B01-96 WT01-B01-133 WT01-B01-133 WT01-B01-95 WT01-B01-95 WT01-B01-
WT01-B01-37 WT01-B01-33 WT01-B01-99 WT01-B01-118 WT01-B01-120
WT01-B01-38 WT01-B01-33 WT01-B01-118 WT01-B01-119 WT01-B01-120 WT01-B01-123 WT01-B01-121 WT01-B01-117 WT01-B01-46 WT01-
WT01-B01-39 WT01-B01-33
WT01-B01-40 WT01-B01-33 WT01-B01-33 WT01-B01-34
WT01-B01-41 WT01-B01-33 WT01-B01-35
WT01-B01-42 WT01-B01-102 WT01-B01-99 WT01-B01-101 WT01-B01-100 WT01-B01-103 WT01-B01-43
WT01-B01-43 WT01-B01-42 WT01-B01-36
WT01-B01-44 WT01-B01-104 WT01-B01-45 WT01-B01-105
WT01-B01-45 WT01-B01-33 WT01-B01-61 WT01-B01-61 WT01-B01-61 WT01-B01-109 WT01-B01-113 WT01-B01-107 WT01-B01-114 WT01-B0
WT01-B01-46 WT01-B01-33 WT01-B01-92 WT01-B01-38 WT01-B01-125 WT01-B01-82 WT01-B01-151 WT01-B01-153
WT01-B01-47 WT01-B01-33 WT01-B01-38 WT01-B01-125 WT01-B01-82 WT01-B01-126
WT01-B01-48 WT01-B01-34
WT01-B01-49 WT01-B01-34
WT01-B01-50 WT01-B01-34
WT01-B01-51 WT01-B01-34
```

Figure 3.1 Data of Inlinks used

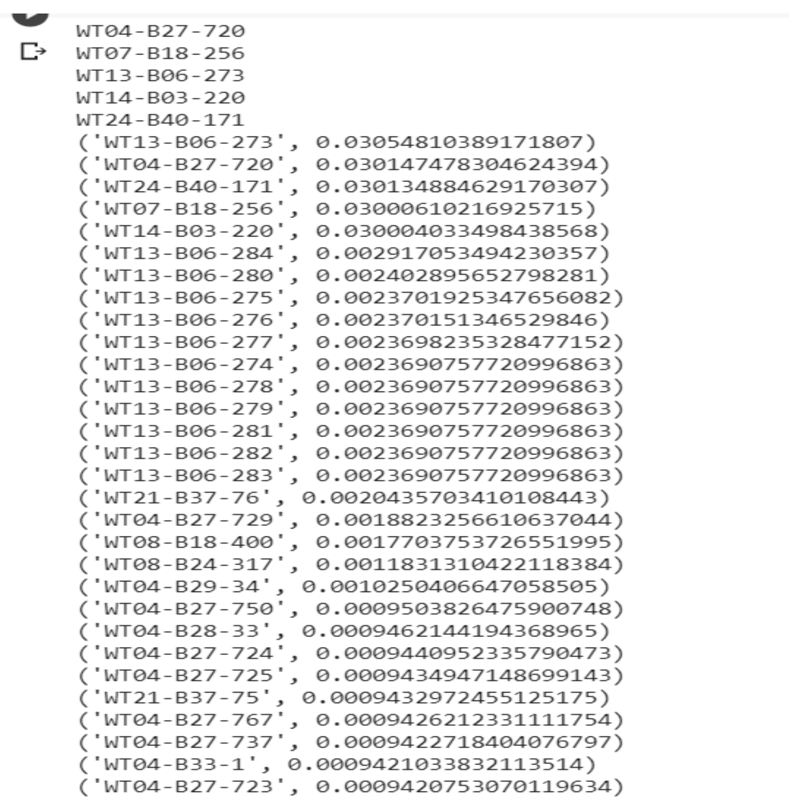
3.2 Data of Example topics that are used for implementation



```
WT24-B40-171
WT13-B06-273
WT04-B27-720
WT07-B18-256
WT14-B03-220
```

Figure 3.2 Example topics used

3.3 Output showing links with top ranks



```
WT04-B27-720
WT07-B18-256
WT13-B06-273
WT14-B03-220
WT24-B40-171
('WT13-B06-273', 0.03054810389171807)
('WT04-B27-720', 0.030147478304624394)
('WT24-B40-171', 0.030134884629170307)
('WT07-B18-256', 0.03000610216925715)
('WT14-B03-220', 0.030004033498438568)
('WT13-B06-284', 0.002917053494230357)
('WT13-B06-280', 0.002402895652798281)
('WT13-B06-275', 0.0023701925347656082)
('WT13-B06-276', 0.00237015134652984б)
('WT13-B06-277', 0.0023698235328477152)
('WT13-B06-274', 0.0023690757720996863)
('WT13-B06-278', 0.0023690757720996863)
('WT13-B06-279', 0.0023690757720996863)
('WT13-B06-281', 0.0023690757720996863)
('WT13-B06-282', 0.0023690757720996863)
('WT13-B06-283', 0.0023690757720996863)
('WT21-B37-76', 0.0020435703410108443)
('WT04-B27-729', 0.0018823256610637044)
('WT08-B18-400', 0.0017703753726551995)
('WT08-B24-317', 0.0011831310422118384)
('WT04-B29-34', 0.0010250406647058505)
('WT04-B27-750', 0.0009503826475900748)
('WT04-B28-33', 0.0009462144194368965)
('WT04-B27-724', 0.0009440952335790473)
('WT04-B27-725', 0.0009434947148699143)
('WT21-B37-75', 0.0009432972455125175)
('WT04-B27-767', 0.0009426212331111754)
('WT04-B27-737', 0.0009422718404076797)
('WT04-B33-1', 0.0009421033832113514)
('WT04-B27-723', 0.0009420753070119634)
```

Figure 3.3 – Output showing links with top ranks

4. CONCLUSION AND FUTURE PLANS

4.1 CONCLUSION

Topic sensitive pagerank algorithm is successfully implemented and top 100 URLs are provided which are having the highest pagerank. Comparing to all other pagerank algorithms like generalized pagerank and trust pagerank etc., More accurate search results will be produced with this Topic Sensitive Page Rank algorithm.

4.2 FUTURE WORK

Integrating TSPR into search engine

1. Decide on the topics for which we shall create specialized PageRank vectors.
2. Pick a teleport set for each of these topics, and use that set to compute the topic - sensitive PageRank vector for that topic.
3. Find a way of determining the topic or set of topics that are most relevant for a particular search query.
4. Use the PageRank vectors for that topic or topics in the ordering of the responses to the search query.

Some other future work may be -

- Use finer grained basis set with in the algorithm.
- Weighting scheme based on page *similarity* to ODP category, rather than page *membership* to ODP category.
- Try a different approach for creating a damping vector ‘p’ used to create the topic sensitive rank vectors. That approach would be more resistant to adversial ODP editors.

5. REFERENCES

1. S. Chakrabarti, M.M. Joshi, K. Punera and D.M. Pennock, "The Structure of Broad Topics on the Web", *Proc. 11th Int'l World Wide Web Conf.*, 2002.
2. M. Diligenti, M. Gori and M. Maggini, "Web Page Scoring Systems for Horizontal and Vertical Search", *Proc. 11th Int'l World Wide Web Conf.*, May 2002.
3. C. Dwork, R. Kumar, M. Naor and D. Sivakumar, "Rank Aggregation Methods for the Web", *Proc. 10th Int'l World Wide Web Conf.*, 2001.
4. T.H. Haveliwala, "Efficient Computation of PageRank", 1999.
5. R. Fagin, R. Kumar and D. Sivakumar, "Comparing Top Lists", *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2003.
6. T.H. Haveliwala, "Efficient Encodings for Document Ranking Vectors", Nov. 2002.
7. T.H. Haveliwala, "Topic-Sensitive PageRank", *Proc. 11th Int'l World Wide Web Conf.*, May 2002.