Web-Site Boundary Detection Using Incremental Random Walk Clustering

Ayesh Alshukri, Frans Coenen and Michele Zito

Abstract In this paper we describe a random walk clustering technique to address the Website Boundary Detection (WBD) problem. The technique is fully described and compared with alternative (breadth and depth first) approaches. The reported evaluation demonstrates that the random walk technique produces comparable or better results than those produced by these alternative techniques, while at the same time visiting fewer 'noise' pages. To demonstrate that the good results are not simply a consequence of a randomisation of the input data we also compare with a random ordering technique.

1 Introduction

There have been numerous studies that perform analysis of the web at the web-page level [8, 2, 6]. A web page can be considered as a low level element on the web. There is potential benefit to studying the web on a higher level, that level being the level of a website. A website can be considered as a single high level entity that is made up of a collection of lower level resources. These resources can include text/html web pages, images, video etc, which together collectively represent a website. Studying the web at this abstract level can have many possible applications, including; web archiving [21, 1], web search and information retrieval [20] and web analysis [7, 10, 17, 8]. The Web-site Boundary Detection (WBD) problem is concerned with discovering all web resources within the boundary of a particular website given some "seed(s)" resources from that website. WBD is an open problem [7] and can be difficult to solve [14].

In this work we follow [5] in postulating that a set of resources contained in a website is, by definition, 'intended' by the author/publisher of the content to be

Ayesh Alshukri · Frans Coenen · Michele Zito
The University of Liverpool, Department of Computer Science, Ashton Building, Ashton Street, Liverpool, L69 3BX, UK. e-mail: {a.alshukri, coenen, michele}@liverpool.ac.uk

deemed related using the high level notion of a website. It is this concept of intention that is used in this paper to define a website, in that the resources featured in the website share similar attributes, and also they are more highly connected.

The WBD problem can be addressed in a static manner by first collecting all pages connected to a given seed page up to a certain distance (depth) from the seed page. If the depth is sufficiently large the resulting collection can be expected to contain both pages belonging to the target web site and noise pages. Typically, so as to ensure that the entire target web site is included in the collection the distance moved from the seed page has to be substantial, hence a significant number of noise pages will be included in the collection. The task is then to differentiate the target pages from the noise pages, i.e. identify the web site boundary. The static WBD problem can thus be argued to be a binary clustering problem where we wish to divide the collection of web pages into a web site cluster and a noise cluster.

An alternative approach, which aims to limit the number of noise pages visited, is to proceed in a dynamic manner and attempt to identify the web site boundary using some form of incremental clustering. There are a number of ways whereby this may be achieved. For example we can proceed in a Breadth First (BF) or Depth First (DF) manner, stopping whenever it is determined that the web site boundary has been identified. In this paper we advocate a random walk based approach. As will be demonstrated the Random Walk (RW) technique produces comparable or better performance than the BF or DF approaches, while at the same time visiting far fewer nodes.

The evaluation is conducted using synthetic data sets generated using the web modelling technique proposed in [15] and [16]. The reason that we elected to use artificial data sets is that, for experimental purposes, we have much greater control over the ratio of web pages against noise pages. We compared the operation of the RW approach with the BF and DF approaches. We also compare the operation of RW with an alternative random walk approach, Random Ordering (RO), to demonstrate that the effectiveness of RW is not simply a result of the randomisation of the input data (which has been shown to have a positive effect with respect to K-means [23, 19]).

2 Preliminaries

The RW approach advocated in this paper is directed at a portion of the world-wide web. We may think of this as a graph $G = (V,E)$, where V is a collection of web-pages, and the set E keeps track of all directed links between pairs of elements of V. Without loss of generality assume that G is connected. Each page has a numerical feature vector associated with it. Technically this can be determined once the page has been downloaded from the internet. Thus, there exists a function $f : V \to \mathbb{R}^k$, for some fixed integer $k \geq 1$, such that for any $P \in V$, $f(P)$ is an ordered sequence of k real numbers characterising the page P. In what follows we will denote the elements of V using italicised capital letters. We will use the terms page, web-page,

node, and vertex interchangeably, but in all cases we will actually be referring to the pair formed by an element of V and its corresponding feature vector. Given a specified web-page $P \in V(G)$, our purpose is to define a set $\mathscr{W} = \mathscr{W}(P) \subseteq V(G)$ called the *web-site* of P, containing all pages of G that are similar to P.

In the work described in this paper we focus on iterative clustering processes founded on the template presented in Table 1. The graph G is assumed to model a portion of the web. For the purpose of our experiments (see Section 3.1) we may assume that it resides on some secondary memory storage device and bits of it are retrieved as needed. If the algorithms were to be used on the real web, G would be distributed at different sites across the internet and its content would have to be downloaded through http requests. Note that the well known k-means [11, 13, 22, 24] clustering method fits this template, however the template can equally be used to represent a host of different algorithms. In the forthcoming sections we provide details concerning the proposed iterative clustering processes proposed in this paper.

```
Algorithm clustering_template (P)
  W = {P}; N = {};
  set up the process internal state;
  repeat
    select a page Q from G;
    add Q to W or N;
    update the process state;
  until convergence;
  return W;
```

Table 1 Clustering algorithm template.

2.1 *k*-means clustering

The classical k-means clustering algorithm is obtained from the template given in Table 1 by assuming that the state of the process contains information about the (two) clusters centroids, that the pages in G are inspected one at a time in some given order (which may vary over subsequent iterations) and that the state update is only performed after a complete sweep of the data has been performed. Also, it is assumed that the graph $G = (V,E)$ is not initially available. It is retrieved from the web incrementally as different pages get requested inside the process main loop. The loop is then further repeated until the system state does not change from one sweep of the graph to the next one.

This paper considers two variants of this process, named Breadth-First (BF) and Depth-First (DF) clustering. They are obtained by assuming that the order in which the pages of G are examined is determined by the way in which they are retrieved from the web, and is given by a fixed and bound breadth-first (resp. depth-first)

traversal of the web starting at P, including all pages up to a certain bounded distance from P.

2.2 Clustering based on random crawling

Many different processes fit the template described above. The order in which the pages are considered needn't be fixed or deterministic. The way in which the web-graph is explored can be quite arbitrary. Furthermore the process state may be updated every time a new page is considered, rather than at the end of a whole sweep of the given dataset. Finally the periodical state update might include major modification of the retrieved web-graph content. In the remainder of this section we describe two possible processes of this type: Random Walk (RW) clustering and Random Ordering (RO) clustering. The first is appealing because of its simplicity, the second will be used to critically analyse the other processes. Other variants, not reported in this paper, were tested but the results were found not to be significantly different from those of the two methods considered.

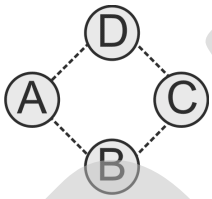

Fig. 1 Example web dataset displayed as a graph; vertices's indicate web pages, dashed edges indicate hyperlinks (directions omitted so as to maintain clarity)

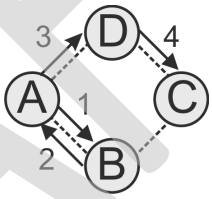

Fig. 2 Sample run of RW (numbered directed edges indicate progress of the random walk)

2.3 Random Walk (RW) Clustering.

Given an arbitrary graph G, the sequence of vertices visited by starting at a vertex and then repeatedly moving from one vertex to the next by selecting a neighbour of the current vertex at random is termed a *random walk* on the graph G (see for instance [18]). Random walks arise in many settings (e.g. the shuffling of a deck of cards, or the motion of a dust particle in the air) and there's a vast literature related to them (the interested reader is referred to the classical [12], or the very recent [3] and references therein). In particular they can be used [4] as a means for exploring a graph.

The process RW is the variant of the clustering_process in Table 1 based on the idea of performing a random walk on G. A pure random walk is not easily simulated

(at least initially) if we want to keep the constraint that the process does not require access to the full dataset to start off with. The sequence of pages to be (re-)clustered is thus given by the order in which they are visited by performing a random walk on the *known portion* of G. Initially the walk has to choose a neighbour of P. In general, given the current page Q, the page to be visited next is selected at random among those pointed at by a link of Q and the set of pages seen so far that point to Q. For example, with reference to the example graph given in Figure 1, the first four steps of the process might be as shown in Figure 2, with clustering performed at every step. Note that the walk can revisit nodes multiple times, even before having completed a full sweep of G. It is therefore convenient to re-compute the centroids after each step of the walk, rather than at the end of a sweep. It is well-known [4] that any random walk on a finite connected graph eventually visits all vertices in it. Thus, in principle, the process could run until convergence as in the standard k-means algorithm. It will turn out, however, that stopping the process after a given maximum number of "steps" (MAXITERATIONS) is more effective and still results in good quality clusters. Some pseudo code describing the RW technique is presented in Table 2. The decision on whether to add Q to W or the *noise cluster* N is based on the computation of the Euclidean distance between Q's feature vector and the centroids of the two clusters $W \setminus \{Q\}$ and $N \setminus \{Q\}$.

```
Algorithm RW (P)
  W = {P}; N = {};
  set Q to P; set a counter to one;
  set up the process internal state;
  repeat
     redefine Q to be a random neighbour of Q in G;
     add Q to W or N;
     increase the counter;
     update the process state;
  until counter goes past MAXITERATIONS;
  return W;
```

Table 2 Pseudo code for RW

2.4 Random Ordering (RO) clustering.

Random walks, as defined above, are examples of so called Markov stochastic processes [12]. The evolution of a process of this type is fully determined by its current state: in the example in Figure 1, and assuming perfect information, every time we visit vertex A we have a 50-50 chance of moving to B or D and no chance at all to visit vertex C next.

The process RW, strictly speaking, already breaks this framework. In this section we describe a simpler process (RO) that moves even further from a pure random walk process. The process traverses the graph vertices in a random order in a similar manner as in the case of RW. However in the case of RO, all vertices are picked with

equal chance, irrespective of edges. In Figure 1, again assuming perfect knowledge (In reality a deterministic crawl can identify all nodes in the structure to a certain depth). Every time we visit vertex A, there is a 1 in 4 chance of visiting either vertex A,B,C or D. In this process edges are disregarded, and thus we produce a complete random ordering of the graph.

```
Algorithm RO (P)
  W = {P}; N = {};
  set Q to P; set a counter to one;
  set up the process internal state;
  repeat
    redefine Q to be a random vertex in G;
    add Q to W or N;
    increase the counter;
    update the process state;
  until counter goes past MAXITERATIONS;
  return W;
```

Table 3 Pseudo code for RO

It is this selection of nodes, chosen in a random order that is the key difference between RO and RW. The traversal of RW is influenced by the graph based on its structural properties. For instance if a graph contains a highly connected sub set of nodes, then RW is more likely to re-visit this locality, in contrast to less connected nodes. The RO method would visit vertices of the graph at random, and would not be effected by any structural properties of the graph.

3 Experimentation and Evaluation

To assess the quality of RW we tested it on a number of artificial data sets. Graphs $G = (V,E)$ containing a particular set of pages similar to some chosen element of V were put together by first creating artificial host graphs using the well established web-graph model proposed by Kumar *et al.* [15]. One of the generated host graphs was then selected to be the target graph (web site). The graphs were then represented using a standard feature vector representation that included noise words randomly selected from a "bag" of noise words. The generation of the experimental data is described in further detail in Section 3.1. To evaluate the operation of RW we used a number of measures, these are given in Section 3.2. The results are presented in Section 3.3.

3.1 Experimental Data

As noted above, for experimental purposes, we modelled a collection of web pages using the process described in [15], but with case 3 (see below) derived from [16]. Given a positive integer m, the process generates a synthetic model of the world

wide web starting from a graph $G_0^{K,m}$ having a single vertex with m links pointing to itself. Then, for $t \geq 1$, $G_t^{K,m}$ is derived from $G_{t-1}^{K,m}$ according to the following procedure (here α, β, and ν are real numbers between zero and one):

1. With probability $\alpha \times \beta$ add a new vertex to $G_{t-1}^{K,m}$ with m links pointing to itself.
2. With probability $\alpha \times (1-\beta)$ choose a random edge in $G_{t-1}^{K,m}$ and make its source point to P.
3. With probability $(1-\alpha) \times \beta$, pick a random copying vertex Q_c; a new vertex P will point to m vertices chosen as follows:

   Uniformly At Random - with probability ν, choose a random vertex Q and add (P,Q) to the graph.
   Preferential Attachment - with probability $1-\nu$, add (P,R) to the graph, where R is a random neighbour of Q_c.

4. With the remaining probability $(1-\alpha) \times (1-\beta)$ no new vertex is generated and a random edge is added to $G_{t-1}^{K,m}$.

In our experiments we used values of m mirroring the fact that the average page on the world wide web contains some 40-50 links [9]. Also $\alpha = 0.01$, $\beta = 0.9$ and $\nu = 0.2$. So, most of the time, new vertices will be generated to which 50 neighbours will be linked according to the procedure described in 3 above. The resulting graph shared many features with the real web, in particular: (i) in and out degree distribution, (ii) the diameter, and (iii) the presence of small bipartite "cliques". For the purpose of our experiments, we generate $G_T^{K,m}$ and then remove $G_0^{K,m}$ from it. Because of the copy mechanism by which we add edges to $G_{t-1}^{K,m}$, the single page in $G_0^{K,m}$ contains a large number of links and is linked by a large number of "younger" pages. Removing it from the graph used for our experiments makes the resulting graph, which we denote by K_T more realistic.

The use of a synthetic version of the web has many advantages. However Kumar's graphs have one disadvantage: the lack of the "clustery" nature of the real web. To complete the definition of our artificial data we need to identify a web-site $\mathscr{W}$ within K_T (and then complete the definition of G by adding some noise cluster, $\mathscr{N}$). To this end we performed the following steps:

1. Given K_T, we picked a random node X from this graph. Such node will represent the home-page in $\mathscr{W}$.
2. To create the set of pages in $\mathscr{W}$ we then performed a breadth-first crawl of K_T starting from X, up to a certain maximum depth. The nodes visited by such process are then added to $\mathscr{W}$ with some fixed probability p.
3. The noise cluster $\mathscr{N}$ is created by continuing the breadth-first crawl until a further maximum depth, it contains all nodes reachable from X that have not been added to $\mathscr{W}$ in the previous step.
4. The graph G is then defined as the subgraph of K_T induced by the set $\mathscr{W} \cup \mathscr{N}$.

For the experiments reported in this paper we generated datasets of four types, which we identify as sets A, B, C and D. In each case the data set included 10 distinct

graphs G. All sets were generated from copies of K_T using $m = 40$ and $p = 0.8$. Sets of type A were derived from graphs that included in $\mathscr{W}$ all nodes (pages) at a distance of at most three links (with $p = 0.8$) from the seed page X, while continuing to a distance of 5 links from the initial page so as to define the noise cluster $\mathscr{N}$. Sets B, C, and D used the same distance of three links for set $\mathscr{W}$, and distance six, eight and nine links respectively to define $\mathscr{N}$. In Table 4 some graph-theoretic statistics are presented the values reported in the table are averages over the 10 graphs included in each set. From the table it can be observed that the size of noise clusters steadily increases from set A to D, while the size of the target cluster $\mathscr{W}$ remains fairly constant. Also $\mathscr{W}$ has many more internal links than links to $\mathscr{N}$ (modelling the fact that the elements of $\mathscr{W}$ represent homogeneous web-pages). Furthermore $\mathscr{W}$ is *popular* in the sense that many links from the noise cluster $\mathscr{N}$ point back to $\mathscr{W}$.

Given a particular graph structure G, we may generate several instances of this structure by changing the feature vectors that we associate with the various pages (vertices). In general, for any page $P \in \mathscr{W}$ (resp. in $\mathscr{N}$, the feature vector $f(P)$ can be chosen from the superposition of two k-dimensional normal multivariate distributions, having different means, $\mu_{\mathscr{W}}$ and $\mu_{\mathscr{N}}$, and equal deviations σ. The vectors associated with the elements of $\mathscr{W}$ may be chosen from the $N_k(\mu_{\mathscr{W}}, \sigma)$ distribution, those for $\mathscr{N}$ from $N_k(\mu_{\mathscr{N}}, \sigma)$. Varying the relative position of the means results in datasets of differing complexity.

For our experiments we simulated such process in a very simple way. Given P in $V(K_T)$, $f(P)$ contains $k = 20$ integer numbers, corresponding to the number of occurrences of the elements of a pool of words S in a bag associated with P. The pool S is split in two disjoint groups of equal size, $S_{\mathscr{W}}$ and $S_{\mathscr{N}}$. The bag of P is defined by sampling, independently, k elements of S. If P is in $\mathscr{W}$, each chosen word has a chance $\pi = 0.7$ (resp. $1 - \pi$) of belonging to $S_{\mathscr{W}}$ (resp. $S_{\mathscr{N}}$), and is selected uniformly at random with replacement, from the chosen group. If $P \in \mathscr{N}$, the selection is symmetrically biased towards $S_{\mathscr{N}}$.

Set	Cluster $\mathscr{W}$			Cluster $\mathscr{N}$		
	Average # nodes	Intra edges	Inter edges	Average # nodes	Intra edges	Inter edges
A	100	389.5	128.2	198.8	224.2	260.7
B	100.8	409.3	159.6	405.4	729.6	692
C	102.1	422.6	145.8	405.4	2015.3	1702.1
D	103.1	392.6	188.4	1440.4	4329.1	2734.4

Table 4 Data set graph-theoretic statistics. *Intra edges* are links connecting two pages in the same cluster. *Inter edges* are links connecting pages in different clusters.

For experimental purposes, using the above described mechanism, 40 graphs were generated and divided into four sets as shown in Table 4. The distinction between the four sets is that they feature an increasing amount of noise. The scenario created by a set reflects a portion of the web containing a particular website, which also contains an amount of noise. In a perfect scenario, this portion of the web would contain only pages from a single website. Then the task of identifying this website would trivially be the whole set. In reality the amount of noise that can be collected

is substantial, as a website that resides on the www is surrounded by a large amount of pages that are not part of the website itself. Therefore each of the four data sets must contain a restricted amount of noise. Set A simulates a scenario where the portion of the web graph is restricted to a noise to class ratio of 1:2, for each set there after the amount of noise is double that of the previous set.

An ever increasing amount of noise in the data to be clustered, has the effect of increasing the complexity of the clustering process in that large numbers of noise items increases the likelihood of outliers thus blurring the distinction between $\mathscr{W}$ and $\mathscr{N}$. Also, in the deterministic BF and DF processes, each of the sets simulates an increasing uncertainty in the amount of the web to collect to ensure all class items are contained.

3.2 Evaluation Criteria

We used a number of measures to compare the performance of the various clustering methods considered in this paper. Given an instance (G, f) generated as described in Section 3.1, the output (W,N) of a particular clustering algorithm $\mathscr{A}$ on (G, f) can be expressed as:

$$W = W_t \cup W_f \qquad N = N_t \cup N_f$$

where W_t (resp. W_f) is the collection of class items (noise items) that are correctly (resp. incorrectly) identified as being within the target web site, and, similarly, N_t (resp. N_f) is the collection of noise items (resp. items in W) that are correctly (resp. incorrectly) identified as noise. The *accuracy* $\gamma = \gamma(\mathscr{A}, (G, f))$ of algorithm $\mathscr{A}$ on input (G, f) is given by the expression:

$$\frac{|W_t| + |N_t|}{|\mathscr{W}| + |\mathscr{N}|}$$

which measure the proportion of correctly classified items.

For completeness we will also track two obvious *coverage* parameters:

$$\chi_{\mathscr{W}} = \frac{|W_t| + |N_f|}{|\mathscr{W}|} \qquad \chi_{\mathscr{N}} = \frac{|W_f| + |N_t|}{|\mathscr{N}|},$$

the number of steps $\varsigma = \varsigma(\mathscr{A}, (G, f))$ (i.e. iterations of the main algorithm loop) performed and the average CPU time to complete a single step, denoted by $\theta = \theta(\mathscr{A}, (G, f))$ calculated as the total execution time to terminate a particular run divided by how many steps the algorithm took to terminate.

3.3 Results

Table 5 shows the performance of RW, in comparison with DF, BF and RO, using data generated as described in Section 3.1. The results show the performance of each method with respect to the test data SetA to SetD. Each of the methods were run 50 times on each group of graphs in each test set.

Table 5 Table shows the average performance of each method (RW, RO, BF and DF) run 50 times on each graph in sets *A* to *D*. Accuracies and coverages are reported as percentages

	Algorithm	Accuracy	Class Coverage	Noise Coverage	Execution	End Steps
SetA	BF	77.21	100	100	0.89	1001
	DF	93.77	100	100	0.85	1001
	RO 10k	94.02	100	100	0.16	10k
	RW 10k	94.66	89.85	66.84	0.12	10k
SetB	BF	81.24	100	100	1.41	1600
	DF	92.12	100	100	1.48	1600
	RO 10k	84.53	99.92	99.93	0.28	10k
	RW 10k	94.04	83.42	48.12	0.15	10k
SetC	BF	74.97	100	100	3.66	5290
	DF	77.08	100	100	3.46	5290
	RO 10k	69.59	97.05	95.96	1.92	10k
	RW 10k	92.04	79.38	19.68	0.21	10k
SetD	BF	51.17	100	100	6.24	11266
	DF	51.55	100	100	6.21	11266
	RO 10k	66.32	79.36	77.76	7.03	10k
	RW 10k	91.75	84.16	08.64	0.24	10k

The effect of an incremental increase in noise on the performance of each of the techniques can be seen in the plots given in Figures 3, 5 and 4. In the accuracy plots (Figure 3) we can observe a sharp decrease in the accuracy of the BF and DF techniques as the noise increases. This is due to the fact that both the BF and DF methods consider all nodes in the graph (Figures 5 and 4) and thus as the amount of noise is increased the performance decreases. RO also displays a decreasing accuracy trend, but is much more resilient to the increasing presence of noise than BF and DF. This is due to the fact that RO effectively randomises the ordering of nodes. This produces a clustering that is much less susceptible to a "bad" initial starting conditions than in the case of the BF and DF methods. As shown in the accuracy plot given in Figures 3, RO has a much more subtle decrease in accuracy. RO also covers slightly less noise nodes than BF and DF (Figure 5), but displays a decreasing trend with respect to the amount of class items that are covered (Figure 5).

The accuracy of RW is only slightly effected by the increasing amount of noise. RW achieves a consistently better accuracy than the RO, BF and DF methods. The reasons for this improved accuracy are: (i) RW has the advantages of covering a smaller percentage of noise nodes, which can make it easier for the clustering algorithm to determine clusters; and (ii) RW randomises the ordering of nodes, which

serves to improve the clustering accuracy by over-coming a possible bad initial conditions.

RW tends not to cover the entire graph (in reasonable time). It is also slower than the DF and BF which both cover all nodes. The performance of RW is relative to its run time. However we argue that, in the context of web-site boundary detection (where the aim is to identify all pages belonging to a website), it is satisfactory to correctly classify 95% of the website page if 40% of noise is not accessed at all. It should also be noted that the overall run time is not a very meaningful measure; the average running time per step is a better comparative measure of the effectiveness of the methods considered.

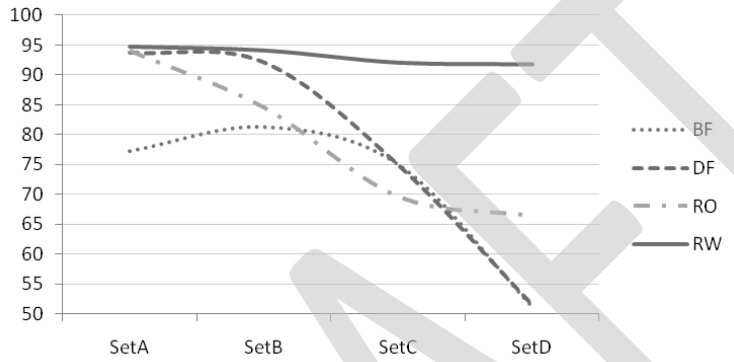

Fig. 3 Accuracy

In summary RW offers the following advantages: (i) it does not need to be re-run several times to overcome a bad initial clustering condition, because it randomises the nodes while traversing the hyperlink structure of the graph; (ii) it visits fewer noise pages thus reducing the resources used for downloading, parsing and pre-processing of web pages; and (iii) produces comparable or better accuracy in relation to the website boundary detection problem.

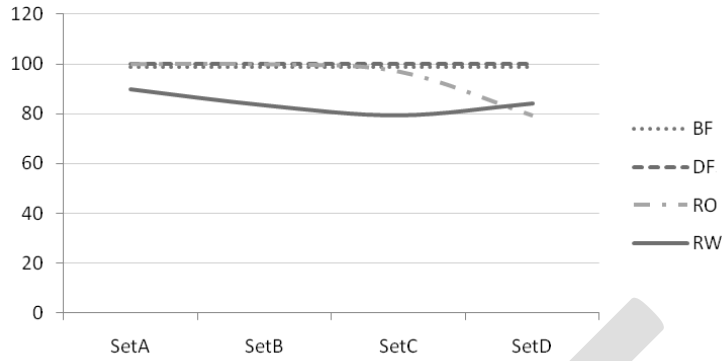

Fig. 4 Class coverage

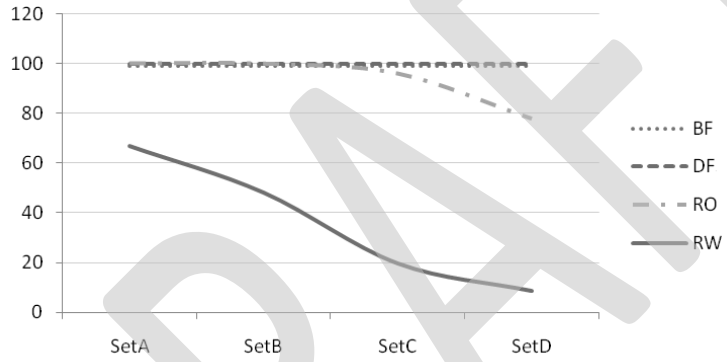

Fig. 5 Noise coverage

4 Conclusion

In this paper we have described a random walk approach (RW) as a solution for the Website Boundary Detection (WBD) problem. The operation of RW was evaluated with respect to three comparator approaches: BF, DF and RO. The main findings are as follows:

The BF and DF methods give accuracies that decrease when the amount of noise increases. This is due to the fact that the inclusion of an increasing amount of noise items make clustering more difficult as noise items blur the distinction between the target web site cluster and the noise cluster.

The RO method covers the elements of a graph in a random ordering, irrespective of previously selected nodes or the link structure. This method does have the

advantage of randomising the node ordering so as to enhance the clustering process. As demonstrated by the results obtained, randomising the node ordering can improve the clustering, thus RO has consistently improved accuracy over BF and DF. However the RO method is also susceptible to noise, as the accuracy decreases when more noise is included in the graph. The RO method could, of course, not realistically be used for the dynamic resolution of the WBD problem.

The RW method offers the advantage of improved clustering accuracy (obtained by randomising the ordering of nodes) over the other methods included in the evaluation. However, because RW utilises the link structure of the graph, it also reduce the number of noise nodes visited. Thus requiring less resources in the context of the WBD problem, while giving the best performance over RO, BF and DF.

References

1. S. Abiteboul, G. Cobena, J. Masanès, and G. Sedrati. A First Experience in Archiving the French Web. In *ECDL '02 Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, volume 2458 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002.
2. R. Albert, H. Jeong, and A-l. Barabási. Diameter of the world wide web. *Computing Research Repository*, 1999.
3. D. Aldous and J. Fill. Reversible Markov chains and random walks on graphs. *Monograph in preparation*, 2002.
4. R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. *20th Annual Symp. on Foundations of Computer Science*, pages 218–223, 1979.
5. A Alshukri, F. Coenen, and M. Zito. Web-Site Boundary Detection. In *Proceedings of the 10th Industrial Conference on Data Mining*, pages 529–543, Berlin, Germany, 2010. Springer.
6. Albert-Laszloand Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 1999.
7. K. Bharat, B-W. Chang, M. Henzinger, and M. Ruhl. Who links to whom: mining linkage between Web sites. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 51–58, Washington, DC, USA, 2001. IEEE Computer Society.
8. A. Z Broder. Graph structure in the Web. *Computer Networks*, 33(1-6):309–320, June 2000.
9. A. Z Broder, M Najork, and J. L Wiener. Efficient URL caching for world wide web crawling. In *WWW'03 Proceedings of the 12th international conference on World Wide Web*, pages 679–689, Budapest, Hungary., 2003. ACM.
10. P. Dmitriev. As we may perceive: finding the boundaries of compound documents on the web. In *WWW'08 Proceeding of the 17th international conference on World Wide Web*, pages 1029–1030, Beijing, China, 2008. ACM.
11. M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2002.
12. W. Feller. Introduction to probability theory and its applications. *WSS*, vol. 1, 1968.
13. J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.
14. M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291. ACM, 2006.
15. R. Kumar. Trawling the Web for emerging cyber-communities. *Computer Networks*, 31(11-16):1481–1493, May 1999.

16. R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the Web graph. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 57–65, Washington, DC, USA, 2000. IEEE Computer Society.
17. B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer, Springer-Verlag New York, Inc., 2007.
18. L. Lovász. Random walks on graphs: A survey. *YaleU/DCS/TR-1029*, 2:1–46, 1994.
19. J.M Peña, J.A Lozano, and P Larrañaga. An empirical comparison of four initialization methods for the K-Means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, October 1999.
20. J. Pokorn. Web Searching and Information Retrieval. *Computing in Science and Engineering*, 6(4):43–48, 2004.
21. P. Senellart. Identifying Websites with Flow Simulation. In David Lowe and Martin Gaedke, editors, *ICWE*, volume 3579 of *Lecture Notes in Computer Science*, Orsay, France., 2005. Gemo, INRIA Futurs., Springer.
22. P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson International Edition, 2006.
23. B. Meck Thiesson, C. Chickering, and D. Heckerman. Learning mixtures of Bayesian networks. Technical report, Microsoft Research Technical Report TR-97-30, Redmond, WA, 1997.
24. I. H. Witten and E. Frank. *Data Mining: practical machine learning tools and techniques*. Morgan Kaufman, 2005.