Rank-Aware Crawling of Hidden Web sites

George Valkanas
Dept. of Informatics &
Telecommunications
University of Athens, Greece
gvalk@di.uoa.gr

Alexandros Ntoulas[*]
Zynga
San Francisco, California
antoulas@zynga.com

Dimitrios Gunopulos
Dept. of Informatics &
Telecommunications
University of Athens, Greece
dg@di.uoa.gr

ABSTRACT

An ever-increasing amount of valuable information on the Web today is stored inside online databases and is accessible *only* after the users issue a query through a search interface. Such information is collectively called the "Hidden Web" and is mostly inaccessible by traditional search engine crawlers that scout the Web following links. Since the only way to access the Hidden Web pages is through the submission of queries to the Hidden Web sites, previous work [14, 18] has focused on how to automatically generate queries in order to incrementally retrieve and cover a Hidden Web site *in depth*, as much as possible.

For certain applications however it is not necessary to have crawled a Hidden-Web site in-depth. For example, a meta-searcher or a content aggregator will utilize only the top portion of the ranked result lists coming from the querying of a Hidden Web site instead of its full content. Hence, if we can crawl a Hidden Web site *in breadth*, i.e. download just the top results for all potential queries, we can enable such applications without the need for allocating resources for fully crawling a potentially huge Hidden Web site.

In this paper we present algorithms for crawling a Hidden Web site by taking the ranking of the results into account. Since we do not know all potential queries that may be directed to the Web site in advance, we study how to approximate the site's ranking function so that we can compute the top results based on the data collected so far. We provide a framework for performing ranking-aware Hidden Web crawling and we show experimental results on a real Web site demonstrating the performance of our methods.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms, Experimentation

Keywords

Web Mining, Hidden Web, Crawling, Ranking

1. INTRODUCTION

The content available through Web-accessible databases today is ever-increasing in quantity and quality. Such content is typically dynamically generated without any pre-existing direct links to it, and is available only after the users provide a query through a Web search interface. As a result, content from Web-accessible databases is essentially "hidden" from Web search engines who scout the Web following links in order to discover and download content. Such content is collectively called the "Hidden Web".

As the Hidden Web contains information of high quality and high value to the users [2] there have been previous efforts in surfacing it, so that it can be part of a search engine's central index. Since the only way for surfacing the Hidden-Web content is by filling the Web forms, previous work [14, 18, 10] has focused on how to automatically generate queries. Such queries are either selected from a pre-compiled list or selected on-the-fly and are iteratively issued to a given Hidden-Web site in order to retrieve its content. The goal of these approaches is to cover a Hidden-Web site as much as possible *in depth*, i.e. download as much of its content as possible with a small number of queries.

However, although downloading all of (or as much as possible) a Hidden-Web site is obviously desirable, there are cases where some Hidden-Web sites offer their services at a cost (e.g.*www.westlaw.com*). If our goal is to download all of the Hidden-Web site at a minimum cost, the techniques presented in [14, 18, 10] are directly applicable. There are however applications on the Web that do not necessarily require a Hidden-Web site to be completely crawled. For example, a meta-search engine needs to operate mostly on the top few results from a Hidden Web site, for a given query. Similarly, a content aggregator service operating over Hidden-Web sites needs to also know the top few items in order to provide a good summary to its readers. Hence, if we can crawl and cache a local copy of a Hidden Web site *in breadth*, (i.e. download only the top results for the potential queries that we care about), we can provide such applications the data they need without paying the cost of downloading fully a potentially very large Hidden-Web site.

One big challenge with this approach is that we do not know the queries that will be issued to, e.g., a meta-searcher in advance. One solution to this problem is to query the Hidden-Web site on-the-fly and cache the top results for every query. This approach however, may lead to unnessecary queries (and thus cost) to the Hidden-Web site. For example, we may identify the top-k results of the query "digital camera" by examining the top results of the query "camera".

[*] Work done while the author was at Microsoft Research.

Fourteenth International Workshop on the Web and Databases (WebDB 2011) June 12, 2011 - Athens, Greece

In this paper, we study the problem of creating a small, broad local copy of a Hidden-Web site in an rank-aware manner. Our goal is to use this copy in order to get the top-k results for a given query instead of querying the Hidden-Web site directly. Our main idea is to approximate the ranking function employed by the Hidden-Web site and leverage it in order to determine how to crawl it. We extend previous techniques for automatic query generation to take into account the ranking function of each Web site and we showcase the performance of our techniques in a real Web site.

In short, our contributions here are the following:

- We extend previous techniques by using an active learning [17] variation to crawl Hidden-Web sites, hence deriving a family of algorithms that cover the site while taking ranking into account. Our technique is generic and uses domain-independent features, so that it may be applied to numerous Hidden-Web sites without major modifications.
- Since our technique determines which queries to use based on the retrieved ranked results, we show that it manages to achieve better coverage performance than previously existing techniques.
- We provide an experimental evaluation of our proposed technique, by crawling YouTube. An interesting outcome of the evaluation is that our techniques can be very easily parallelized in order to crawl Hidden-Web sites.

The rest of the paper is organized as follows: Section 2 formally defines the problem, followed by Section 3 which introduces our proposed approach. Section 5 demonstrates our experimental findings. Section 6 presents related work and finally Section 7 concludes the paper, with ideas for future work.

2. BACKGROUND

We start by formally defining the problem of ranking-aware Hidden-Web crawling and our goals. To make our discussion more concrete, we assume that a Hidden-Web site is associated with a database D, containing documents $D = \{d_1, d_2, ..., d_{|D|}\}$. We represent each document d_i within D as a bag of words, i.e. $d_i = \{w_i^1, w_i^2, ..., w_i^{|w_i|}\}$. Similarly, we represent the set of unique terms contained by all documents in D by $T = \{t_1, t_2, ..., t_{|T|}\}$.

We define $D_q \subseteq D$ to be the set of documents retrieved by query q. These documents are the *relevant* ones w.r.t. q as they were returned by the Hidden-Web site. Let R be a *ranking* (permutation) of D_q, i.e. $R = R(q, D_q) \rightarrow \{1, 2, .., |D_q|\}$. Note that we are only interested in the *rank* (position) of each document. Also each document appears only once in a ranking. Based on this notation, we can now formally define our goal in rank-aware crawling of a Hidden-Web site:

Definition 1. Given a Hidden-Web site, identify the minimum set of queries that covers the content from the site that allows us to approximate the ranking of all potential queries.

Intuitively, we want to retrieve as much content from the Hidden Web site as possible, while maintaining the queries probed to the site at a minimum. At the same time, we are interested in the ranking that each query is associated with. Since the set of unique terms in D is T, there are $|T|$ rankings for single-keyword queries, and $2^{|T|}$ combinations of queries. However, since keyword search commonly employs AND semantics which restrict the results, we can achieve better coverage by issuing only single keyword queries. Therefore, since coverage is a basic goal, we want to issue fewer queries than $|T|$ while being able to approximate the ranking of any $t_i \in T$ that has been retrieved so far.

3. RANK-AWARE CRAWLING

In this section we develop and describe our proposed metric, to show how we may include the ranking performed by a Hidden-Web site as a paremeter while crawling it.

3.1 Efficiency

As we already discussed in the previous section, a fundamental goal of our techniques is to minimize the download cost when crawling a Hidden-Web site while achieving high coverage. Similar to [14], in order to compare among crawling techniques, we use the notion of *efficiency* for a query term t, defined as:

$$Efficiency(t) = \frac{P_{new}(t)}{Cost(t)}$$

where P_{new} is the fraction of new items (over all current items) that term t is expected to retrieve and $Cost$ is the overall cost associated with issuing term t, measured for example in money, bandwidth, or communications between the crawler and the Hidden-Web site. In [14] the authors use the $Efficiency$ metric described above to determine which queries the crawler should issue to the Hidden-Web site. Their goal is to maximize the coverage of the site using the minimum number of queries. Since we are also interested in coverage we will be using the $Efficiency$ metric as well. However, as we also aim at approximating the ranked results of the queries coming from a Hidden Web site, we will extend this metric by considering the "ranking gain" of the query term as discussed next.

3.2 Ranking Gain

Apart from the coverage aspect, we are also interested in the ranking associated with each term. First, we describe the intuition behind the ranking gain of a term: it is a measure of the degree to which we can approximate the term's ranking, i.e. the ordering of results we would obtain by querying this term. In other words, we evaluate how similar a *derived* ranking would be to the actual one. If we are able to accurately reconstruct the ranking of a term, then that term's gain ($w.r.t.$ ranking) would be close to 0. On the contrary, if the derived ranking introduces a big error in our attempt to approximate the actual one, then, we would need to query this term. The need to correctly derive the ranking of a term is important for several reasons, both for single and multi-keyword queries. Finally, we note that already queried terms have a ranking gain of 0, as their actual ranking is known.

Therefore, to evaluate the ranking gain of a term we would need to know its actual ranking, so that we can measure the distance between the two. However, in that case, we would not need to derive it in the first place, hence, we take a different approach on computing the ranking gain. The idea is that documents in which the term already exists in may provide clues about the overall ranking for that term. Even if they do not provide evidence for the actual ranking itself, they can still be useful.

Once the ranked result of querying the site with term t has been retrieved, we parse the documents and extract the terms they contain. For each term, we maintain a set of inverted indexes of the documents it is contained in with respect to each probed query, keeping the ordering in which the result was returned. This results in a set of orderings for each term, which we can aggregate to obtain a single one and compute the ranking gain as their level of disagreement. The more these rankings disagree among them, the more likely we consider it to be that the inferred ranking will be misleading. That is because their aggregate list, which seems as a natural choice for the derived list, would try to average all distances, which would result in a lot of information being lost. We can then define the measure used to compute the ranking gain of a term t as:

$$RankingGain(t) = \sum_{i=1}^{N} d(r_i, agg(t))$$

where $agg(t)$ is the aggregate list of the N rankings r_i, $\forall i = 1, .., N$, of term t and d is a distance function between two ranked inputs. An aggregate list is one that minimizes the distance between itself and all other input lists.

What we state in the above equation is that by querying t, what we gain for its ranking aspect is equal to the overall disagreement of the rankings that t belongs to already. Terms with identical rankings among queries have a ranking gain of 0, as we do not benefit with respect to this parameter. Terms with rankings that exhibit a strong correlation will have a lower benefit for that factor compared to ones with higher discordance.

A major drawback of this approach is that it needs to compute the aggregate list of each term t. However, ranking aggregation is known to be NP-Hard [5].Moreover, it can not be efficiently maintained incrementally. This entails recomputation of the aggregate lists from scratch. Hence, it is in our best interest to avoid consuming the crawler's resources on computing aggregate lists. Instead, we use the following observation:

Observation 1. The bigger the distance is between two ranked inputs of a term t, the bigger the ranking gain of this term will be.

We can then reformulate the ranking gain as

$$RankingGain(t) = \frac{2}{N*(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} d(r_i, r_j)$$

hence computing the pair-wise distances between all ranked inputs of a term. To avoid boosting inputs with bigger lengths, we take the average of pair-wise distances, by dividing with the number of all possible pairs.

3.3 Putting it all together

Since we are interested in both maximizing coverage and achieving good results in approximating the ranking of the Hidden-Web site, we combine the two cost models presented above into a single one. More specifically, we use their weighted average as:

$$Gain(t) = \frac{(1-w) * P_{new}(t) + w * RankingGain(t)}{Cost(t)}$$

The cost is paid only once, as it is related to the query term and to the number of pages that it will retrieve, and is the same for both coverage and ranking gain. Retrieved documents are parsed and terms are extracted, and we select as the subsequent query the one with the highest (overall) *Gain*.

4. RANKING DISTANCES

There are several measures that are particularly suited to compute the distance between two or more ordered lists, most of which aim at finding the degree of correlation. Such rankings include Kendall τ, Spearman footrule and Spearman ρ, top-K variants, nDCG etc. We have used the ones that are most commonly used in the bibliography, due to several properties that they exhibit (i.e. extended Condorcet criterion). In our work we experimented with different metrics for computing ranking distances as they directly affect our crawling strategies. More specifically, we experimented with Kendall *tau*, *Spearman Footrule*, *Top-K* and variations of these, to account for potential ties or different ranking lengths.

4.1 Kendall τ

Kendall τ measures the correlation of two given lists, as the number of pairwise swappings, given by the equation

$$\tau = \frac{2*(n_c - n_d)}{n*(n-1)}$$

where n_c is the number of concordant pairs, n_d the number of discordant pairs and n the number of distinct elements in both lists. We normalize the result in the $[0, 1]$ range.

We have also used Kendall $\tau - b$ to account for ties, following the methodology in [6]

4.2 Spearman Footrule

Another commonly used ranking distance is the *Spearman footrule*, where the distance of two rankings $r1$ and $r2$ is given by:

$$D(r_1, r_2) = \sum_{i=1}^{n} |r_{1_i} - r_{2_i}|$$

and r_{i_j} is the index of $j-th$ element in ranking r_i. To address the problem of ranking length variation, we have also used *scaled Spearman footrule (SSF)*, given by

$$D(r_1, r_2) = \sum_{i=1}^{n} |\frac{r_{1_i}}{|r_1|} - \frac{r_{2_i}}{|r_2|}|$$

4.3 Top-K

In essence, users are not interested in the entire ranking that a web site performs but rather in the top ranked ones, known as the top-k documents. Hence, the top-k distance captures the difference between the first k ranked documents, disregarding all others. Given a ranking r, we denote by $r(k)$ its first k entries. Using the formula from [7], the distance between r_1 and r_2 up to position i is given by

$$\delta_i(r_1, r_2) = \frac{|(r_1(i) \cup r_2(i)) - (r_1(i) \cap r_2(i))|}{2 * i}$$

This distance captures the fraction of non commonly shared items in the first i positions. Then, to get the overall distance, we sum δ's, for $i = 1, ..., k$. More formally

$$D(r_1, r_2) = \frac{1}{k} \sum_{i=1}^{k} \delta_i(r_1, r_2).$$

5. EXPERIMENTAL EVALUATION

5.1 Experimental setup

We have conducted a set of experiments to measure our approach in terms of effectiveness, on YouTube [19]. YouTube is a social sharing video service web site, where users are able to upload videos and search for them through a simple keyword interface.

The YouTube service limits its results to 1000 items, however it contains duplicate entries, i.e. the same video (identified by url) appears in different ranking positions. In such cases, we maintain only the first occurrence of a video, as this is when a user will see it for the first time and possibly select it. After removing duplicates, the maximum number of returned videos per query is about 800 on average. This fact poses an upper bound on the number of videos we expect to see in our experimental results. Our results are from crawling the site between March 1st and March 28th, 2011. Instead of letting the crawlers run indefinitely, we limit the number of queries each one may probe to 300. Each configuration runs independently of the others and computes its statistics based on the set of documents that it has retrieved by itself.

We consider as documents the text-based information of the videos, i.e. title, description and tags. We did not include the user comments as they are not always directly related to the video at hand.

5.2 Harvest rate

We start our presentation of the experimental results by first evaluating the number of documents that each approach manages to retrieve from the Hidden Web site. We compare our methods with the ones presented in [14].

Figures 1 and 2 show the number of documents that each of the configurations retrieved from the Hidden Web site. Each configuration is selecting keywords based on our discussion in Section 3 but using a difference distance metric as shown in Table 1. Knowing the upper limit of YouTube query results (i.e. 800), we display our findings as a percentage of the *optimal* retrieval case, where each new query retrieves the maximum number of new distinct documents. It is interesting to note that the approach that does not take ranking into account at all, performs the lowest among all of the techniques.

The graphs in Fig. 1 are a direct application of the discussion in Section 3, using different distances. We have also experimented with a variation of these techniques, the results of which are shown in Fig. 2. In this case, we have weighted the outcome of the distance function by the *Jaccard coefficient* of the two rankings, i.e. the fraction of their common documents.

Table 1: Crawling configurations

ID	Distance
CVR	*None*
KTA	*Kendall τ-a*
KTB	*Kendall τ-b*
SF	*Spearman Footrule*
SSF	*Scaled Spearman Footrule*
TOPK	*Top-X*
cKTB	*KTB with Jaccard weighting*
cSF	*SF with Jaccard weighting*
cSSF	*SSF with Jaccard weighting*
cTOPK	*TopX with Jaccard weighting*

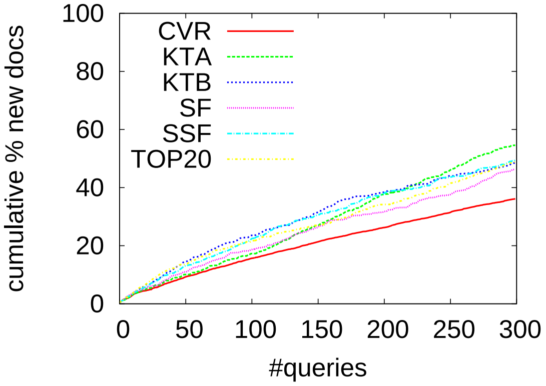

Figure 1: Harvest rate (%) of approaches

As we observe from the graphs, the different configurations of our crawling policy perform differently. Overall, all policies achieve better coverage than the one in [14] which does not optimize for the rankings. The highest harvest rate is given by Kendall τ-a, followed by SSF. The reason that the coverage-only approach performs the lowest, is that it relies on static features alone, whereas YouTube ranks results in a query dependent manner. Though we do not know the exact ranking function of the Hidden-Web site, loosing the (explicit) coverage aspect and increasing other, more dynamic, features seems to be beneficial. Interestingly, we also observe that the policies start performing better after issuing about 100 queries. This is due to the fact that the policies need to acquire some knowledge about the document collection before they start selecting good queries to probe.

5.3 Empirical comparison of the approaches

Table 2 shows the 15 first terms for each of the approaches. Apart from the 1st query, which was the seed term and the 2nd one, where the ranking has not yet taken effect, it is clear that all configurations differentiate from each other as soon as the 3rd query. An interesting observation is that, apart from the coverage approach, the first 10 query terms in all configurations appear to be semantically related to the first one, i.e. *food*.

Figure 3 provides a more general view of query correlation among configurations. This grid map shows the number of commonly queried terms between any two of our about 30 configurations, measured as the percentage of their intersection. The closer the value is to 1.0, the higher the corre-

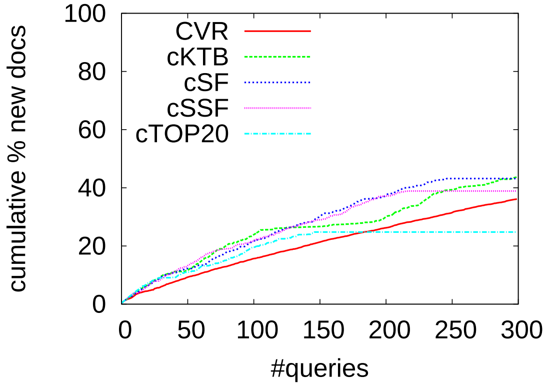

Figure 2: Harvest rate of approaches

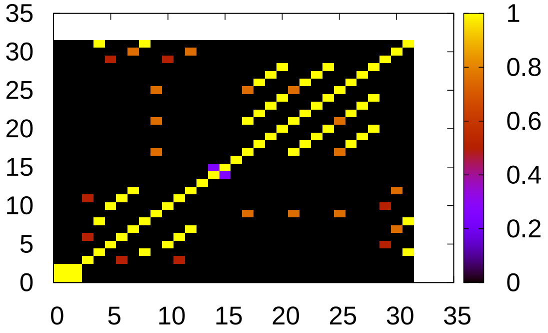

Figure 3: Probed queries correlation among configurations

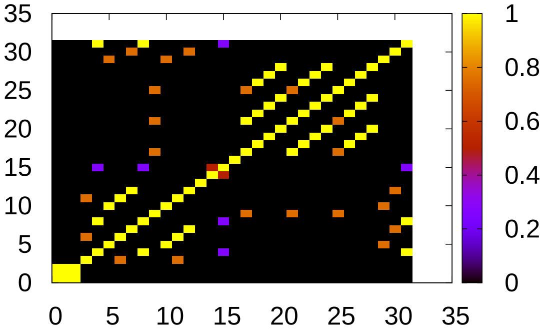

Figure 4: Retrieved documents correlation among configurations

lation. It is clear from Fig. 3, that certain configuration, not on the diagonal, exhibit a high correlation. These in fact use the same distance measure (e.g. KTb), with different weight w. Nevertheless, apart from these configurations, the rest have very low correlation (below 25% in most cases). This means that the query terms selected to probe the Hidden-Web site are entirely different, despite the fact that all configurations started with the same one.

Moreover, the fact that configurations probe with different queries is not sufficient on its own, as they could be retrieving similar documents. For this reason, Fig. 4 shows a similar map, measuring the correlation of retrieved documents by each configuration. Again, this is computed as the pairwise intersection of documents for all configurations. We can clearly see that unless the terms are the same, the configurations retrieve entirely different portions of the Hidden Web site. This is the main hint that using the ranking aspect results in a more *breadth*-wise search.

6. RELATED WORK

Our work is related to several fields pertaining the web, such as Web Mining and Web IR. More specifically, we focus on Hidden Web (HW), the existence of which was brought to light in the early years of this millennium [2]. Research on HW sites has focused on various aspects, such as understanding query forms [8, 16, 20], classifying the sites based on their content [11], accessing it [9, 4] and searching the surface web to discover HW entry points [18]. Our work focuses on surfacing HW content, i.e. retrieving content from those sites, so that it can be indexed, thereby it relates to the works in [14, 12, 13]. The general technique in these works, as in ours, is to probe queries to the HW site, retrieve the actual content and then select the next query with which to probe. These techniques focus on coverage, i.e. retrieve as big a portion of the site's content as possible. The key difference is that they do not take the ranking aspect into account.

Ranking has been extensively studied both within and outside the scope of web-related disciplines. Its importance is significantly higher in the web domain, due to the size of the web and the fact that users rarely view more than the top ranked documents. It is, therefore, an ever-going research topic, with various settings and applications. To rank documents on the web, several approaches have been proposed, such as machine learning [15], link structure [3], web content analysis and anchor text. Our work differs in that we do not aim at building a new ranking function for HW sites. Rather we are interested in taking ranking performed by HW sites into account, while retrieving their content. Ranking in the context of HW sites has been studied in [1], where the authors want to identify the order in which to probe sources, in order to provide users with appropriate information. Their work differs from ours in that we are interested in the ranking performed by HW sites *per se*, not to globally rank the HW sites themselves with respect to a user need. Moreover, the authors in [1] perform sampling of HW sites, as an external step of their approach, whereas our primary goal is to crawl HW sites and crawling is an integral part of our strategy.

Finally, our employed approach relates in notion to the active learning paradigm [17]. Active learning is based on the idea that a classifier "*may achieve higher accuracy if it is allowed to choose the next training label from which to learn*". To do this, a measure of the error that is introduced by a potential label is required. Then, the label that is expected to maximize the gain is selected and used to train the classifier. Although existing crawling algorithms for HW sites mostly rely on mathematical models to choose their next query, simplifying our technique's intuition, our work relates in the following manner: we choose the term for which we can not accurately predict its global ranking, in case we probed it to the database. Nevertheless, our work differs from active learning approaches, as we do not use a machine learning classifier, nor is it our primary concern to

Table 2: First 15 query terms by configuration

No Ranking	KTa	KTb	SF	SSF	Top-20
soup	soup	soup	soup	soup	soup
http	http	http	http	http	http
twitter	homemade	parents	annual	prepares	listening
www	doneness	retro	bold	coulis	sausage
youtube	panlasangpinoy	residents	casserole	fresco	nutrition
watch	margarine	traumatic	acidity	penne	steamed
follow	toweling	deborah	aparta	unbelievable	songifications
add	fashioned	ixzz	beet	mixer	wonton
video	foodwishes	casserole	antioxidants	shawtayee	siu
user	foodies	litre	absorbable	stvplayer	intestines
center	dmark	hing	acquire	sinatras	powders
subscription	swirls	hamburgers	antibacterial	larder	enhances
machinima	dmarkii	mattar	antiparasitic	broadcaster	intricate
tags	secretlifeofabionerd	sooji	antiinflammatory	gilbrook	craftsmanship
high	hungrynation	drizzle	acnes	tyres	luggage

learn the ranking function of each HW site. Moreover, we are interested in coverage as well. Finally, to the best of our knowledge, active learning approaches have not been used in the context of Hidden Web sites.

7. CONCLUSIONS AND FUTURE WORK

In this work we introduced the problem of crawling Hidden Web sites in a rank-aware manner, so that we make use of the ranking aspect that these sites perform. We proposed a family of algorithms which address this problem and are a generalization of existing approaches. We evaluated our techniques on real web sites. We show that our methods achieve better coverage of the Web site than existing methods and that overall, they visit different portions of the site, and thus are easily parallelizable. Future directions include the application of an active learning classifier on ranking functions and using the Hidden-Web site ranking as a building block to identify more accurately sources that fulfill user requirements. We are also interested in studying how the query terms used by each configuration correlate with topically-similar document clusters.

8. ACKNOWLEDGMENTS

This research has been co-financed by the EU and Greek national program “Heracleitus II” and by the SemSorGrid4Env (FP7-223913) and MODAP European Commission projects.

9. REFERENCES

[1] B. Arai, G. Das, D. Gunopulos, V. Hristidis, and N. Koudas. An access cost-aware approach for object retrieval over multiple sources. *PVLDB*, 3(1):1125–1136, 2010.

[2] M. Bergman. The deep web: Surfacing hidden value. Technical report, „ 2001.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 107–117, 1998.

[4] A. Dasgupta, G. Das, and H. Mannila. A random walk approach to sampling hidden databases. In *SIGMOD ’07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 629–640, 2007.

[5] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

[6] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing partial rankings. *SIAM J. Discret. Math.*, 20(3):628–648, 2006.

[7] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *SODA ’03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 28–36, 2003.

[8] H. Garcia-Molina. Challenges in crawling the web. In *BNCOD*, page 3, 2003.

[9] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. *Commun. ACM*, 50(5):94–101, 2007.

[10] P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. Towards a query optimizer for text-centric tasks. *ACM Trans. Database Syst.*, 32(4):21, 2007.

[11] P. G. Ipeirotis, L. Gravano, and M. Sahami. Probe, count, and classify: categorizing hidden web databases. In *SIGMOD ’01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 67–78, 2001.

[12] J. Liu, Z. Wu, L. Jiang, Q. Zheng, and X. Liu. Crawling deep web content through query forms. In *WEBIST*, pages 634–642, 2009.

[13] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Googleś deep web crawl. *Proc. VLDB Endow.*, 1(2):1241–1252, 2008.

[14] A. Ntoulas, P. Zerfos, and J. Cho. Downloading textual hidden web content through keyword queries. In *JCDL*, pages 100–109, 2005.

[15] M. Richardson. Beyond pagerank: Machine learning for static ranking. In *In WWW âĂŹ06: Proceedings of the 15th international conference on World Wide Web*, pages 707–715. ACM Press, 2006.

[16] P. Senellart, A. Mittal, D. Muschick, R. Gilleron, and M. Tommasi. Automatic wrapper induction from hidden-web sources with domain knowledge. In *WIDM ’08: Proceeding of the 10th ACM workshop on Web information and data management*, pages 9–16, 2008.

[17] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[18] K. Vieira, L. Barbosa, J. Freire, and A. S. da Silva. Siphon++: a hidden-webcrawler for keyword-based interfaces. In *CIKM*, pages 1361–1362, 2008.

[19] YouTube Service. http://www.youtube.com/.

[20] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: best-effort parsing with hidden syntax. In *Proc. of the 2004 ACM SIGMOD Int. Conf. on Management of data*, SIGMOD ’04, pages 107–118, 2004.