Web Page Clustering using Heuristic Search in the Web Graph

Ron Bekkerman
University of Massachusetts
Amherst MA, USA
ronb@cs.umass.edu

Shlomo Zilberstein
University of Massachusetts
Amherst MA, USA
shlomo@cs.umass.edu

James Allan
University of Massachusetts
Amherst MA, USA
allan@cs.umass.edu

Abstract

Effective representation of Web search results remains an open problem in the Information Retrieval community. For ambiguous queries, a traditional approach is to organize search results into groups (clusters), one for each meaning of the query. These groups are usually constructed according to the topical similarity of the retrieved documents, but it is possible for documents to be totally dissimilar and still correspond to the same meaning of the query. To overcome this problem, we exploit the thematic locality of the Web—relevant Web pages are often located close to each other in the Web graph of hyperlinks. We estimate the level of relevance between each pair of retrieved pages by the length of a path between them. The path is constructed using multi-agent beam search: each agent starts with one Web page and attempts to meet as many other agents as possible with some bounded resources. We test the system on two types of queries: ambiguous English words and people names. The Web appears to be tightly connected; about 70% of the agents meet with each other after only three iterations of exhaustive breadth-first search. However, when heuristics are applied, the search becomes more focused and the obtained results are substantially more accurate. Combined with a content-driven Web page clustering technique, our heuristic search system significantly improves the clustering results.

1 Introduction

Clustering of Web search results has been in the focus of the information retrieval community since the early days of the Web [Hearst and Pedersen, 1996; Zamir and Etzioni, 1998]. The reasons for clustering of search results are two-fold. The first is that the IR research community has long recognized the validity of the *cluster hypothesis* [van Rijsbergen, 1971] in top-ranked documents, i.e. similar documents tend to be relevant to the same requests. A second (related) reason is that the ranked list is usually too large and contains many documents that are irrelevant to the particular meaning of the query the user had in mind. Thus, it would be beneficial to group search results by various meanings of the query. Recently, successful academic [Zeng *et al.*, 2004] and industrial (`vivisimo.com`) attempts have made the clustering of search results plausible for many WWW users. However, it is still not accurate enough to attract an average user. The main drawback of many Web page clustering methods is that they take into account only the *topical* similarity between documents in the ranked list.

Topical similarity metrics between Web pages would not help solving the clustering problem in at least two cases: (a) when there is not enough contextual information on a page: for example, within 20 first hits on a query `jaguar` one can find a Web site `savethejaguar.com`, which presents a large picture of the wild cat on the background, but does not contain enough topical words to automatically associate the page to the correct group; (b) when Web sites are contextually different but actually refer to the same meaning of the query. For instance, given a query `Michel Décary`, one can retrieve Web pages of at least three individuals: a computer scientist (`www.zoominfo.com/MichelDecary`), a lawyer (`www.stikeman.com/cgi-bin/profile.cfm?P_ID=366`), and a chansonnier (`www.decary.com`). All three studied at the University of Montréal and at McGill University in Canada. Are they three different people or actually one person?

These problems can be resolved by exploiting the *thematic locality* of the Web graph (the directed graph in which nodes are Web pages and edges are hyperlinks). Hypothetically, if page A hyperlinks page B, then the creator of page A *intentionally* raised the topic of page B in the context of page A which indicates that pages A and B are semantically close. Davison [2000] empirically justifies this hypothesis. In an average case, if two static Web pages are located in a short proximity to each other in the Web graph, then they stand in a (probably vague) semantic relation. For the two examples presented above, the site `savethejaguar.com` hyperlinks `wcs.org`, the Wildlife Conservation Society, which reveals its topic; while the pages of `Michel Décary` the scientist and the chansonnier point to `cogilex.com`, Michel's previous enterprise, which implies that two of the three individuals are in fact the same person. Note that no language modeling method would resolve this dilemma because the two pages are strictly different (they are even written in different languages).

Link analysis has been successfully applied to various Web mining tasks. A related task is identification of Web communities (see, e.g., Gibson *et al.* 1998), which are defined as heavily connected components of the Web graph. Research methods for this task are primarily graph-theoretic (graph partitioning, network flow etc.). Unfortunately, these methods are inapplicable to our task, because we cluster *isolated*, unconnected Web pages retrieved by an arbitrary query. The most relevant previous work is He *et al.* [2002], who build a Web page clustering system that exploits the hyperlink structure of the Web: they consider two Web pages to be similar if they are in parent/child or sibling relations in the Web graph. We propose a more general framework that incorporates both topical and topological closeness: Web pages belong to the same cluster if they are similar in content *or* close to each other in the Web graph.

To approximate the distance between two pages in the Web graph, we apply the *heuristic search* paradigm [Pearl, 1984]. To our knowledge, this paper is the first work that applies heuristic search (specifically, beam search) in the domain of the Web graph. Heuristics have been used for focused Web crawling (e.g. [Davidov and Markovitch, 2002]), where the goal is to collect as much useful information as possible while crawling the Web, and the heuristics estimate the amount of information available in a particular Web sub-graph. In contrast, we use heuristics to estimate the utility of expanding the current node in terms of leading to the *target node*.

Since the heuristic search can be computationally hard, we perform *bidirectional* search: we start searching from both source and target nodes and expand hyperlinked nodes until the two search frontiers meet at a common node or until the resources are depleted. In this setting, the computational complexity is no longer an issue: after only three search iterations, we can construct paths of length up to 8,[1] which are long enough to potentially diminish any semantic relation between the starting nodes. Thus, since short searches are acceptable in our case and since the out-degree of Web pages is on average just about 8 [Kleinberg *et al.*, 1999], even exhaustive breadth-first search methods are feasible. Moreover, modern search engines store the adjacency table of most of the Web, i.e. no Web crawling is required for the heuristic search. We use heuristics not to reduce the search time, but to improve the search *accuracy*. As we discuss below, the modern Web is tightly interconnected, so heuristics are used as filters to prune branches of search trees that are likely to establish undesired connections between unrelated Web pages.

To distribute the heuristic search, we build a multi-agent system: given n Web pages in the ranked list, we construct n *collaborative* Web agents each of which is assigned one page of the initial dataset. Each agent then performs heuristic search to traverse the Web graph in order to meet as many other agents as possible. If an agent reached a dead end and cannot continue the search, it can move up in the hierarchy of Web directories which would presumably lead to a more general page that has more hyperlinks. Pages whose agents managed to meet within the given budget on resources are then placed in the same cluster. By this, we construct a set C of $k \ll n$ topological clusters (we consider only k largest clusters of those constructed). In parallel to that, we apply a traditional topical clustering method that assigns each document from the original ranked list into one cluster from a set C' of $k' > k$ topical clusters. After that, for each cluster $c_i \in C$ we find its closest cluster c'_j from C': $j = \arg\max_{j'} |c_i \cap c'_{j'}|$. Each cluster c_i is then enriched with elements of c'_j that do not appear in any cluster of C. By this, we construct larger clusters c_i that contain documents that are either topologically or topically related. This technique shows excellent results both in terms of precision and recall (see Section 4).

Besides the clustering of Web search results, the proposed system can be applied to various information retrieval and Web mining tasks, such as Web appearance disambiguation [Bekkerman and McCallum, 2005], acronym disambiguation [Pakhomov *et al.*, 2005], interactive information retrieval [Leuski and Allan, 2004], Web search with Web pages provided as queries [Dean and Henzinger, 1999], as well as for Homeland security analysis and other important problems.

In this study, we test our system on two applications: search result clustering and Web appearance disambiguation. In the latter, the goal is to identify Web appearances of particular people with potentially ambiguous names. The problem is solved given *a few* names of people who are known to belong to the same social network. Disambiguation of each person's name is allowed by the presence of other names that are likely to correlate with it. We represent Web appearance disambiguation as a special case of the search result clustering task: given m queries of people names, we construct *one* cluster of Web pages that mention the people of our interest, while disregarding pages that mention their unrelated namesakes. We generalize our multi-agent heuristic search by constructing $m \times n$ agents that search for each other in the Web.

2 Multi-agent heuristic search

We propose two multi-agent heuristic search algorithms for topologically clustering n pages (we call them *source pages*). Algorithm 1 is called Sequential Heuristic Search (SHS). We start with n singleton clusters of the source pages. We create a collection of n Web agents each of which is assigned one source page. Each agent maintains a *search frontier*: a list of nodes (URLs) to be expanded (initially, the URL of its source page). At any search iteration each agent obtains URLs hyperlinked from the nodes of its search frontier. It then applies heuristics to select potentially good URLs to become its new search frontier. After that, we intersect the sets of URLs obtained by all agents. If a common URL is found for two source pages, we merge the clusters they belong to. The system stops after a predefined number of iterations (usually, a small number of 2 or 3, as discussed in Section 1).

The SHS algorithm, while being simple and intuitive, suffers from one crucial drawback: there is no possibility to control the topology of the constructed clusters. In a worst case, after l search iterations, if a path is found between page A and B, as well as between pages B and C, and between pages C and D (while no other links are found), then pages A and D

[1] Starting with nodes A_0 and B_0, after three search iterations the following path of length 8 can be constructed: $A_0 \rightarrow A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow C \leftarrow B_3 \leftarrow B_2 \leftarrow B_1 \leftarrow B_0$.

Input:
$S = \{s_1, s_2, \ldots, s_n\}$ – URLs of source pages
l – number of search iterations
Output:
Clusters $C_1, \ldots, C_k$

For each $s_i \in S$ **do**
  Initialize agent a_i's search frontier $F_0(a_i) \leftarrow \{s_i\}$
  Initialize agent a_i's set of extracted URLs $T_0(a_i) \leftarrow \{s_i\}$
For each $j = 0, \ldots, l$ **do**
  ———***Distributed search phase:***———
  For each $s_i \in S$ **do**
    Construct $F_{j+1}(a_i) \leftarrow$ **Extract_URLs**$(F_j(a_i))$
    Filter $F_{j+1}(a_i)$ using a set of heuristics
    Update $T_{j+1}(a_i) \leftarrow T_j(a_i) \cup F_{j+1}(a_i)$
  ———***Result collection phase:***———
  Construct all pairs $(s_i, s_{i'})$ s.t. $T_{j+1}(a_i) \cap T_{j+1}(a_{i'}) \neq \emptyset$
  Initialize singleton clusters $C_i \leftarrow \{s_i\}$
  For each pair $(s_i, s_{i'})$ **do**
    If $(s_i \in C_t) \wedge (s_{i'} \in C_{t'}) \wedge (C_t \neq C_{t'})$ **then**
      Merge C_t and $C_{t'}$

Algorithm 1: Sequential Heuristic Search (SHS).

Input:
$S = \{s_1, s_2, \ldots, s_n\}$ – URLs of source pages
$CC_1, \ldots, CC_k$ – core clusters obtained at iteration 0 of SHS
l – number of search iterations
Output:
Enlarged core clusters $CC_1, \ldots, CC_k$

For each $s_i \in S$ **do**
  Initialize agent a_i's frontier $F_1(a_i) \leftarrow$ **Extract_URLs**(s_i)
  Initialize a_i's set of extracted URLs $T_1(a_i) \leftarrow \{s_i\} \cup F_1(a_i)$
For each $j = 1, \ldots, l$ **do**
  ———***Distributed search phase:***———
  For each $s_i \in S$ **do**
    Construct $F_{j+1}(a_i) \leftarrow$ **Extract_URLs**$(F_j(a_i))$
    Filter $F_{j+1}(a_i)$ using a set of heuristics
    Update $T_{j+1}(a_i) \leftarrow T_j(a_i) \cup F_{j+1}(a_i)$
  ———***Result collection phase:***———
  Construct all pairs $(s_i, s_{i'})$ s.t. $(T_{j+1}(a_i) \cap T_{j+1}(a_{i'}) \neq \emptyset) \wedge$
    $(\exists t : s_i \in CC_t) \wedge (\forall t' : s_{i'} \notin CC_{t'})$
  For each pair $(s_i, s_{i'})$ **do**
    Add $s_{i'}$ to CC_l

Algorithm 2: Incremental Heuristic Search (IHS).

will be placed in the same cluster despite that the semantic relation between them is probably weak, as their distance in the Web can be $6l$,[2] which is too long even if $l = 2$. A method for building tightly connected clusters should be proposed.

Solving the Web appearance disambiguation problem, Bekkerman and McCallum [2005] noticed that matching hyperlinks of the source pages leads to a small but clean cluster of relevant pages (called the *core cluster*). We adopt this idea and propose another multi-agent heuristic search algorithm, called Incremental Heuristic Search (IHS)—see Algorithm 2. In IHS, we start with a set of core clusters generated at iteration 0 of SHS. [3] The distributed search phase of IHS is exactly the same as of SHS, but at the result-collection phase we now select only pairs where one member belongs to a core cluster while the other does not, so we add it to the corresponding core cluster. We ignore pairs in which both members belong to different core clusters. Proceeding incrementally, we keep track of the diameter of each constructed cluster, which is now independent of the cluster's size.

2.1 Useful heuristics

Two types of heuristics can be proposed in the Web domain: *topology-driven* and *content-driven*. Topology-driven heuristics are based on the layout of the Web graph, while content-driven heuristics are based on features extracted from the interior of Web pages. In this section we propose one topology-driven and two content-driven heuristics, all of which are fairly straightforward, but still prove to be effective when used in our framework of heuristic search in the Web graph. In our future work, we will explore other heuristics as well.

[2]Since our search is bidirectional, after l iterations a hyperlink path of length up to $2l$ can be constructed.

[3]We do not attempt to solve the fundamental problem of inferring the correct number of clusters. Instead, we preset this number for each particular task: for Web appearance disambiguation, only one core cluster is needed, while for clustering Web search results, the number of clusters equals the number of main meanings of the query.

Our topology-driven heuristic is *high-degree node elimination* (or, in short, *high-degree heuristic*): after each search iteration, from the search frontiers we remove high out-degree and high in-degree URLs that often connect between semantically unrelated pages. For example, both `macromedia.com` and `historians.org` point to `google.com`, which does not imply that there is a tight semantic relation between Macromedia Inc. and the American Historical Association. For the graphical interpretation of the high-degree heuristic, see Figure 1(a). To detect high out-degree URLs, we simply count the number of hyperlinks at each page. To detect high in-degree URLs, we use Google's `link:` operator.

A successful content-driven heuristic is the *person name heuristic*. Figure 1(b) illustrates the idea. An agent has a good chance to meet another agent, if it expands a page that shares a person name with a page expanded by another agent. To extract person names from expanded Web pages, we first remove markup, and then apply NER (Wei Li's named entity tagger, see McCallum and Li, 2003). We extract only entities tagged as `PERSON` and consider people names that consist of two, three or four words. We exclude people names that are too common (again, we use Google's `link:` operator).[4]

In analogy to the person name heuristic, we also propose the *anchor text heuristic* that matches snippets of anchor text extracted from the Web pages. We ignore too common anchor texts, such as `contact us` or `copyright`. Eiron and McCurley [2003] perform a comprehensive analysis of anchor texts and show that they usually summarize the content of the hyperlinked Web pages. Such summarization can be very useful in our case, when we attempt to predict a possible benefit of expanding pages from the search frontier. Note that in contrast to people names, anchor snippets can be easily identified by shallow parsing of the pages' markup language.

[4]In many cases, an entity tagged as a person name has millions of Google's hits if it is a tagger error. Examples of such entities are `Price Range` and `Mac Os`.

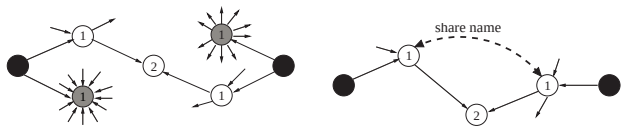

Figure 1: An illustration of applying heuristics, at the first search iteration. Black nodes are the source pages. **(left) High-degree heuristic.** Gray nodes are high in- or out-degree pages, eliminated from search frontiers. **(right) Person name heuristic.** A hyperlink path between two nodes is constructed over which a person name is also shared.

Category	# of pages	Category	# of pages
Car	36	Cornell project	2
Mac OS	11	Metal Band	1
Wild cat	23	Movie	1
Biotech firm	2	Photo gallery	1
Youth org	1	Atari game	5
Maya culture	1	Guitar	1
Resin models	1	TV channel	1
Web hosting	1	Web designer	2
Reef lodge	2	E-commerce firm	1
Book	1	Game archieve	1
Singer	2	Aircraft	1
Emulator	2		

Table 1: Statistics on the Jaguar dataset.

3 Datasets

We use two datasets for evaluation of our methods: one for Web appearance disambiguation and another for clustering Web search results.

3.1 Web appearance disambiguation dataset

We downloaded Bekkerman's Web appearance disambiguation dataset from `www.cs.umass.edu/~ronb`. It consists of 1085 Web pages retrieved on 12 names of people from Melinda Gervasio's social network (mostly, SRI engineers and university professors). The dataset is labeled according to the person's occupation. Two of the 12 people appear to be unique in the Web, while the rest have relatively common names. Some of the names are extremely ambiguous, e.g. given a query `''Tom Mitchell''`, 37 different Tom Mitchells are found within the first 100 Google hits. The dataset contains pages of 187 unique people overall, while only 12 of them are relevant (mentioned at 420 pages). For the statistics on the dataset as well as for the preprocessing procedure, see Bekkerman and McCallum [2005].

We crawled the Web starting with these 1085 pages (*source pages*). We retrieved all available pages hyperlinked from the source pages, as well as the pages located one level above the source pages in the hierarchy of Web directories. We continued this process until all the pages within three hops of the original dataset were retrieved. In order not to produce a priori weak connections and to still preserve a reasonable size for our dataset, we did not retrieve pages located at extremely popular domains, such as `amazon.com`. We also ignored pages of non-textual format. At each crawling iteration our dataset grew almost an order of magnitude: we downloaded 7009 pages at the first hop, 69,454 pages at the second hop and 592,299 pages at the third hop, resulting in 669,847 unique Web pages overall.

3.2 Jaguar dataset

We built a new dataset for the problem of clustering Web search results. We retrieved and labeled 100 first Google hits obtained on the query `jaguar`. We found 23 different categories within the 100 retrieved pages: the largest ones are obviously the car, the wild cat and the Mac operating system (version 10.2). Table 1 presents statistics on this dataset.

Exactly as for the Web appearance disambiguation dataset, we crawled three hops off the Jaguar source pages, retrieving 883 pages on the first hop, 8548 pages on the second hop and 56,287 pages on the third hop. At each iteration the dataset grew on average by a factor of 8, which corresponds surprisingly well to the growth of the Web appearance disambiguation dataset and to the findings of Kleinberg *et al.* [1999].

4 Results and discussion

4.1 Web appearance disambiguation

First, we apply both sequential and incremental search algorithms on the Web appearance disambiguation data in an exhaustive manner, i.e. without applying heuristics. Surprisingly, we discover that the dataset is heavily interconnected. After each iteration of the sequential search, the connected pages compose one large cluster of size 208, 543, 728, and 786 (72.5% of the entire dataset) respectively. Some connections are extremely weak: 10% of 66,561 hyperlink paths found at the last iteration go through `www.adobe.com/products/acrobat/readstep2.html`, a page with over 600,000 Google hits on it.

On this data, we report on precision, recall and F-measure of constructing *one* cluster of documents that mention relevant people. Precision/recall curves in Figure 2 show that the exhaustive sequential and incremental algorithms do quite poorly on this data, with slight advantage to the incremental approach. After four iterations, we end up with above 80% recall, but the precision is very low (under 50%). However, the performance is improved when we apply the high-degree heuristic. We set the threshold of in/out hyperlinks at 1000—all pages with more than 1000 Google hits and pages containing more than 1000 hyperlinks are filtered out. We also tried other thresholds, such as 100 and 10,000, without any significant change in the performance. Note that only short paths are effective: the precision drops at the second and third hops of the source pages. The reason for such a drop is that the high-degree heuristic is *topology-driven*—it ignores the content of the pages, which introduces a lot of noise while moving far away from the source pages.

The person name heuristic turned out to be more effective. We notice that since we perform short searches (up to three hops from the source pages), there is no need in narrowing the search beam with the heuristic. Moreover, such narrowing may hurt the recall of our system. Instead, we apply the heuristic as a filtering method: at each iteration

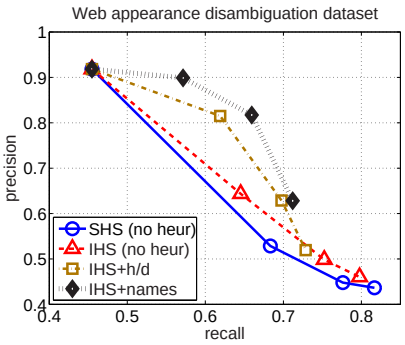

Figure 2: Precision/Recall curves for four algorithms on the Web appearance disambiguation dataset. *h/d* means high-degree heuristic, *names* means person name heuristic. Four nodes in each curve correspond to search iterations 0, 1, 2 and 3. At iteration 0 (only original nodes expanded) the core cluster is built (for the IHS algorithm).

j, we first use an incremental exhaustive search in order to find pages that are linked with the core cluster, and then we apply our Information Extraction module that extracts people names from pages expanded during the search. For each source page s_i we build two sets: $T_j(s_i)$ of all URLs found during the search and $N_j(s_i)$ of all people names extracted from the search tree. For the core cluster CC we construct $T_j(CC) = \bigcup_{s_i \in CC} T_j(s_i)$ and $N_j(CC) = \bigcup_{s_i \in CC} N_j(s_i)$. We put page s_i into CC if there is a hyperlink path from s_i to CC *and* a common person name is found: $(T_j(s_i) \cap T_j(CC) \neq \emptyset) \wedge (N_j(s_i) \cap N_j(CC) \neq \emptyset)$. Note that the only difference from Algorithm 2 is that the common person name may not be *on* the constructed hyperlink path between s_i and CC. This method shows good results on our data (see Figure 2). The best F-measure (73%) is achieved at the second iteration, while at the third one the precision drops by almost 20%, which implies that two iterations are enough. We also tried to apply the high-degree and the person name heuristics together, but did not see any improvement in precision, while hurting recall.

To compare our results with the ones reported by Bekkerman and McCallum [2005], we use their topical clustering method called Agglomerative/Conglomerative Distributional Clustering (A/CDC), which is a state-of-the-art information-theoretic technique. The results are shown in Table 2 (A/CDC result is by Bekkerman and McCallum, 2005). We see that after the first iteration the heuristic search method is competitive with A/CDC in precision, but is inferior in recall. However, when combining the two methods, we obtain excellent results in terms of both precision and recall. After the second search iteration the precision trades off against the recall (more noise is added) and the F-measure slightly decreases.

Heuristic search allows addition of 49 previously undiscovered documents to the topical cluster, 32 of which refer to Adam Cheyer and Steve Hardt. Bekkerman and McCallum [2005] notice that these two researchers work in industry so their pages use different vocabulary than most of other academic-style pages in the cluster. Our heuristic search method is especially designed to overcome this problem.

Method	Precision	Recall	F-measure
Web appearance disambiguation			
Topical (A/CDC)	87.3%	71.3%	78.4%
IHS (iteration 1)	89.9%	57.1%	69.9%
Hybrid (iteration 1)	84.5%	83.3%	**83.9%**
IHS (iteration 2)	81.7%	66.0%	73.0%
Hybrid (iteration 2)	78.5%	86.2%	82.2%
Clustering of Web search results			
Topical (A/CDC)	75.0%	64.3%	69.2%
IHS (iteration 1)	93.3%	40.0%	56.0%
Hybrid (iteration 1)	77.1%	77.1%	**77.1%**
IHS (iteration 2)	78.6%	47.1%	58.9%
Hybrid (iteration 2)	72.7%	80.0%	76.2%

Table 2: Results of topical clustering (A/CDC), topological clustering (IHS) and their hybrid, on two datasets. The IHS clustering (and the hybrid) results are obtained after the first and second iterations of heuristic search (hyperlink paths of length up to 4 and up to 6 respectively).

4.2 Clustering of Web search results

In contrast to Web appearance disambiguation, the problem of clustering Web search results is not a one-class problem. We evaluate our system on k largest classes of the data. For our Jaguar dataset we chose $k = 3$, so we build three clusters (of cars, Mac OS, and wild cats). Let CC_i be one of these clusters and Cl_i be its corresponding class. Let $Corr_i$ be a set of pages from Cl_i that have been correctly assigned into CC_i by our system. Then the micro-averaged precision and recall of the system are:

$$Prec = \frac{\sum_{i=1}^{k} |Corr_i|}{\sum_{i=1}^{k} |CC_i|}; \quad Rec = \frac{\sum_{i=1}^{k} |Corr_i|}{\sum_{i=1}^{k} |Cl_i|}.$$

On the Jaguar dataset, the sequential exhaustive search fails: after three iterations, 70 of the 100 pages are all connected together. However, the incremental algorithm shows better results (see Figure 3): at the first iteration it obtains 82.4% precision but then the precision drops. When applying the high-degree heuristic (with the threshold at 10000 hyperlinks), the result is even better, especially after the first iteration (93.3% precision). We use the three clusters constructed at this iteration as the core clusters (instead of using the core clusters constructed at the previous iteration—this design choice leads to a higher recall), and add the anchor text heuristic (which improves the precision). The resulting system demonstrates good performance, while the F-measure is consistently improved from 56% to 59% and then to 62% at the third hop from the source pages. This is the only result we could obtain that shows usefulness of expanding pages at the third hop.

When comparing the heuristic search method with topical clustering, we observe exactly the same trend as for the Web appearance disambiguation task (see Table 2). The best performance (77.1% F-measure) is obtained by the combination of the two methods after the first heuristic search iteration, which is a strong result for an unsupervised method on a multi-class task.

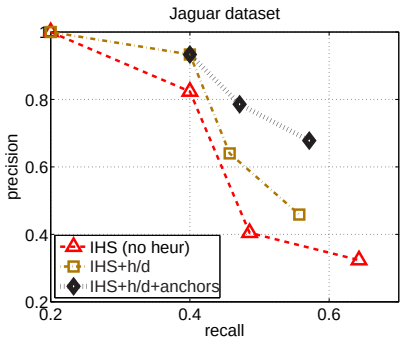

Figure 3: Precision/Recall curves on the Jaguar dataset (the three-class problem of recognizing pages related to cars, Mac OS and wild cats). *h/d* means high-degree heuristic, *anchors* means anchor text heuristic. Four nodes in the curves of IHS and IHS+h/d correspond to search iterations 0, 1, 2 and 3. For IHS+h/d+anchors only 3 iterations are performed, with the clusters of IHS+h/d (iteration 1) taken as its core clusters.

5 Conclusion and future work

To our knowledge, this paper is the first study of heuristic search in the Web graph. The proposed framework is highly promising: various information retrieval and Web mining tasks can be tackled in this framework. The main contribution of this paper is thus in making the heuristic search viable in the vast domain of the WWW and applicable to clustering of Web search results and to Web appearance disambiguation.

We show that the Web is highly interconnected: heuristics are used not to seek hyperlink paths but rather to prune many irrelevant ones. Despite that we obtain good results with our heuristic search method, it is still inferior to a state-of-the-art machine learning topical clustering technique. However, the two methods find different types of connections between Web pages. We empirically prove that the highest benefit is in *combination* of topological and topical clustering methods that demonstrates commercially acceptable performance.

Clustering Web pages using heuristic search in the Web graph might be considered burdensome but it actually is not. Modern search engines store link structure of a large part of the Web, so neither retrieval nor parsing of Web pages should be performed in real time. Since bidirectional search for hyperlink paths between clustered pages is applied, only one or two (maximum three) search iterations are usually enough to construct meaningful clusters. The process is fully distributed so the map-reduce paradigm [Dean and Ghemawat, 2004] can be employed. While the person name heuristic can be difficult to compute, the other two heuristics proposed (high-degree and anchor text) are straightforwardly applicable.

The framework of heuristic search in the Web graph poses a wide variety of interesting research problems. How to adapt heuristic search to various real-world tasks? Which heuristics are the best for these tasks? Could heuristics estimate the *distance* between two nodes in the Web graph? Which search control strategies should be played by multiple agents while exploring the Web graph? Our results open up a range of theoretical and practical opportunities yet to be addressed.

6 Acknowledgments

We thank Tsuyoshi Murata and Kevin McCurley for fruitful discussions. This work was supported in part by the Center for Intelligent Information Retrieval and in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

[Bekkerman and McCallum, 2005] R. Bekkerman and A. McCallum. Disambiguating Web appearances of people in a social network. In *Proceedings of WWW*, pages 463–470, 2005.

[Davidov and Markovitch, 2002] D. Davidov and S. Markovitch. Multiple-goal search algorithms and their application to Web crawling. In *Proceedings of AAAI*, pages 713–718, 2002.

[Davison, 2000] B. D. Davison. Topical locality in the Web. In *Proceedings of SIGIR*, pages 272–279, 2000.

[Dean and Ghemawat, 2004] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In *Proceedings of Operating System Design and Implementation*, 2004.

[Dean and Henzinger, 1999] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks*, 31:1467–1479, 1999.

[Eiron and McCurley, 2003] N. Eiron and K. McCurley. Analysis of anchor text for Web search. In *Proceedings of SIGIR*, pages 459–460, 2003.

[Gibson *et al.*, 1998] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring Web communities from link topology. In *Proceedings of the 9th ACM conference on Hypertext and Hypermedia*, 1998.

[He *et al.*, 2002] X. He, H. Zha, C. H. Q. Ding, and H. D. Simon. Web document clustering using hyperlink structures. *Computational Statistics & Data Analysis*, 41:19–45, 2002.

[Hearst and Pedersen, 1996] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR*, pages 76–84, 1996.

[Kleinberg *et al.*, 1999] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph: Measurements, models and methods. *LNCS*, 1627, 1999.

[Leuski and Allan, 2004] A. Leuski and J. Allan. Interactive information retrieval using clustering and spatial proximity. *User Modeling and User-Adapted Interaction*, 14:259–288, 2004.

[McCallum and Li, 2003] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and Web-enhanced lexicons. In *Proc. of CoNLL*, 2003.

[Pakhomov *et al.*, 2005] S. Pakhomov, T. Pedersen, and C. Chute. Abbreviation and acronym disambiguation in clinical discourse. In *Proceedings of the Annual Symposium of the American Medical Informatics Association*, 2005.

[Pearl, 1984] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[van Rijsbergen, 1971] C. J. van Rijsbergen. An algorithm for information structuring and retrieval. *Computer Journal*, 14:407–412, 1971.

[Zamir and Etzioni, 1998] O. Zamir and O. Etzioni. Web document clustering: a feasibility demonstration. In *Proc. of SIGIR*, 1998.

[Zeng *et al.*, 2004] H.U. Zeng, Q.C. He, Z. Chen, W.Y. Ma, and J. Ma. Learning to cluster Web search results. In *Proceedings of SIGIR*, 2004.