International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume III, Issue IV, April- 2016

DUPLICATE REMOVER

Vaishakhi V K[1], Neethu.T.Regi[2]
Mtech computer science
Mit
Kannur, india
vaishakhivk@gmail.com[1], neethu.t.regi@gmail.com[2]

Abstract

Most of the URLs collected by web crawlers contains duplicate contents. It becomes a wastage of resources .Thus storing and crawling duplicate data leads to wastage of resources .Hence to solve this several studies are done to detect and remove duplicate contents .To accomplish this ,the methods uses normalization rules to transform all duplicate URLs into the same canonical form. Hence a set of general and precise rules are generated. Here I present the Duplicate Remover which is a new approach of detecting two web pages are copies of each other in terms of their contents. A full multi-sequence alignment is performed on URLs with duplicate content, then the contents of these selected urls are compared using the SizeSpotSigs. Then rules are generated which lead to the deployment of very effective rules.

1. Introduction

Duplicate Remover is a method that takes advantage of multiple sequence alignment. Its importance in biology has motivated many efforts concerning the proposition of optimized algorithms. By applying multiple sequence alignment we can identify similarities and differences among strings. These similarities and differences are used to identify fixed and mutable strings in URLs, resulting in normalization rules. As multiple sequence alignment methods identify all the available strings, this method is able to find more general rules. By using the algorithm called SizeSpotSigs that chooses only stopword features to identify the near duplicate in long Web pages. The similarity between two documents is computed based on the common Jaccard overlap measure between these document set. The duplicate detection technique is to generate a document fingerprint, which is a compact description of the document, and then to compute pair-wise similarity of document fingerprints. The assumption is that fingerprints can be compared much fastly than the

entire document. A common method of generating fingerprints is to select a set of character sequences from a document, and to generate a fingerprint based on the hash values of these sequences. Different algorithms are characterized, and their computational costs are determined, by the hash functions and how character sequences are selected. Similarity between two documents is measured by Jaccard. If the documents have similar content then its urls are selected. We align all the URLs in the dup-clusters generating consensus sequences for each dup-cluster. Rules are then generated from these sequences

2 .Proposed system And Expected Results

To detect if two Web pages are near duplicates in terms of their contents is important in many web applications. Size Spot Sigs is a near duplicate detection algorithm considering the size of the core content in Web page. Here

a set of urls are aligned. Aligned urls are grouped into clusters and rules are generated from it. Learned rules are inserted into the crawler in order to avoid getting more than one URL from the same canonical form. Unlike other approaches, which convert a given URL to another which has the same content here we treat the output of the normalization rule r as a signature to represent all URLs that have very similar page content. Whenever a set of URLs is mapped by the rules to a specific canonical form, they all represented by just one of them and the

others are discarded without checking their content. Here the only situation in where a web crawler loses unique URL is when two or more URLs, that are not duplicate are converted to the same canonical form. Hence to avoid such situation the SizeSpotSigs is used before the urls aligned using multiple url alignment algorithm. After the similar urls are duped into clusters then the similarity in their content is calculated using the SizeSpotSigs algorithm. For calculating the similarity, we need to extract features from Web pages. We define all the features from one page as page-feature set Also we split these features into content-feature set and noise-feature set. A feature comes from the core content of page is defined as content feature and belongs to them content based feature set; otherwise, the feature is called noise feature and belongs to the noise feature set. The noise-content ratio represents the ratio between the size of noise feature set and the size of content feature set.

3.THEORETICAL ANALYSIS

As we know, in fact, near-duplicate detection is to compare the similarity of the core contents of two pages, but Web pages have many noisy content, such as banners and ads.P1, P2 are pages and P1c and P2c are the content feature set P1n and P2n are noise feature set. Most of algorithms is to use sim(P1, P2) to approach sim(P1c, P2c). If sim(P1, P2) is close to sim(P1c,P2c), it shows that the near-duplicate detection algorithm works well, and vice versa. The algorithm Size Spot Sigs is used here to check whether two page contents are same.

3.1 SIZESPOTSIGS ALGORITHM

Input: document vectors di with weighted spot signatures sij .

1. partitions P with boundaries [pk, pk+1] and inverted lists listkj
2. pairs assigned null
3. for all di in random order of |di| using t threads in parallel do
4. partitionk assigned the value P.get(|di|)
5. sort all sij element di by asc. document frequency in partition k
6. D1 assigned 0
7. check di assigned null
8. for all sij element di do
9. listkj assigned partitionk.get(sij)
10. D1 assigned 0
11. for all dj element listkj sorted by descending |dj| do
12. D2 assigned the difference of |di| , |dj|
13. if di = di0 or di0 element of di then
14. continue
15. else if D2< 0 and D1 - D2 > (1 -T) |dj| then
16. continue
17. else if D2< 0 and D1 + D2 > (1 - T) |di| then
18. break
19. else if sim(di, dj) T then
20. pairs assigned pairs ∪ (di, dj)
21. checkedi assigned checkedi ∪ dj
22. end if
23. end for
24. D1 gets D1 + freqdi(sij)
25. if D1(1 - T)max|di0|di0 element of partition check di then
26. break
27. end if
28. end for
29. if pk+1 - |di| (1T)pk + 1 then
30. partition k gets the value P.get(pk+1)
31. goto 6
32. end if
33. end for
34. return pairs

3.2 MULTIPLE URL ALIGMENT ALGORITHM

Input: A dup-cluster C with n duplicate URLs

Output: A tuple p= (consensus; domains; support).

1. LetQbe a priority queue in which tuples s d =(xy; consensusxy ;scoring)

are sorted in descending order according to the alignment scoring.

2. Domains= null; Sequences =null; Support =null; Aligned =null;
3. for all pairs of distinct urls (u1 u2) in C do
4. Support = Support U(u1,u2)
5. Domains = Domains U (domain(u1))U (domain(u2))
6. x = tokenize(u1)
7. y = tokenize(u2)
8. Sequences = Sequences U (x) U (y)
9. s =PairURLAlignment(x,y)
10. add s to Q
11. end for
12. while Q is not empty do
13. Pop the first tuple s from Q
14. if s.x not Aligned and s.y not Aligned then
15. Aligned = Aligned U (s.x) U (s.y)
16. Sequences = Sequences - Aligned
17. for all sequences s in Sequences do
18. e= PairURLAlignment(s.consensus,s)
19. add e to Q
20. end for
21. Sequences = Sequences U (s.consensus)
22. end if
23. end while
24. Let s be the unique consensus sequence in Sequences
25. return p = (s,Domains, Support)

3.3GENERATE CANDIDATE RULES ALGORITHM

Input: Training Set TS = (c1,..., cn) with n duplicate clusters Output: Set of
m candidate rules CR = (r1,....,rm)
1. Create table RT (context, transformation, domains, support)
2. Create table CRT (context, transformation, domains, support)
3. for all cluters ci element of TS do
4. T = selectKRandomlyURLsFrom(ci)
5. p = MultipleURLAligment(T)
6. r = generateRule (p.consensus)
7. add (r.context, r.transformation, p.domains, p.support) to RT
8. end for
9. group tuples in RT into buckets by (context, transformation)
10. for all buckets B do
11. if(|B| >= min freq) then
12. Domains = null ; Support =null ;
13. for all tuples t element of B do
14. Domains = Domains U t.domains
15. Support = Support U t.support
16. end for
17. a = the first tuple in B
18. add (a.context, a.transformation, Domains, Support) to CRT
19. end if
20. end for
21. return a set CR of rules created from CRT

3.4 VALIDATE RULES ALGORITHM

Input: VS: validation set, CR: Set of n candidate rules, fprmax: maximum false-positive rate that can be tolerated, minsupp: minimum number of instances required.
Output: Set of n valid rules VR fr1; . . . ; rng
1. Create table CT (canonical, url)
2. Create table RT (context, transformation, domains, support)
3. for all candidate rules r in CR do
4. Nsupp = 0; Nfpp = 0; Support = null;
5. U = all d element of r.Domains URLs from domain d in VS.
6. for all urls u in U do
7. if r.context matches with u then
8. canonical = NormalizeURL (u, r)
9. add (canonical, u) to CT
10. end if
11. end for
12. group tuples in CT into buckets by (canonical)
13. for all buckets B do
14. if (|B| > 1) then
15. for all pairs of distinct tuple t1,t2 element of B do
16. Nsupp = Nsupp + 1

17. Support = Support U (t1.url, t2,url)
18. if (t1.url and t2.url are not DUST) then
19. Nfpp = Nfpp + 1
20. end if
21. end for
22. end if
23. end for
24. if (Nsupp >= minsupp) then
25. fpr = Nfpp/Nsupp
26. if (f pr <= f prmax) then
27. add (r.context, r.transformation, r.domains, Support)to RT
28. end if
29. end if
30. Clear table CT
31. end for
32. return
28. end if
29. end if
30. Clear table CT
31. end for
32. return a set of all rules in RT

4.CONCLUSION

In this work, I present Dust Remover, a new method to address the DUST that is, the detection of distinct URLs that correspond to pages with duplicate or near duplicate content. SizeSpotSigs checks by the contents whether the urls have the same content. It learns normalization rules that are very precise in converting distinct URLs which refer the same content to a common canonical form, making it easy to detect them. To achieve this, it applies a novel strategy based on a full multi-sequence alignment of training URLs with duplicate content. By analyzing the alignments obtained, accurate and general normalization rules can be generated.

5. FUTURE WORK

As future work, we intend to improve the scalability and precision of our method, as well as to evaluate it using other datasets. By the use of more efficient multiple sequence alignment algorithms which are able to align n sequences in time proportional to O(n log n).

ACKNOWLEDGMENT

Any mission never concludes without cordial co-operation from surroundings. I take this opportunity to acknowledge all the people who have helped me kind heartedly in every stages of this work. It is a matter of great pleasure for me to express my sincere gratitude and appreciation to Prof. Dr. Syed Ummer M Thangal, Principal, Mr.Rijin I.K, Asst. Professor, Head of Department of Computer Science and Engineering. I express my sincere thanks to PG co-ordinator Ms.Lijina S.S, Asst. Professor , Department of Computer Science and Engineering , for the valuable guid- ance and continuous support.I am really thankful to my Project guide Mrs.Neethu.T.Regi Asst.Professor, Department of Computer Science and Engineering, co-ordinator Ms.Navya E.K, Asst. Professor, Department of Computer Science and Engineering for their sincere co-operation for success of this work. I am also thankful to my classmates and well-wishers for sharing their knowledge and suggestions.

References

[1] Kaio Rodrigues, Marco Cristo, Edleno Moura, and Altigran da Silva, Removing dust using multiple alignment of sequences.

[2] Martin Theobald, Jonathan Siddharth, and Andreas Paepcke, Spotsigs: robust and e_cient near duplicate detection in large web collections, Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2008, pp. 563–570.

[3] William Pugh and Monika H Henzinger, Detecting duplicate and nearduplicate files, December 2 2003, US Patent 6,658,423.