Adaptive Web Sites: Conceptual Cluster Mining

Mike Perkowitz Oren Etzioni
Department of Computer Science and Engineering, Box 352350
University of Washington, Seattle, WA 98195
{map, etzioni}@s.washington.edu
(206) 616-1845 Fax: (206) 543-2969

Abstract

The creation of a complex web site is a thorny problem in. user interface design. In IJCAI '97, we challenged the AI community to address this problem by creating *adaptive web sites.* In response, we investigate the problem of *index page synthesis* — the automatic creation of pages that facilitate a visitor's navigation of a Web site. Previous work has employed statistical methods to generate candidate index pages that are of limited value because they do not correspond to concepts or topics that are intuitive to people. In this paper we formalize index page synthesis as a conceptual clustering problem and introduce a novel approach which we call *conceptual cluster mining:* we search for a small number of cohesive clusters that correspond to concepts in a given concept description language *L.*

Next, we present SGML, an algorithm schema that combines a statistical clustering algorithm with a concept learning algorithm. The clustering algorithm is used to generate seed clusters, and the concept learning algorithm to describe these seed clusters using expressions in *L.* Finally, we offer preliminary experimental evidence that instantiations of SGML outperform existing algorithms (e.g., COBWEB) in this domain.

1 Introduction

Designing a complex web site so that it readily yields its information is tricky First, different visitors have distinct goals. Second, the same visitor may seek different information at different times. Third, many sites outgrow their original design, accumulating links and pages in unlikely places. Fourth, a site may be designed for a particular kind of use, but may be used in many different ways in practice; the designer's *a priori* expectations may be violated. Too often web site designs are fossils cast in HTML, while web navigation is dynamic, time-dependent, and idiosyncratic.

In [Perkowitz and Etzioni, 1997], we challenged the AI community to address this problem by creating adaptive web sites: *sites that automatically improve their organization and presentation by learning from visitor access patterns.* Many AI advances, both practical and theoretical, have come about in response to such challenges. The quest to build a chess-playing computer, for example, has led to many advances in search techniques (e.g., [Anantharaman *et al*., 1990]). Similarly, the autonomous land vehicle project at CMU [Thorpe, 1990] resulted not only in a highway-cruising vehicle but also in breakthroughs in vision, robotics, and neural networks. We believe that the adaptive web sites challenge will also drive AI advances.

Much of the previous work on adaptive web sites has focused on fairly simple adaptations (e.g., automatically creating shortcuts in the site) and on *customization* —- "personalizing" a web site to suit the needs of each individual user. In contrast, our own work has been motivated by two goals: first, we seek to demonstrate that relatively sophisticated adaptations can be generated; second, we seek to aggregate information gleaned from a population of users to *transform* the site — altering it to make navigation easier for a large number of users. These goals have led us to investigate the problem of *index page synthesis:* the automatic creation of navigational pages that consist of a comprehensive set of links to pages at the site on a particular topic (e.g., an index page on "electric guitars" at a musical instruments web site).

In [Perkowitz and Etzioni, 1998], we formally defined the *index page synthesis problem* and presented an algorithm called PageGather for discovering candidate link sets which would form the basis for new index pages, based on visitor access logs. In that paper, we compared PageGather to classical clustering algorithms [Voorhees, 1986; Rasmussen, 1992; Willet, 1988] and to Apriori, the classical data mining algorithm for the discovery of frequent sets [Agrawal and Srikant, 1994] using the "cohesiveness" of the link sets generated as the basis for comparison (see Section 4 for a precise definition of our measure). We found that PageGather produced substantially better candidates in our domain. Surprisingly, we also found that PageGather's candidates were better than human-authored index pages available at our experimental test site.

More detailed examination of the data reveals the reason for PageGather's "super-human" performance: human index page authors operate under a constraint that

Given:

1. A data collection *D* (e.g., a set of pages at a web site).
2. A pairwise similarity measure m defined over *D* (e.g., page co-occurrence frequencies derived from access logs).
3. A conceptual language *L* for describing elements of *D* (e.g., conjunctions of descriptive features).
4. A description in *L* of each object in *D.*

Output all subsets *c* of *D* such that

1. c is highly cohesive with respect to *m* (e.g., the average pairwise similarity of objects in *c* exceeds some threshold)
2. *c* corresponds to a concept describable in *L.*

Figure 1: The Conceptual Cluster Mining Problem. Given a data collection, a similarity measure, and conceptual descriptions of data objects, find sets of objects that are both similar and conceptually related.

PageGather ignored for successful navigation, index pages typically correspond to a topic or concept that is intuitive to visitors; they cannot be composed of a cohesive, but arbitrary, set of links. For example, if the topic of the index page is "electric guitars", the page should contain no links to information about pianos or drums, and there should be no local pages about electric guitars that are not linked to from the index page.

PageGather relies on a statistical approach to discovering candidate link sets; its candidates do not correspond precisely to intuitive concepts, whereas human-authored index pages do. In this paper, we present a key extension to PageGather that guarantees that PageGather will generate only link sets that correspond to topics, given a language for describing topics. Our extension also constitutes a novel approach to the long-standing problem of conceptual clustering [Michalski and Stepp, 1983].

This paper is organized as follows. We first define the conceptual cluster mining problem and discuss previous work. Next, we present the SGML algorithm schema and discuss several instantiations. We experimentally evaluate instantiations of SGML and compare them to the COBWEB conceptual clustering algorithm and to human-authored clusters. We then discuss ways to automatically generate the object descriptions that conceptual clustering algorithms take as input. We conclude with a discussion of contributions and future work.

2 Conceptual Cluster Mining

In this section, we define the conceptual cluster mining problem, describe previous work, and present our own approach.

2.1 Problem Definition

In [Perkowitz and Etzioni, 1998], we presented a novel approach to clustering called *cluster mining:* instead of attempting to partition the entire data space into disjoint clusters, we search for a small number of cohesive (and possibly overlapping) clusters. In that paper we snowed that, in our domain, cluster mining outperformed traditional clustering algorithms.

In this paper we introduce *conceptual cluster mining:* given a collection of objects, a pairwise similarity measure over those objects, and a conceptual language for describing them, we search for a small set of cohesive clusters that *correspond to concepts expressible in the language.* We formalize this problem in Figure 1.

2.2 Previous Work

Relevant previous work is of two types: *statistical* approaches to clustering and data mining, and *conceptual* clustering. *Statistical clustering* (see [Voorhees, 1986; Rasmussen, 1992; Willet, 1988]) is a technique for partitioning a set of objects in a multidimensional data space into "clusters" of objects that are close in that space. *Cluster mining* is a statistical technique for finding only cohesive clusters in a data space, instead of partitioning the entire space. *PageGather* (see [Perkowitz and Etzioni, 1998]) is a cluster mining algorithm which has been applied to the problem of synthesizing web pages. *Frequent set* algorithms are designed to find sets of similar items in large collections (see [Agrawal and Srikant, 1994; Savasere *et a/.,* 1995; Toivonen, 1996]). In a traditional frequent set problem, the data is a collection of *market basket* information. Each basket contains a set of items, and the goal is to find sets of items that appear together in many baskets. The standard frequent set algorithm is Apriori [Agrawal and Srikant, 1994]. All of these statistical approaches are useful for finding cohesive sets of objects in large collections of data, but make no attempt to ensure that their results correspond to a intuitive concept.

Conceptual clustering algorithms (see [Michalski and Stepp, 1983]), like their statistical counterparts, partition a data collection into clusters of similar objects. In a conceptual clustering problem, however, objects are described in a conceptual description language, and each cluster corresponds to a concept expressible in that language. These conceptual languages may be simple conjunctions of attributes as in [Michalski and Stepp, 1983], probabilistic descriptions as in [Fisher, 1987], or complex and cognitively inspired as in [Hanson and Bauer, 1989]. All of these are clustering rather than cluster mining algorithms; they attempt to find the best partition of the entire space rather than the best concepts. For our problem, we require an algorithm that can discover a small set of clusters that are both cohesive and "conceptual".

2.3 The SCML Algorithm Schema

The naive algorithm for the conceptual cluster mining problem might look like this:

```
The SGML Algorithm Schema $\{\Omega, \Gamma\}$
1. Run statistical clustering algorithm $\Omega$ on the data collection D, producing a set C of clusters.
2. For each cluster c in C
    (a) Tag the objects in c as positive examples.
    (b) Tag remaining objects (D - c) as negative examples.
    (c) Use these as input to concept learning algorithm r.
    (d) Find the concept v — $\Gamma(c, D - c, L)$
    (e) Find the set of objects $c_v$ which is the extension of v.
3. Return all the sets $c_v$ found.
```

Figure 2: The SGML algorithm schema, which is instantiated with a statistical cluster mining algorithm and a. concept learning algorithm.

1. Enumerate all concepts / expressible in *L*
2. For each /, compute /'s cohesiveness with respect to the similarity measure m.
3. Return all / with a cohesiveness score over a certain threshold.

Of course, the number of expressions in *L* could be quite large. For example, when *L* contains conjunctions of attributes, the number of expressions in *L* is exponential in the number of attributes, making this algorithm impractical for real applications. How can we devise a tractable algorithm for this problem?

Our previous work has presented cluster mining, a tractable technique for finding cohesive *statistical* clusters; if we could efficiently transform these clusters into conceptual ones, we would have a tractable algorithm. In essence, we would like to find the conceptual cluster that best approximates each statistical one. Our key insight was that the members of a, statistical cluster can be viewed as positive examples of the desired concept, and objects outside the cluster as negative examples, enabling us to apply concept learning techniques to efficiently generate the desired approximations.

We therefore propose the SGML algorithm schema (Figure 2), which uses statistical cluster mining to find cohesive clusters and concept learning to find a concept that describes each statistical cluster. As the match between statistical and conceptual clusters will be imperfect, we will require a noise-tolerant learning algorithm. The output of this schema is a set of clusters (and their conceptual descriptions). This approach is not guaranteed to find the optimal set of conceptual clusters, but it is tractable and it enables us to leverage continuing advances in concept learning research.

3 SCML and Index Page Synthesis

We have presented the general conceptual cluster mining problem. In this section we frame the index page synthesis problem as conceptual cluster mining. *Page synthesis* is the automatic creation of web pages. An *index page* is a page consisting of links to a set of pages that cover a particular topic (e.g., electric guitars). Given this terminology we define the index page synthesis problem: given a web site and a visitor access log, create new index pages that contain *coherent* collections of links: ones that exhaustively cover specific topics at the site. We are not provided with a pre-selected set of topics. Possible topics are instead described in a conceptual language, and the exponential number of expressible topics makes enumerating them impractical. Instead, we must search the space of possible concepts. An access log contains one entry for each page requested of the web server. Each request lists at least the origin (IP address) of the request, the URL requested, and the time of the request.

Previously, our approach to this problem has been to use statistical cluster mining, embodied in PageGather, our statistical page synthesis algorithm. Briefly, PageGather creates a graph in which each node represents a page at the web site. The algorithm then processes the access logs to find pages that are often visited together by the site's users; such pages are connected by an edge in the graph. Finally, the algorithm finds maximal cliques in the graph and outputs those sets of pages.[1]

We instantiate the SCML schema using PageGather for Ω. We use three different algorithms for *T*. As a baseline, we wished to use a simple algorithm that would perform a greedy search from general to specific, constructing a conjunctive rule, as this approach seems well-tailored to our problem. Accordingly, we based our algorithm on GREEDY-3 [Pagallo and Haussler, 1990]. The algorithm iteratively builds a conjunctive rule, choosing a conjunct that maximizes the scoring function at each iteration.

We use two scoring functions, producing two instantiations of SCML. As positive examples in our domain are typically outnumbered by negatives, the first scoring function — called PG_{POS} — is a simple count of positives (with negatives used to break ties); we use a maximum rule length to limit the complexity of rules. The second PG_{MDL} is based on minimum description length (MDL) [Quinlan and Rivest, 1989]. As the MDL measure includes a tradeoff between the complexity of a rule (its length in conjuncts) and its effectiveness, we do not need an explicit maximum length; the simple measure provides no such tradeoff.

The third algorithm is PG_{RIP}, which uses RIPPERK [Cohen, 1995] for the concept learning algorithm T RIPPERK: is a rule-learning algorithm designed to work well with large, noisy datasets and is competitive with the classic concept learning systems C4.5 and C4.5rules in both speed and performance. In section 4, we present results from PG_{POS}, PG_{MDL}, and PG_{RIP}.

4 Experiments

In this section, we evaluate the SCML algorithm schema, comparing it to PageGather, COBWEB (a popular conceptual clustering algorithm), and clusters derived from human-authored pages at a web site. Furthermore, we compare several instances of the SCML algorithm schema.

[1] The cluster size is bounded to ensure a polynomial-time algorithm.

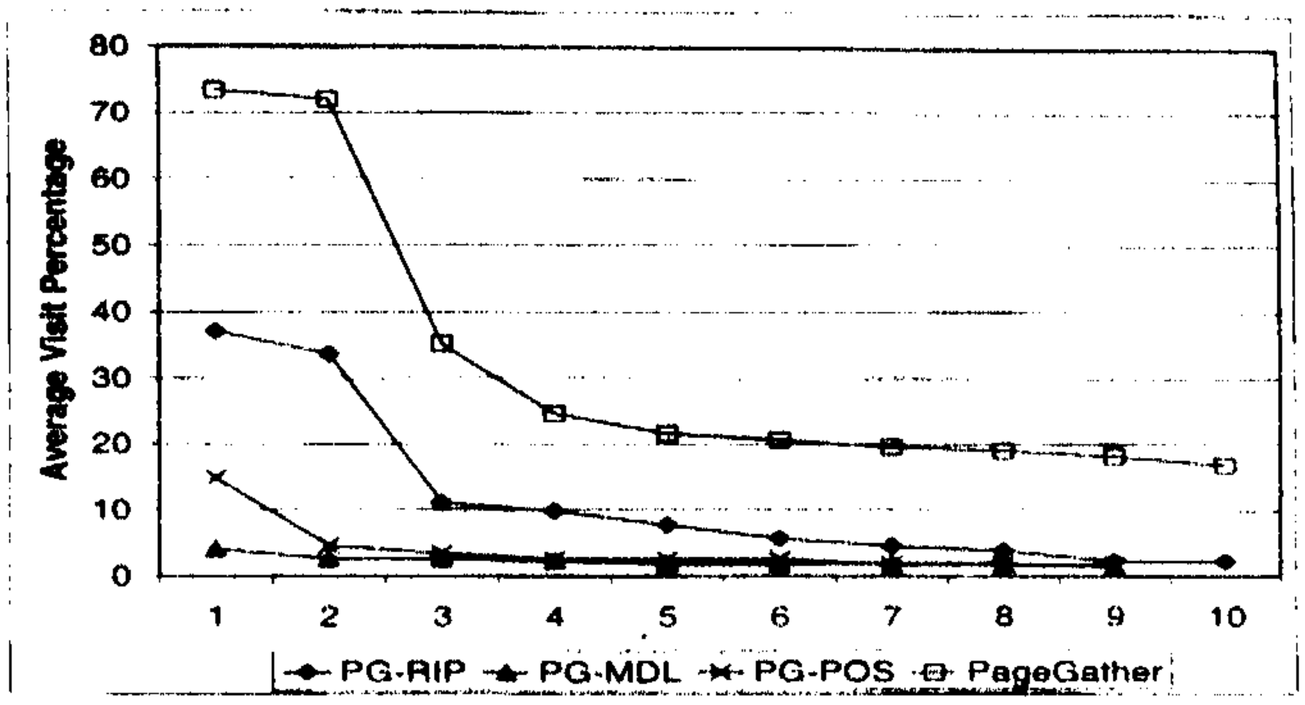

Figure 3: The performance of various instantiations of ►SGML compared with PageGather, Clusters found by PageGather have substantially higher visit percentages (but do not correspond to expressible concepts). Clusters found by PG_{RIF}• show substantially higher average visit percentages than PG_{POS} or PG_{MDL}

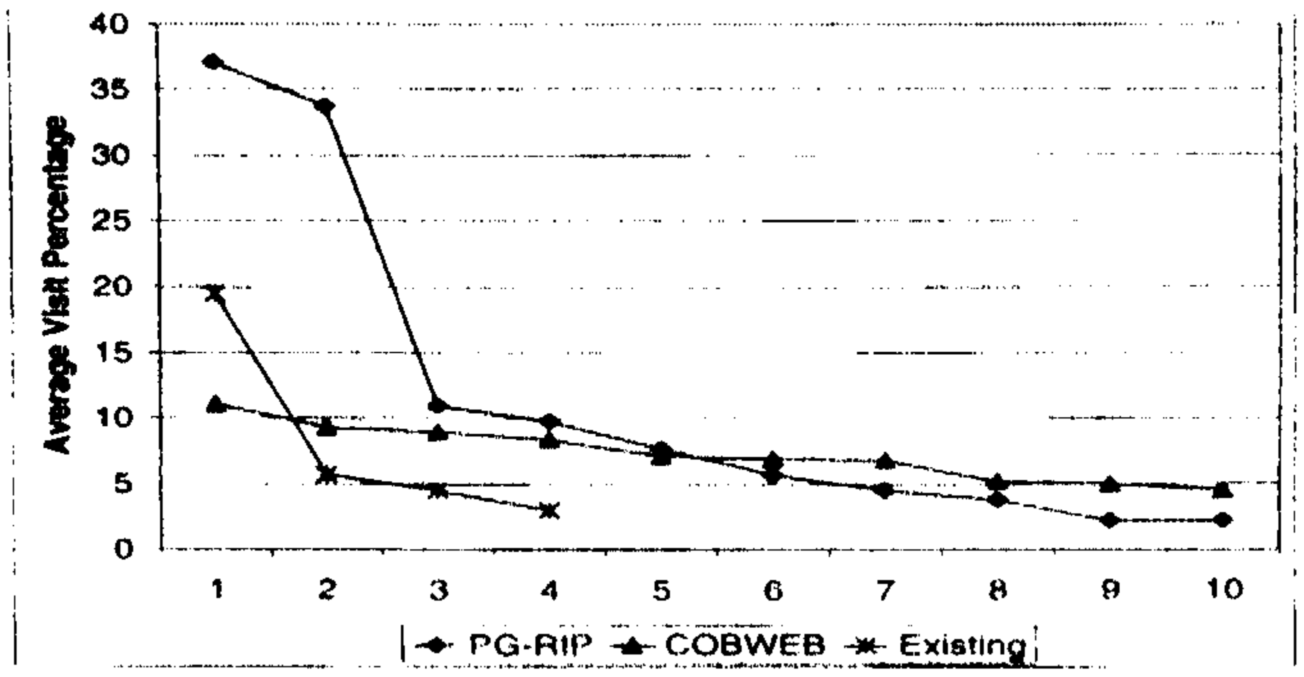

Figure 4: *PGRIP* compared with the popular conceptual clustering algorithm COBWEB and clusters corresponding to human-authored pages at the web site. The best clusters found by *PGRIP* are better to those found by COBWEB; otherwise, the two algorithms are comparable. Both score higher than the pre-existing clusters.

Our experiments draw on data collected from the Music Machines web site,[2] a site devoted to information about many kinds of electronic musical instruments. The site contains approximately 2500 distinct documents, including HTML pages, plain text, images, and audio samples. Music Machines receives approximately 10,000 hits per day from roughly 1200 distinct visitors. In our experiments, the training data is a collection of access logs for six months; the test data is a set of logs from a subsequent one-month period.[3] Our conceptual language *L* consists of conjunctions of attributes describing pages and musical instruments (e.g., type of instrument, price, and type of file; see Figure 5 for examples).

We compare the algorithms in terms of the quality of the candidate index pages they produce. Measuring cluster quality is a notoriously difficult problem. To measure the quality of a cluster as an index page candidate, we need some measure of whether the cluster captures a set of pages that are viewed by users in the same visit. If so, then grouping them together on an index page would save the user the trouble of traversing the site to find them. As an approximate measure we ask: if a user visits any one page in the cluster, what percentage of pages in the cluster is she likely to visit overall? More formally, let *V* be the set of user visits to the site and V_c be the set of $v \in V$ that include at least one page from cluster *c.* For a particular visit $v \in V_c$, the set of pages visited in *r* is $v \cap c$. The average number of hits to a cluster, over all visits, is $\sum_{v \in V_c} \frac{|v \cap c|}{|V_c|}$. Finally, the average percentage is this average divided by the size of c: $\frac{\sum_{v \in V_c} \frac{|v \cap c|}{|V_c|}}{|c|}$. In each experiment, each algorithm outputs a ranked list of clusters. In all graphs, these clusters are sorted by average visit percentage for the sake of clarity.

First, we compare instantiations of SCML. In section 3, we described three different learning algorithms to plug into the SCML schema: (1) RIPPERK *(PGRIP);* (2) a greedy algorithm with a scoring function based on positive examples *(PGpos)',* and (3) the greedy algorithm with an MDL-based scoring function *(PGMDL)-* Figure 3 compares these three variations. We see that *PGHIP* performs substantially better than either of the other algorithms. All of *PGRIP'S* clusters are better than all but one of those found by the other algorithms.

In Figure 3, we also compare the PageGather algorithm with SCML. We see in this graph that the clusters found by PageGather are substantially more cohesive than those found by SCML. The best PageGather clusters have an average visit percentage of over 70%, while SCML tops out below 40%. PageGather, in essence, finds the most cohesive clusters in the data, regardless of their conceptual coherence; SCML must respect the conceptual constraint and so cannot necessarily return the most cohesive clusters. Although PageGather produces more cohesive clusters, we feel that conceptual coherence is essential; coherence makes the pages easier to name automatically, easier to evaluate by the human webmaster, and easier to navigate by site visitors.

In Figure 4 we compare SCML with both COBWEB and human-authored clusters. COBWEB is a popular conceptual clustering algorithm, first presented in [Fisher, 1987]. COBWEB builds a hierarchical partition of the object space, using probabilistic representations of clusters. Typically, the top level of the hierarchy is the *basic level,* or most appropriate level at which to partition the space. However, since our domain dictates that the clusters be in a certain size range (it is impractical to create a web page containing hundreds of links), we choose the level of the hierarchy at which the average cluster size is in the right range typically 20-30. The best clusters found by SCML are substantially better — with average visit percentages at around 35-40% — than any of those found by COBWEB. Otherwise, both algorithms find clusters with visit percentages around 10%.

It is natural to ask how good our clusters really are, and how high of an average visit percentage is high enough. To attempt to answer this question we look to the "ideal" case: index pages created by a human webmaster. Music Machines contains many index pages

"http://machines.hyperreal.org

* Data sets are available from the authors.

	Rule
PG_{RIP}	filetype=index, age=vintage, instrument-type=sequencer
	filetype=index, instrument-type=controller
	instrument-type=sequencer, price=expensive
	filetype=samples, age=vintage, price=cheap
	filetype=samples, form=tabletop, price=cheap
COBWEB	age=vintage, filetype=mods, form=keyboard, instrument-type=synth, midi=no, pagetype=instrument, price=midpriced, synthesis=analogue, sonic=monophonic
	age=modern, instrument-type=synth, pagetype=instrument, price=midpriced, synthesis=analogue, sonic=monophonic
	filetype=info, instrument-type=synth, pagetype=instrument
	filetype=info, form=module, instrument-type=sampler, midi=yes, pagetype=instrument, price=midpriced, synthesis=digital, sonic=sampler
	age=vintage, form=keyboard, instrument-type=sampler, midi=yes, pagetype=instrument, synthesis=hybrid, sonic=sampler

Figure 5: Conceptual descriptions of the top five clusters found by *PGRJP* and COBWEB.

of different kinds. Expecting all of the site's index pages to score substantially better than PageGather's output, we chose index pages pertaining to four representative topics: (1) instruments produced by a particular manufacturer (Roland); (2) documents pertaining to a particular instrument (the Roland Juno keyboard); (3) all instruments of a particular type (drum machines); and (4) all files of a particular type (audio samples). The set of outgoing links on each page (excluding standard navigation links) was treated as a "cluster" and evaluated on our test data. Figure 4 shows the comparison of these clusters to the output of *PGRIP*. Although *PGRIP'S* clusters show higher average visit percentages than the human-authored pages, the human-authored pages tend to be substantially larger, often leading to lower percentage scores. If we instead compare the average number of hits to each cluster, the human-authored pages perform at least as well as *PGRIP'S*.

Visit percentage measures only the statistical cohesiveness of our clusters with respect to the data. Of course, as demanded by our problem definition, we also require that these clusters correspond to intuitive concepts. Excepting the original PageGather algorithm, all the algorithms we have discussed produce conjunctive concepts in *L*. Keeping in mind that we wish to find concepts that we can express to users of the web site, these concepts should be fairly simple — i.e. without too many conjuncts. Figure 5 shows the top five concepts found by both *PGRIP* and COBWEB. Although both algorithms produce clusters of a reasonable size, COBWEB's *concepts* tend to be more complex.

5 Automatically Derived Conceptual Descriptions

We have presented two general algorithms for finding clusters of related pages at a web site: PageGather and SCML. PageGather requires no input beyond the web site and access log, but cannot produce conceptually coherent clusters. SCML, on the other hand, produces conceptual clusters but carries the strong requirement that *all* documents at the site be fully described in a conceptual language. We now explore the question of whether it is possible to gain some of the advantages of conceptual clustering without paying the price of generating full meta-information.

Our approach is to derive meta-information from the structure of the web site or the content of the pages themselves, which may be done completely automatically or with some assistance from the webmaster. We describe two such approaches and present preliminary experimental results. Both approaches are based upon defining a different conceptual language in which to describe web pages.

The first approach is based on file types. Rather than describing every document at the site with a set of attributes (as was done for SCML), we tag each page with its filetype. Filetypes are derived solely from the file suffix; common suffixes include JPG, GIF, ZIP, HTML, TXT, and so on. Each statistical cluster found by PageGather is partitioned by filetype; small partitions are discarded. The remaining partitions are our final clusters. Note that we only guarantee that all pages in a cluster are of the same type; we do not add all other pages of that type to the cluster as SCML would. Therefore, this approach can find concepts that SCML could not. We have implemented this algorithm — *PGTYPE* — and tested it on our data.

The second approach is based on *content-based clustering.* PageGather clusters pages based on visitor access patterns; perhaps by clustering them based on their content, we will discover clusters that are more conceptually coherent. We apply a clustering algorithm to the pages at the site to obtain a set of content-based clusters. Next, for each cluster output by PageGather, we find the closest content-based cluster — the one with the highest overlap. These closest clusters are our final output. Note that these concepts are defined with respect to keywords rather than more intuitive concepts. Keywords may indicate conceptual similarity, but are not perfect due to synonyms and homonyms. We have implemented this algorithm using suffix-tree clustering (STC) [Zamir and Etzioni, 1998] and tested *PGSTC* on our data.

In Figure 6, we compare *PGRIP* with *PGTYPE* and *PGSTC*. Clusters found by *PGTYPE* have the highest average visit percentages; this is not surprising, as *PGTYPE'S* output is closest to PageGather clusters.

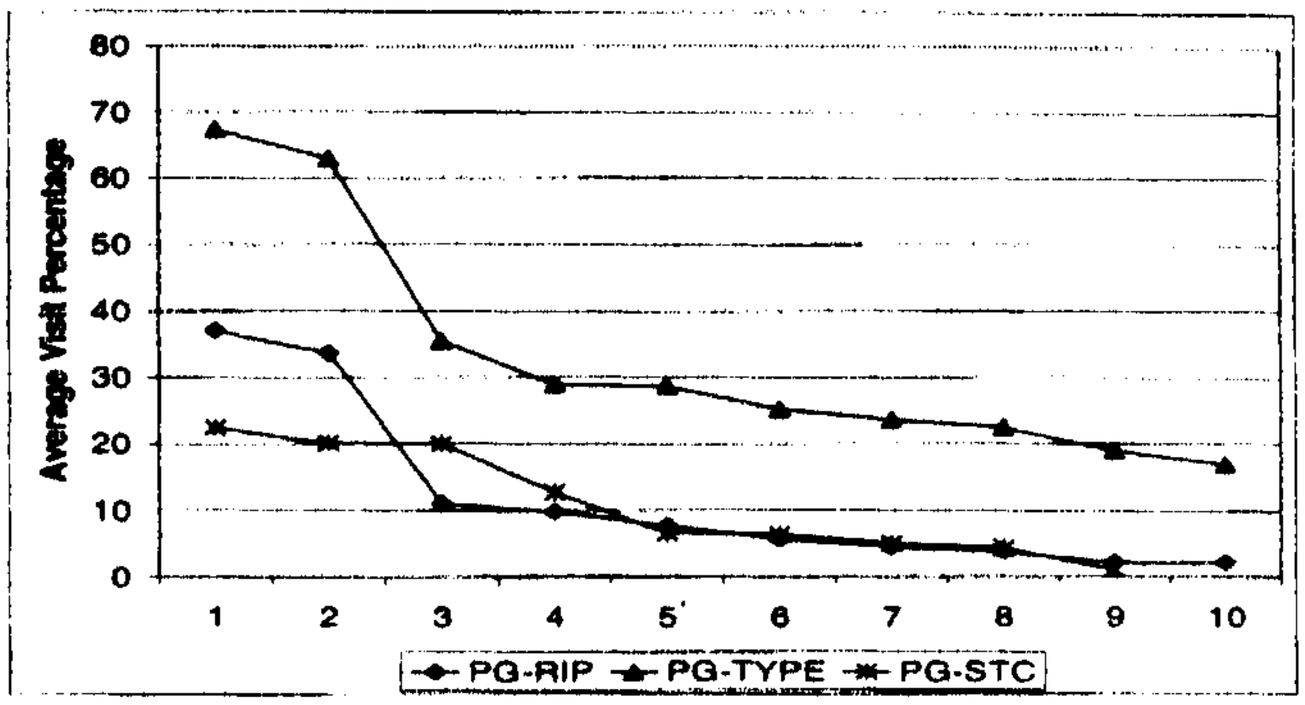

Figure 6: PG_{RIP} compared with PG_{TYPE} and PG_{STC}. Clusters found by PG_{TYPE} have the highest average visit percentages.

6 Conclusion and Future Work

This paper has responded to the broad Adaptive Web Sites challenge task by presenting and evaluating a method for index page synthesis. More precisely, the method solves the problem of discovering coherent and cohesive link sets, which can be presented to a human webmaster as candidate index pages. While our method is based on previous work, this paper introduces and evaluates a key extension which ensures that the candidate index pages generated correspond to an intuitive concept, which makes the pages easier to name automatically, easier to evaluate by the human webmaster, and easier to navigate by site visitors. We have also explored two ways of relaxing the requirement of full meta-information by deriving information from the site's structure and content.

The challenge has also led us to formulate the domain-independent problem of Conceptual Cluster Mining (Figure 1), and an algorithm schema for its solution. We have instantiated SGML with various concept learning algorithms and conceptual languages, and compared these instantiations experimentally. We have shown that *PGRIP* outperforms COBWEB (a leading conceptual clustering algorithm) in our domain.

In future work, we plan to investigate alternative measures of index page quality and also to carry out user studies (with both Web site visitors and webmasters) to assess the impact of the suggested adaptations on users in practice. In addition, we plan to test our approach on additional web sites, including our department's web site, in the near future. Finally, index page synthesis itself is a step towards the long-term goal of *change in view:* adaptive sites that automatically suggest re- organizations of their contents based on visitor access patterns.

References

[Agrawal and Srikant, 1994] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th VLDB Conference,* 1994.

[Anantharaman *et al.,* 1990] T. Anantharaman, M. Campbell, and F. Hsu. Singular extensions: adding selectivity to brute-force searching. *Artificial Intelligence,* 43(1):99-109, 1990.

[Cohen, 1995] W. Cohen. Fast effective rule induction. In *Proc. 12th Int. Conf. Machine Learning,* 1995.

(Fisher, 1987] D. Fisher. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning,* 2:139-172, 1987.

[Hanson and Bauer, 1989] S. Hanson and M. Bauer. Conceptual Clustering, Categorization, and Polymorphy. *Machine Learning,* 3:343-372, 1989.

[Michalski and Stepp, 1983] R. Michalski and R. Stepp. *Learning from Observation: Conceptual Clustering,* pages 331-363. Morgan Kaufman Publishers, 1983.

[Pagallo and Haussler, 1990] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning,* 5:71-100, 1990.

[Perkowitz and Etzioni, 1997] M. Perkowitz and O. Etzioni. Adaptive web sites: an AI challenge. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence,* 1997.

[Perkowitz and Etzioni, 1998] M. Perkowitz and O. Etzioni. Adaptive Web Sites: Automatically Synthesizing Web Pages. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence,* 1998.

[Quinlan and Rivest, 1989] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation,* 80:227-248, 1989.

[Rasmussen, 1992] E. Rasmussen. Clustering algorithms. In W.B. Frakes and R. Baeza-Yates, editors, *Information Retrieval,* pages 419-442. Prentice Hall, Eaglewood Cliffs, N.J., 1992.

[Savasere *et al.,* 1995] A. Savasere, E. Omiecinski, and S. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. In *Proceedings of the 21st VLDB Conference,* 1995.

[Thorpe, 1990] C. Thorpe, editor. *Vision and Navigation: the Carnegie Mellon Navlab.* Kluwer Academic Publishing, Boston, MA, 1990.

[Toivonen, 1996] H. Toivonen. Sampling Large Databases for Association Rules. In *Proceedings of the 22nd VLDB Conference,* pages 134 145, 1996.

[Voorhees, 1986] E.M. Voorhees. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. *Information Processing & Management,* 22:465-476, 1986.

[Willet, 1988] P. Willet. Recent trends in hierarchical document clustering: a critical review. *Information Processing and Management,* 24:577-97, 1988.

[Zamir and Etzioni, 1998] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrival,* August 1998.