ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

Priority Based Search Using Fuzzy Logic and Hadoop

Tushar Kapadnis, Harshal Patil, Abhimanyu Malave, Rupali Vairagade

Department of Computer Engineering, Sinhgad Institute of Technology & Science, Pune, India

ABSTRACT: Efficient Search is an emerging information retrieval technology used to retrieve information from the World Wide Web or say Internet. It uses various techniques together to achieve good quality results in minimum time. It uses mainly fuzzy logic, proximity ranking. When users search on line for anything master dataset allocates the dynamic priority for the results. Then the results is saved in file which is uploaded in HDFS. When system is off line, To determine the most searched results on the Internet Proximity ranking is used to rank results so that it can show the most desired results first then less desired one. Algorithms have been used to scan the files created when content is searched on line and generate the relevant results according to the dynamic priority allocation. The main challenge in this work is to minimize the search time so it also implements instant search technique. Algorithm has been implemented for finding the most appropriate results so that most searched results according to the priority will appear first, also proximity ranking algorithm has been used to implement ranking. This paper includes implementation of all above techniques together to create Search optimization technique and to find most searched results according to the dynamic priority allocation. It also shows various experiments carried out on this system. It also presents deep analysis of efficiency & performance of this new system and old one.

KEYWORDS: Search engine Optimization, Ranking, Hadoop, HDFS, fuzzy logic,Dynamic priority.

I. INTRODUCTION

World wide web say Internet is a dynamic concept. Searching for data online can be complicated sometimes due to vastness of Internet. To find the most relevant answers can be challenging sometimes. To remove the ambiguity and complexity from the results and the task of information retrieval itself we need to have a efficient search method to generate the most relevant results. In this paper, we focus on combining the efficient search methods and optimizing large databases to generate the most desired results according to priorities of the user.

Efficient instantsearch : As an emerging information-access paradigm, instant search returns the answers immediately based on a partial query a user has typed in. For example, a music application Gaana, has a search interface that offers instant results to users while they are typing queries 1.When a user types in “Arjit”, the system returns answers such as “Arijit”, “Arijit Singh”, and “Arjit Singh”. Many users prefer the experience of seeing the search results instantly and formulating their queries accordingly instead of being left in the dark until they hit the search button. Recent studies showed that this new information-retrieval method helps users find their answers quickly with fewer efforts.

Fuzzy Search: Users often make typographical mistakes in their search queries. Meanwhile, small keyboards on mobile devices, lack of caution, or limited knowledge about the data can also cause mistakes. In this case we cannot find relevant answers by finding records with keywords matching the query exactly. This problem can be solved by supporting fuzzy search, in which we find answers with keywords similar to the query keywords. Combining fuzzy search with instant search can provide an even better search experiences, especially for mobile-phone users, who often have the “fat fingers” problem, i.e., each keystroke or tap is time consuming and error prone.

HDFS (Hadoop / Bigdata): HDFS is used fordata sets that are so large or complex that traditional data processing application software is inadequate to deal with them. Challenges include capture, storage, analysis, data cu-ration, search, sharing, transfer, visualization, querying and updating. Our approach demands tofind all the answers, compute the score of each answer based on a ranking function, sort them using the score, and return the top results. That is why we have used HDFS at the back-end of storage & analytics.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

II. OUR CONTRIBUTION

We study various solutions to this important aspect. One approach is to first find all the answers, compute the score of each answer based on a ranking function, sort them using the score, and return the top results. However, enumerating all these answers can be computationally expensive when these answers are too many. That's why we have used Hadoop distributed file system to store, analyze, handle and compute the operations involved. When a user searches online for any contents, priority or category is allocated to the result according to the dynamic priority allocation predefined in the master dataset created which contains priorities. So when system is offline, user can select the category/priority and most searched entities of that particular priority will be generated as a result in order of highest rank to lowest one. This approach can be used in aspects like search engine optimization, trending etc. We propose a system which can be used to identify the most searched things on the world wide web say Internet according to the the predefined priorities allocated by master dataset even when system is offline.

III. IMPLEMENTATION

We have elected following three algorithms for implementing the proposed system:
1.Reverse searching :
Input : seed sites and harvested deep websites
Output: relevant sites
1while # of candidate sites less than a threshold do
2 // pick a deep website
3 site = getDeepWebSite(siteDatabase,
seedSites)
4 resultP age = reverseSearch(site)
5 links = extractLinks(resultP age)
6 foreachlink in links do
7 page = downloadPage(link)
8 relevant = classify(page)
9 if relevant then
10 relevantSites = extractUnvisitedSite(page)
11 Output relevantSites
12 end
13 end
14 end

2.Incremental site prioritizing :
Input :siteFrontier
Output: searchable forms and out-of-site links
1 HQueue=SiteFrontier.CreateQueue(HighPriority)
2 LQueue=SiteFrontier.CreateQueue(LowPriority)
3 while siteFrontier is not empty do
4 if HQueue is empty then
5 HQueue.addAll(LQueue)
6 LQueue.clear()
7 end
8 site = HQueue.poll()
9 relevant = classifySite(site)
10 if relevant then
11 performInSiteExploring(site)
12 Output forms and OutOfSiteLinks
13 siteRanker.rank(OutOfSiteLinks)

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

14 if forms is not empty then
15 HQueue.add (OutOfSiteLinks)
16 end
17 else
18 LQueue.add(OutOfSiteLinks)
19 end
20 end
21 end

3.K-nearest algorithms :To demonstrate a k-nearest neighbor analysis, let's consider the task of classifying a new object (query point) among a number of known examples. This is shown in the figure below, which depicts the examples (instances) with the plus and minus signs and the query point with a red circle. Our task is to estimate (classify) the outcome of the query point based on a selected number of its nearest neighbors. In other words, we want to know whether the query point can be classified as a plus or a minus sign.

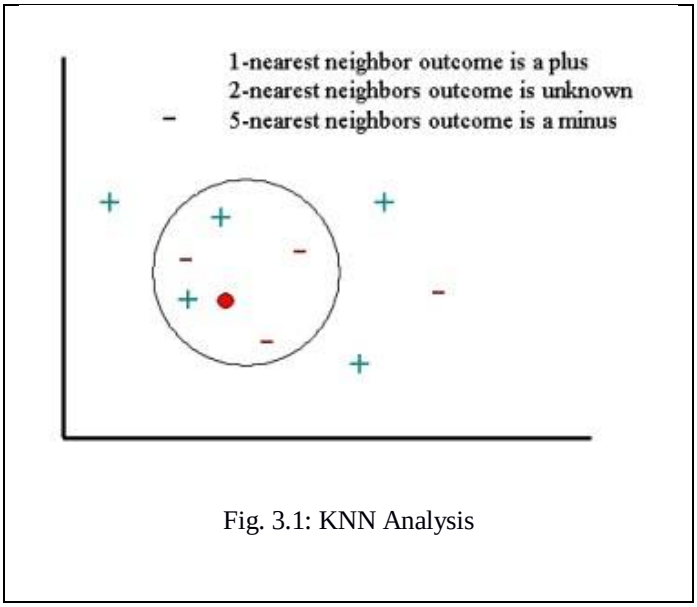

Fig. 3.1: KNN Analysis

To proceed, let's consider the outcome of KNN based on 1-nearest neighbor. It is clear that in this case KNN will predict the outcome of the query point with a plus (since the closest point carries a plus sign). Now let's increase the number of nearest neighbors to 2, i.e., 2-nearest neighbors. This time KNN will not be able to classify the outcome of the query point since the second closest point is a minus, and so both the plus and the minus signs achieve the same score (i.e., win the same number of votes). For the next step, let's increase the number of nearest neighbors to 5 (5-nearest neighbors). This will define a nearest neighbor region, which is indicated by the circle shown in the figure above. Since there are 2 and 3 plus and minus signs, respectively, in this circle KNN will assign a minus sign to the outcome of the query point.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

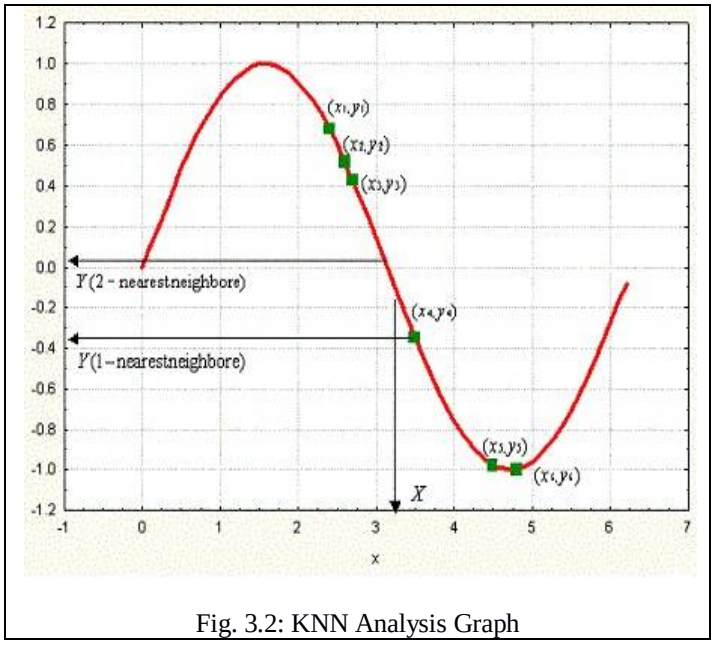

Fig. 3.2: KNN Analysis Graph

IV. SYSTEM ARCHITECTURE

The figure below shows the system architecture of the system. User interacts with the system when online and offline. Online part of the systems allows user to search for content online through Google real time API's. Result content is stored In HDFS and is given a priority by Master Dataset through dynamic priority allocation. Thus when system is offline, user accesses offline system and is given choice of priorities. When selected most searched results are displayed in order of highest rank to the lowest one. Master Dataset is the centre of the whole proposed system architecture.

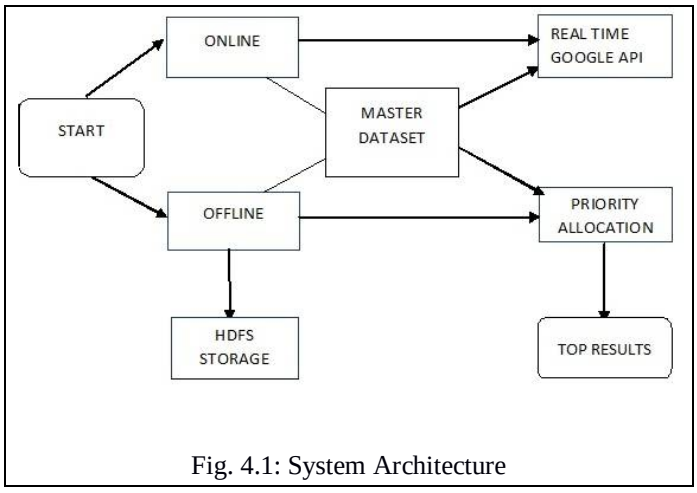

Fig. 4.1: System Architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

V. CONCLUSION

In Proposed system, we can improve ranking of an instant-fuzzy search system by considering proximity information when we need to compute top answers. We presented an incremental-site prioritizing algorithm for finding the more relevant results according to the predefined priorities. We conducted a very thorough analysis by considering space, time, and relevancy tradeoffs of these approaches. We concluded that computing all the answers for the other queries would give the best performance and satisfy the high efficiency requirement of instant search. To conclude, We have proposed a system which can rank the highest searched result content according to the dynamic priority allocation through master dataset even when system is offline.

REFERENCES

[1] Cetindil, J. Esmaelnezhad, C. Li, and D.Newman,"Analysis of instant search query logs," in WebDB, 2012, pp. 7–12.
[2] R. B. Miller, "Response time in man-computer conversational transactions," in Proceedings of the December 9-11, 1968, fall joint computer conference, part I, ser. AFIPS '68 (Fall, part I). New York, NY, USA: ACM, 1968, pp. 267–277. [Online]. Available : http://doi.acm.org/10.1145/1476589.1476628
[3] C. Silverstein, M. R. Henzinger, H. Marais, and M. Moricz, "Analysis of a very large web search engine query log," SIGIR Forum, vol. 33, no. 1, pp. 6–12, 1999.
[4] G. Li, J. Wang, C. Li, and J. Feng, "Supporting efficient top-k queries in type-ahead search," in SIGIR, 2012, pp. 355–364.
[5] R. Schenkel, A. Broschart, S. won Hwang, M. Theobald, and G. Weikum, "Efficient text proximity search," in SPIRE, 2007, pp. 287– 299.
[6] H. Yan, S. Shi, F. Zhang, T. Suel, and J.-R. Wen, "Efficient term proximity search with term-pair indexes," in CIKM, 2010, pp. 1229–1238.
[7] M. Zhu, S. Shi, N. Yu, and J.-R. Wen, "Can phrase indexing help to process non-phrase queries?" in CIKM, 2008, pp. 679–688.
[8] Jain and M. Pennacchiotti, "Open entity extraction from web search query logs," in COLING, 2010, pp. 510–518.
[9] K. Grabski and T. Scheffer, "Sentence completion," in SIGIR, 2004, pp. 433–439.
[10] Nandi and H. V. Jagadish, "Effective phrase prediction," in VLDB, 007, pp. 219–230.
[11] H. Bast and I. Weber, "Type less, find more: fast autocompletion search with a succinct index," in SIGIR, 2006, pp. 364–371.
[12] H. Bast, A. Chitea, F. M. Suchanek, and I. Weber, "Ester: efficient search on text, entities, and relations," in SIGIR, 2007, pp. 671–678.
[13] H. Bast and I. Weber, "The completesearch engine: Interactive, efficient, and towards ir&db integration," in CIDR, 2007, pp. 88–95./
[14] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactivefuzzy keyword search," in WWW, 2009, pp. 371–380.