Mining themes from bookmarks

Soumen Chakrabarti
IIT Bombay
soumen@cse.iitb.ernet.in

Yusuf Batterywala
IIT Bombay
yusuf@cse.iitb.ernet.in

ABSTRACT

Many Web surfers maintain bookmark files containing URLs organized into folders which represent topics by function or content. We propose a new framework for discovering themes among such folders created by a community of surfers with overlapping interests. In principle, surfers need not create personal folders; they can just use a 'standard' taxonomy such as Yahoo. In practice, a standard taxonomy is neither necessary nor sufficient; it is too detailed in most part, but likely to be inadequate in the community's areas of deep interest. Our framework, based on the minimum description length principle, exploits two semantically different relations: "term occurs in document" and "document occurs in folder," and combines them without the need for heuristic weighting of attributes. Themes, in effect, finds common factors in people's interests when it can, while maintaining their individuality when it must. We present experimental results to illustrate our framework and show how it is different from content-based clustering. We also propose a number of applications to finding themes, including collaborative recommendation and resource discovery.

1. INTRODUCTION

Three sources of information are currently used widely for hypertext analysis for the Web: text (terms and phrases, used by early search engines such as Alta Vista), markup tags, and hyperlinks (used by later search engines, such as Google).

Popular search sites answer tens of millions of queries per day. We speculate that the total number of browser clicks per day is at least three orders of magnitude larger. This third source of information has a scale that dwarfs the Web itself, and yet is perhaps as rich and valuable as text and links. Some centralized services such as (**Alexa**[1], **Third Voice**[2]) distribute a browser plug-in to monitor clicks and add page reviews. Others (**Hotbot**[3], **WiseWire**[4]) monitor clicks limited within their site to improve relevance ranking or dynamically reorganize content.

[1]`http://www.alexa.com`
[2]`http://www.thirdvoice.com`
[3]`http://www.hotbot.com`
[4]`http://www.wisewire.com`

ACM SIGKDD KDD 2000 Workshop on Text Mining Boston

Nevertheless, most of the information latent in years of clicking by millions of surfers is carelessly discarded by browsers, unless they are deliberately bookmarked. Even bookmarks are stored passively, in browser-dependent formats; this separates them from the dominant world of HTML hypermedia, even if their owners were willing to share them.

In 1945, Vannevar Bush dreamt of *Memex*: an enhanced, intimate supplement to personal and community memory [5]. Memex would unobtrusively archive, analyze, and index our complete experience, covering books, movies, music, conversation etc. Since then this theme of a 'living' hypermedia into which we "weave ourselves" has been emphasized often, e.g., by **Douglas Engelbart**[5] and **Ted Nelson**[6], and of late by **Tim Berners-Lee**[7] and **Jim Gray**[8].

We are building the beginnings of a Memex for the Web: a tool that archives and analyzes community Web surfing experience [8]. Memex stores two kinds of surfing information: browser clicks (history) which is unsupervised data and bookmarks/favorites placed by the surfer into designated folders, which is supervised data. The nature of this data is richer that HTTP access logs at a fixed site, or even proxy logs at an establishment; the preservation of user and session boundaries, and especially human input in the form of bookmarked topic folders, are invaluable.

How can this glut of data be used? Traditionally, such data have been massaged until they start looking like market basket data, at which point discoveries of the form "people who visit document d_1 also visit d_2" may be attempted. Because of the extreme sparsity of the data, such techniques cannot infer that two people are interested in sky-diving unless they visit some common URLs. Another standard application is to use machine learning to organize the topic folders maintained by the user.

A commonly cited goal, one whose precise definition is very hard to find, is to "build a user profile." What is a "user profile?" As argued earlier, modeling a profile as a set or sequence of URLs is crude. A distribution of keywords is better, but still limited, because for most real surfer, this distribution is a superposition of many interests, and we would do well to model people as such superpositions of topic-based information needs.

One simple approach is to map (probably using an automatic classifier) each URL visited by a user to a topic node in Yahoo's directory system. The Yahoo directory, together with visit counts on each node, would form a better representation of a 'profile.' However, mapping everyone to this one-size-fits-all directory may be problematic: Yahoo

[5]`http://jefferson.village.virginia.edu/elab/hfl0035.html`
[6]`http://www.sfc.keio.ac.jp/~ted/`
[7]`http://www.w3.org/1999/04/13-tbl.html`
[8]`http://research.microsoft.com/~gray/papers/MS_TR_99_50_TuringTalk.pdf`

is neither necessary nor sufficient for a specific individual or community. For example, the Computer Science Department here cares nothing about skiing, but wishes the Yahoo directory on compiler optimizations were deeper and better differentiated.

Our contributions:. We claim that the best characterization of an individual's profile as part of a community is in the form of a taxonomy synthesized by a generalized 'merge' of their bookmarks (and history if available); i.e., we want a taxonomy to be tailor-made for each designated community.

To this end, we propose a new notion of *themes*. Themes are like clusters, except they are synthesized out of not only the text on Web pages but also how members of the community bookmark them. More precisely, we want to mine the two relations

```
occurs_in(token, document)
belongs_to(document, folder).
```

to extract themes. We give a precise coding interpretation for finding themes, and propose and test various heuristics. Our experiments show that our heuristics go beyond simple content-based clustering, in a quantifiable sense. Our formulation may be of independent interest in clustering data with a mix of quantitative and categorical attributes.

Our notion of themes has various applications, including collaborative filtering and resource discovery. Content-based clusters are difficult to name, because of the structure of our formulation themes have a rather natural notion of names.

2. RELATED WORK

We review related work in the areas of unsupervised and semi-supervised learning, subspace selection for clustering, and clustering data with categorical attributes. It should be noted that in spite of some similarities to clustering, our problem has some distinctive features as a semi-supervised scenario.

2.1 Clustering hypertext

Documents have been clustered by representing them in the vector space [19] and then using some of the techniques for clustering structured quantitative data, e.g., bottom-up agglomerative method [11, 10], k-means [15], etc. With hypertext, there are more sources of information about similarity. Beyond content similarity, there is now structural similarity expressed via markup and formatting. Moreover pages are connected by hyperlinks, which must be factored into the notion of similarity.

For both retrieval and clustering, researchers have suggested that document models somehow capture non-local terms, i.e., terms that occur in a document that is different from the document being modeled, where the two documents may share non-textual properties, e.g., be from the same site, one hyperlink to the other, have common citations from other pages, cite common pages, etc.

Graph properties have been used for clustering for a while [3, 2]. Less clear is the manner in which non-local terms should influence the model. HyPursuit [20] was an early system to recognize that clustering hypertext involved combining textual features with features derived from markup and links. However, HyPursuit uses ad-hoc weights and combining functions to attempt to bring semantically diverse attributes to a common comparable unit.

It is our explicit goal to avoid any parametric assumptions, and instead start with a simple generative model for documents and folders, for which parameters can be readily estimated from data alone.

2.2 Subspace feature selection

Clustering methods which are density based may be able to avoid embedding discrete attributes in geometric space. For example, there are algorithms that use a density threshold to select a subset of attributes such that a projection of the original data points satisfies the density requirements [1]. Unfortunately, these attribute sets are built bottom up, and this method cannot easily cope with the extremely large number of dimensions common in hypertext.

2.3 External probes

The general difficulty of clustering data with categorical attributes has led to many techniques for assigning numeric weights (or numerical vectors) to categorical attribute values, so that the clustering problem reduces to one in a geometric space. Whereas measures of inter-attribute correlation or association give direct evidence of attribute similarity, a different, indirect notion of attribute similarity has been suggested [12]. In market basket data, it may be rare to find Coke and Pepsi in the same basket, because they are complementary and competitive products. Therefore the similarity between Coke and Pepsi will not be discovered by merely looking for associations; it has to be inferred based on the similarity of items bought together with Coke or Pepsi. Our work is influenced by some of the basic machinery of external probes, but we address a different problem.

2.4 Dynamical systems

Another recent technique for embedding categorical attributes in a geometric space uses non-linear dynamical systems [13]. In our setup comprising terms, documents, and folders, the input table has one row per document d; each document belongs to exactly one of a set of folders F. Let $\mathcal{T}$ be the universe of terms, or the *vocabulary*, over a set of documents that should be clear from context. There is a column for each term in $\mathcal{T}$, plus one column to indicate which folder the document belongs to. For simplicity let us regard a document as a *set* of terms; i.e., let us ignore term counts and position information. Thus each term column can have two possible values. The folder column can have $|F|$ possible values.

Each value is regarded as a node in a hypergraph; each document is a hyperedge incident on the appropriate nodes. A weight w_v is associated to each node v, initialized with some random value. Each document also has an associated weight x_d. Thereafter, one repeats the following steps:

> For each v:
>   Find $D(v)$, the set of documents incident on v
>   For each $d = (v, v_1, \ldots, v_k) \in D(v)$
> $x_d = \varphi(w_{v_1}, \ldots, w_{v_k})$
>   Create a new weight for v: $w'_v = \sum_{D(v)} x_d$
> Normalize w' so that $\sum_v (w'_v)^2 = 1$
> Overwrite w with w'.

Here φ could be a variety of operators, such as product, sum, max, etc. A generalization is to maintain not one but several w vectors, apply the above transformation to each, and orthogonalize them using the Gram-Schmidt procedure [14].

Although very little is known about the convergence of non-linear systems, convergence is often quick in practice. Once the system converges, the row corresponding to v can be used as a quantitative vector surrogate for v.

This application of dynamical systems was inspired by the use of linear eigen solvers to find 'popular' pages on the Web [16, 4]; there are close relations between the converged weight of a node and the number of paths of various length leading to it. Whereas this paradigm of "weight-pushing" makes some semantic sense for measures of popularity, its effect on embedding diverse attributes in the same space is not well understood; we will revisit this issue through experiments in §5.

3. PROBLEM FORMULATION

Whereas nothing short of full natural language understanding may suffice to understand why a user placed a URL in a certain folder, we will restrict our scope to statistical modeling alone. Our framework is based on statistical model selection and optimization. We do not know how to provably maximize the goodness of our model, but we know how to measure the goodness. This is in contrast to dynamical systems where the computation is simple but evaluating or assigning meaning to the answers is difficult and subjective.

There are two aspects to documents in our problem: terms occur in documents, and documents are placed in folders. Accordingly, our formulation has two main aspects.

3.1 Document models

The bag-of-words model has been widely used for search, clustering, and classification. In the *binary* model, the parameters are ϕ_t, $t \in \mathcal{T}$, which indicate the probability that a document will mention term t at least once. With this definition,

$$\Pr(d) \quad = \quad \prod_{t \in d} \phi_t \prod_{t \in \mathcal{T}, t \notin d} (1 - \phi_t). \tag{1}$$

Since typically $|\mathcal{T}| \gg |d|$, short documents are discouraged by this model. Also, in assuming term independence, the second product term results in very small probabilities for documents, much lower than is intuitively admissible. (Specifically, the sampled set, from which the model parameters are derived, usually has very low probability!)

In the *multinomial* model, each class has an associated die with $\mathcal{T}$ faces. The ϕ parameters are replaced with θ_t, the probability of the face $t \in \mathcal{T}$ turning up on tossing the die. For this model,

$$\Pr(d \,|\, \|d\|_1) \quad = \quad \binom{||d||_1}{\{n(d,t)\}} \prod_{t \in d} \theta_t^{n(d,t)}, \tag{2}$$

where $\binom{||d||_1}{\{n(d,t)\}} = \frac{||d||_1!}{n(d,t_1)!\, n(d,t_2)! \cdots}$ is the multinomial coefficient. Like the binary model, the multinomial model ignores inter-term correlation. Moreover, for each occurrence of t, the document probability takes a multiplicative 'hit' of θ_t, which again goes against intuition; having seen a term 50 times in a document, we are not very surprised to see it the 51st time.

Later we will deal with documents which are in different sets, also called *folders* (denoted f) here. The ϕ and θ parameters are suitably subscripted as $\phi_{f,t}$ and $\theta_{f,t}$ in such cases.

3.2 Folders and the tripartite graph model

The input is a set of bookmark files. Each bookmark file contains a set of folders. We assume for simplicity that folders are not nested. Each folder f contains a set of documents; a document d may be contained in multiple folders (even belonging to the same surfer). The placement of URLs (documents) into folders can be captured in a straightforward manner using a many-to-many bipartite graph (F, D) where edge (f, d) signifies that document d is included in folder f. This is called the *original* graph.

For reasons that will become clear, we extend this bipartite graph into a *canonical* tripartite form (F, G, D) where the layers are called folders, groups, and documents. A (F, D) graph is converted to an initial (F, G, D) equivalent by choosing $|G| = |D|$, and a one-to-one correspondence between G and D. If in (F, D) f_i mapped to d_i, in the new graph we create two edges (f_i, g_i) and (g_i, d_i). (F, G, D) is the 'raw' or initial form of the graph, and our goal is to 'simplify' this graph.

The simplification will leave F and D unchanged, but replace the (trivial) set of groups G with a new set G', and the mappings between F and G and G and D will change as well. The intent is, whereas the initial groups G are singleton documents, the final groups G' will succinctly capture shared and private interests among surfers. (See figure 1.)

3.3 Notions of cost

Any algorithm that simplifies G to G' must be guided by a precise notion of cost associated with such a tripartite graph. One part of the cost is the representation of edges in the graph; roughly speaking, the smaller number of edges, the better. Of course, it makes no sense to apply such a criterion by itself, or all documents will get collapsed to one group.

Balancing the need to simplify the graph is the need to preserve the composition of user folders. In the original graph, each folder f had an associated document set, which induced a term distribution T_f. In the simplified graph, we say f contains d if there is a g such that (f, g) and (g, d) are edges. Considering this new document set associated with folder f, one can characterize the simplified term distribution T'_f. Attempts to simplify the graph may in general lead to a *distortion* of the original distribution T_f into the simplified distribution T'_f.

Provided we can bring graph mapping cost and distortion cost to a comparable unit, we invoke the Minimum Description Length principle to claim that minimizing the overall cost will lead to the "best learning" of themes.

3.4 Estimating distortion cost

As we seek to simplify the tripartite graph, we might need to change the set of documents assigned to a folder. Indeed, if a broader section of the community is interested in that same topic, a central purpose of theme-finding is to alert the user to additional resources. As we change the document set, the user will find the folder more and more 'surprising.' Distortion to us is synonymous with this surprise factor.

For measuring distortion one may regard the original term distribution, as induced by user-defined folders, as a reference distribution. With this reference, it is natural to use the Kullbach-Leibler (KL) distance [9]. Let $t \in \mathcal{T}$, and for a given document, I_t be a random binary variable denoting whether the term occurred in the document or not. Thus I_t

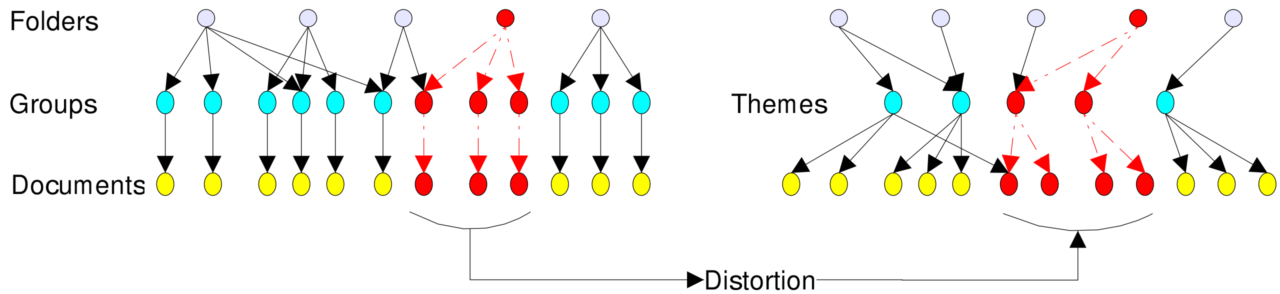

Figure 1: Illustration of the tripartite graph model, showing the formation of themes which simplify the original graph at the cost of distortions of folder term distributions.

can assume values $i_t \in \{0,1\}$. A document is thus a binary vector $\vec{i} = (i_1, i_2, \ldots, i_{|\mathcal{T}|})$.

The KL distance of a distribution T' with reference to another distribution T is given by

$$\mathrm{KL}(T'||T) = \sum_{\vec{i}\in\{0,1\}^{|\mathcal{T}|}} \Pr{}_{T'}(\vec{i}) \log \frac{\Pr_{T'}(\vec{i})}{\Pr_{T}(\vec{i})}. \qquad (3)$$

The KL distance is not symmetric; however it is non-negative and zero only for $T = T'$.

The sum over all $\vec{i}$ is impractical to compute; even with the binary model, there will be $2^{|\mathcal{T}|}$ terms in the sum. We consider two possible approaches to stem this complexity.

In practice, we are not given distributions for T and T' directly but must estimate them from observed data. One approach is to use a *parameterized* approximation to the reference distribution T, estimate the parameters from the data corresponding to T, and then find the probability of generating the data corresponding to T'.

A computationally simpler option is to assume attribute independence in equation (3), approximating it by:

$$\begin{aligned}\mathrm{KL}(T'||T) &= \sum_{t\in\mathcal{T}} \sum_{i_t\in\{0,1\}} \Pr{}_{T'}(i_t) \log \frac{\Pr_{T'}(i_t)}{\Pr_{T}(i_t)} \qquad (4)\\ &= \sum_{t\in\mathcal{T}} \big\{ \phi'_t(\log\phi'_t - \log\phi_t) + \\ &\qquad (1-\phi'_t)\big(\log(1-\phi'_t) - \log(1-\phi_t)\big)\big\}, \qquad (5)\end{aligned}$$

where ϕ_t and ϕ'_t are our estimates of $\Pr_T(t)$ and $\Pr_{T'}(t)$ respectively. It can be verified that $\mathrm{KL}(T'||T) = 0$ for $T = T'$.

The independence assumption will typically lead us to overestimate the distribution distance. One way to get around this is to perform some form of feature selection and limit the number of 'independent' terms that can contribute to (5).

At this stage we can design the subroutine $\mathrm{KL}(D_1||D_2)$ where D_1, D_2 are two sets (folders) of documents. We set the vocabulary $\mathcal{T}$ to the set of terms found in documents belonging to $D_1 \cup D_2$, and estimate the parameters $\phi_{f,t}$, where $f = 1, 2$, as

$$\phi_{f,t} = \frac{1 + |\{d \in D_f : d \ni t\}|}{2 + |D_f|}. \qquad (6)$$

Here we have applied the Laplace correction [17, 18] to make sure that none of ϕ, ϕ' becomes zero, and (5) can be suitably evaluated. Other smoothing techniques, such as Lidstone's law, can also be used.

Instead of the asymmetric KL distance, one may use a symmetric distance such as the Jensen-Shannon divergence:

$$\mathrm{JS}(T', T) = \mathrm{KL}\left(T'||\tfrac{T'+T}{2}\right) + \mathrm{KL}\left(T||\tfrac{T'+T}{2}\right), \qquad (7)$$

which can be computed with only the non-zero parts of both distributions. This eliminates the need for probability smoothing.

3.5 Estimating mapping costs

As mentioned before, we wish to add and compare, in some standard units, the distortion of term distributions and the complexity of mapping folders to groups to documents. Accordingly, in this section we will estimate the cost of encoding the tripartite graph.

There are two bipartite mappings in the tripartite graph: one between documents and themes (groups) and the other between groups and folders. We need to design an efficient scheme for coding a bipartite graph. This can be an easy or a hard problem depending on our assumptions. Consider the (G, D) graph, where $|G| \geq |D|$. It has $|G|\,|D|$ possible edges. Suppose e edges actually exist. If all such e-edge graphs are equally likely, we must spend $\log\binom{|G|\,|D|}{e}$ bits. Since this is roughly $e(\log|G| + \log|D|)$, no complicated code is needed.

Actually we do not permit all possible e-edge graphs, since each $d \in D$ must be assigned to some group. Some groups have more documents assigned to them than others, and we should like to exploit this. Pretend for a moment that each document is assigned to *exactly* one group. Let the degree of node $g \in G$ be δ_g; $\sum_g \delta_g = |D|$. Then the empirical entropy of the symbol set $\{g\}$ is $-\sum_g (\delta_g/|D|)\log(\delta_g/|D|)$. This estimate can be used to calculate the average number of bits needed to encode the group assignment of each document. An ordering of D is decided ahead of time, and a sequence of group IDs are encoded. Now some documents may refer to two or more groups, this is done by using one "continuation bit" which affects the total number of bits negligibly.

This, then, is our coding scheme for the tripartite graph; an identical code is used for the (F, G) layer. (In the most general case, encoding a bipartite graph is rather complicated, with connections to finding low-rank approximations to the adjacency matrix corresponding to the graph. Such coding schemes are beyond our scope.)

4. ALGORITHMS

Our formulation is as hard as the set-cover problem. Hence we look for practical heuristics. We propose and study three algorithms to simplify the tripartite graph model: **SingleBest**, **PatchHAC**, and **Bicriteria**. In addition we implemented a dynamical system to try to embed categorical values in a (2d) geometric space.

4.1 SingleBest

To a first approximation, folders are created to designate content-based *topics*. Hence one approach to simplify the tripartite graph would be to:

1. Perform a content-based hierarchical agglomerative clustering (HAC) on D (details next). This results in a binary tree with $|D|-1$ internal vertices, where the $|D|$ original documents are leaves.

2. For each original folder $f \in F$, choose that vertex g in the HAC tree which yields the smallest $\mathrm{KL}(D_g, D_f)$. Here D_f is the original set of documents associated with folder f, and D_g is the set of leaf documents of the HAC subtree rooted at g.

3. Replace the original G with the chosen HAC tree vertices. Complete the (F, G) and the (G, D) mappings in the obvious manner.

SingleBest tries to minimize the (F, G) mapping cost at the expense of increased distortion. The HAC process is standard [11, 10]. Represent documents in the vector space model [19] and let the similarity $s(d_1, d_2)$ between documents d_1 and d_2 be the cosine of the angle between their corresponding vectors. Let the *self-similarity* $s(\Gamma)$ of a set of documents be the average pair-wise similarity:

$$s(\Gamma) = \frac{1}{|\Gamma|(|\Gamma|-1)} \sum_{d_1, d_2 \in \Gamma, d_1 \neq d_2} s(d_1, d_2), \qquad (8)$$

The algorithm initially places each document into a group by itself. While there is more than one group the algorithm looks for a Γ and a Δ such as $s(\Gamma \cup \Delta)$ is maximized, and merges these two groups. This algorithm takes $O(|D|^2 \log |D|)$ time with $O(|D|^2)$ space.

4.2 PatchHAC

In our experience, HAC tends to assign a folder to a set of groups that is insufficient in driving down the folder's distortion. PatchHAC seeks to locally and greedily fix the oversights of HAC in the following steps.

1. Apply HAC and SingleBest.

2. Find the most distorted original folder f and the set of HAC groups G_f that it maps to.

3. Find, if any, a document d originally placed in folder f which now cannot be found mapped from any $g \in G_f$.

4. Try adding d to some $g \in G_f$ to reduce total codelength as much as possible. (A reasonable heuristic is to try to choose a g which is mapped to from the fewest f's.)

5. Similarly, look for extraneous documents that got into G_f and again eliminate them if that results in a reduction of total cost.

6. Repeat the above while codelength improvement is possible.

4.3 Bicriteria

A more circumspect approach is to start with singleton groups of one document each, but guide the pair merging not by content-based similarity alone.

1. Initially, pick as many groups in G as there are documents in D, assign one-to-one, and map folders in F to corresponding groups in G.

2. While an improvement in total cost is possible, repeat the following steps.

3. Choose two groups g_1 and g_2 to tentatively merge. (Content-based similarity could be one guiding heuristic. Number of common folders mapping into them could be another.)

4. Pretend to merge g_1 and g_2 into g_0. For all folders f that mapped into g_1 or g_2, remap them to g_0.

5. Evaluate the new overall codelength and commit the change if favorable.

Unlike HAC where various clever data structures and sharing of similarity computations can be used to cut down the complexity, the additional complication of mapping and distortion w.r.t. an initial distribution prevents such optimizations.

5. EXPERIMENTAL RESULTS

5.1 Data set and test-bed

For our experiments we obtained bookmark files from 33 Web surfers. This took some effort, as many people are not comfortable sharing their entire bookmark file. Eventually we succeeded by letting volunteers send us only selected folders. Our volunteers are Web-savvy. About 30% are computer professionals.

After converting their Netscape and Explorer bookmark to a common format and throwing away default folders inserted by those browsers, we were left with 502 folders. All but two folders were at the first level; for simplicity we flattened those. These folders together pointed to 1756 distinct URLs. About 1400 of these links were alive.

We randomly subset the above data for two reasons: first, some folders were degenerate and contained only one URL (which made them useless for our study), and second, our algorithms have worse-than-quadratic complexity. We performed experiments on various randomly picked subsets to make sure our observations are robust and are not artifacts of specific samples. Here we will present results for one particular sample of 272 URLs belonging to 93 folders, each folder having two or more URLs.

Our prototype code has been implemented in Java on top of a relational database (IBM Universal Database 5.2) to store document, folder, and group metadata, and a lightweight storage manager called Berkeley DB to store term distributions. The space and time overhead of storing fine-grained term frequency information in the RDBMS would be prohibitive. Our code runs on a 450 MHz Pentium2 running RedHat Linux 5.2.

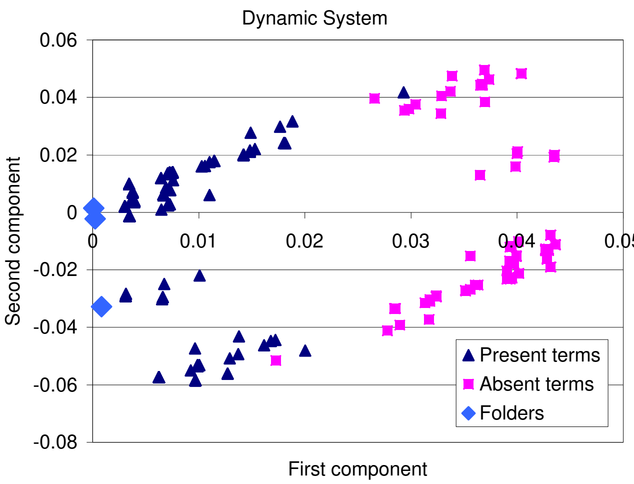

Figure 2: Embedding our categorical attributes (terms and folders) using dynamical systems fails to collect folders with salient terms; instead, the folders from a cluster with small coordinates, which forecasts that a clustering with these geometric surrogates will not behave very differently from content-based HAC.

5.2 Embedding approaches

Algorithms for embedding categorical data in geometric spaces suggest the following alternative to our approach: first use such an algorithm to convert the data to points in a space of suitably dimensionality, then apply a standard distance-based clustering algorithm to derive the themes or groups. In this section we give circumstantial evidence that such an approach may not succeed (at least for documents and folders).

We used the dynamical systems algorithm to create a two-dimensional embedding of a further sampled subset of collected documents from three folders. The chosen folders were named (by their creators) *Books* (970 terms, 531 distinct), *Reference* (529 terms, 406 distinct), and *Basic Portals* (309 terms, 254 distinct). We expected the first two to be relatively similar to each other as compared to the third: there were 83 shared terms (total count 162) between Reference and Basic Portals, 166 shared terms (total count 276) between Books and Reference, and 93 shared terms (total count 132) between Books and Basic Portals.

In our setup, the categorical values were

- Terms present in documents,
- Terms absent in documents, and
- The three class labels.

Convergence is quick. The results are shown in Figure 2.

Visually, we do see three clusters of present terms and three clusters of absent terms. However, it is noteworthy that absent terms have the largest magnitude of coordinates, followed by present terms, and folders have the smallest magnitude of coordinates. This is because terms are rare in general and thus negative terms are incident to a greater number of documents (hyperedges), while a folder is mentioned only once in a document. Readers familiar with hyperlink-based Web page ranking algorithms such as HITS [16], Clever [6] or Google (`http://google.com`) can reinterpret the previous sentence to the effect that negative terms are the most 'popular' and folders are not popular at all.

Furthermore, we were expecting folders to be cluster exemplars (from conventional k-means wisdom). However, it turns out that folders are more similar to each other than are terms! One can rationalize this with the hindsight that similarity is based on the function an attribute performs in each document. It is also to be noted that the distance between the three folders is contrary to the kind of documents that we would assume they contain.

These observations imply that the folder nodes, being close together and having small coordinates, will only have a feeble effect on similarity computations for clustering. In effect, the large number of negative term nodes will swamp out the supervised information present in the folder labels. This is quite reminiscent of our earlier experience with supervised and semi-supervised topic learners for hypertext [7].

5.3 SingleBest

Going back to our data set with 93 folders and 272 URLs, we first calculated the *bipartite* direct (F, D) mapping cost. Let δ_f be the degree of $f \in F$, where $\sum_f \delta_f = |D|$. To encode symbol f will therefore take roughly $-\log(\delta_f/|D|)$ bits. Let F_d be the set of folders that d belongs to. Then the total mapping cost for (F, D) is roughly

$$\sum_{d \in D} \sum_{f \in F_d} -\log \frac{\delta_f}{|D|}. \tag{9}$$

For our data set this evaluates to **1265 bits**, and the distortion is zero, by definition.

Comparing this with the results of our algorithms will not be entirely fair, as those are tripartite graphs. There are two extreme ways to convert the bipartite graph into a canonical tripartite graph:

- Make G isomorphic to D and (G, D) a one-to-one map, then map F to G according to the isomorphism. This is the (expensive) starting point for our algorithms, costing 1265 bits for the (F, G) map, 1525 bits for the (G, D) map, totaling **2790 bits**[9].
- Make G isomorphic to F and (F, G) a one-to-one map, then map G to D according to the isomorphism. Since $|G| < |D|$, this is much cheaper than the first option, 422 bits for the (F, G) map, 1265 bits for the (G, D) map, totaling **1687 bits**.

The first representation is just a baseline estimate and clearly wasteful. The important question is, can any of our algorithms beat the second number. Otherwise, the data has only trivial 'compressibility' (hence, learnability).

SingleBest beats the 1687-bit mark by a noticeable margin. It picks $|G| = 36$ groups from among the HAC tree vertices. The (F, G) map costs 362 bits. The 36 groups contain 81 documents (out of the 272 for the original folders). The (D, G) map costs 361 bits. As might be expected, the distortion soars from zero to 839 bits, for a total of **1563 bits**. This is higher than the bipartite baseline, but

[9]It may seem wasteful to encode the isomorphism (G, D) using so many bits, but the algorithm might and does change it substantially.

that is understandable. (In all our experiments we have used KL distance.)

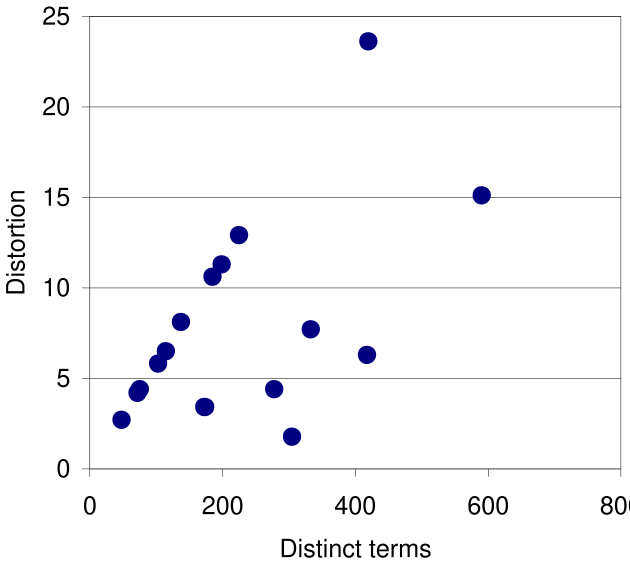

Figure 3: Scatter of distortion vs. the number of distinct tokens in a folder. Large folders are more susceptible to higher distortion because they often miss original documents in the HAC group mapping.

To study the prospects of PatchHAC, we plot a scatter of distortion vs. the number of distinct tokens in a folder in figure 3. Large folders are more susceptible to higher distortion. Closer inspection of SingleBest output shows that this is because they often miss original documents in the HAC group mapping.

5.4 PatchHAC and Bicriteria

The next important question is, can we improve beyond SingleBest by running PatchHAC? In figure 4 we show the results of a few iterations of PatchHAC. Indeed there is much room for improvement beyond SingleBest. In a few iterations, the total cost (broken up as distortion + (F, G) mapping + (D, G) mapping) comes down from 839+362+361 bits to 609+361+383 = **1354 bits**.

Finally we ran Bicriteria from the expensive starting point costing 2790 bits. Although Bicriteria is much slower than PatchHAC, it emerges as the winner after several iterations, with a cost of 550+361+395 = **1306 bits**. This is now getting close to the bipartite baseline cost. We conjecture that a better search strategy may even find a configuration with a cost that is lower than the bipartite baseline.

To make the trade-off between total mapping and distortion costs clear, we plot in figure 5 a scatter plot of the two costs for some of the tripartite configurations visited by our various algorithms. There is a clear inverse relationship. All the costs are summarized in figure 6.

5.5 Anecdotes

Inspection of the (F, G) and (G, D) maps revealed many interesting mergers, splits, and reorganizations. In particular, the (user-labeled) folders which mapped to a group gave an automatic annotation to the group, which is difficult in traditional clustering. As two examples, folders called *Linux*

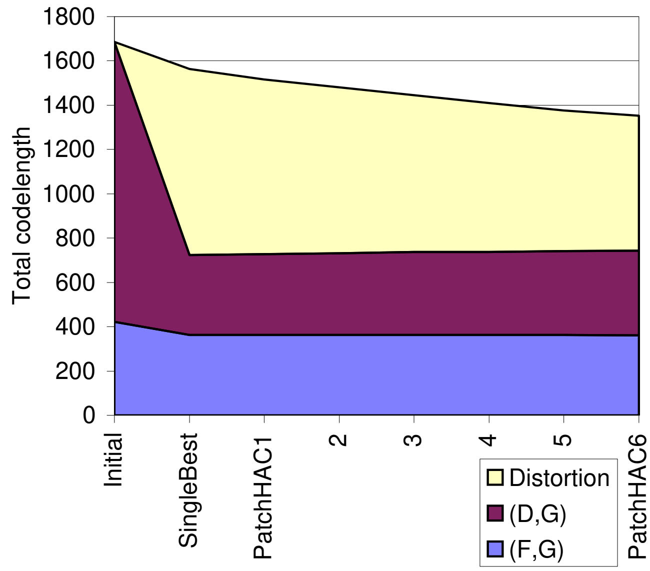

Figure 4: PatchHAC can reduce total codelength from where SingleBest left off. It does this by reducing distortion at the expense of a modest rise in (D, G) mapping cost.

and *Linux Stuff* were connected to a set of groups, and folders called *Broadcasting* and *Radio Links* were connected to a different set of groups. These folder names are very helpful in picking names for the groups. In many cases, these descriptive words were not very frequent in the documents associated with that group, and would therefore not be reported by term statistics alone.

6. CONCLUSION

We claim that a one-size-fits-all topic taxonomy is neither necessary nor sufficient to characterize the interests of an individual or a community of Web surfers. We propose the notion of *themes*, which are clusters discovered from the bookmarking activity of surfers. Themes try to simplify topics as far as possible while maintaining the individuality of surfers as far as necessary. We model this goal as a (computationally hard) compression/coding problem, and propose heuristics to optimize the objective code length. We show through experiments that there is a clear tradeoff between the complexity of representing the mappings between folders and documents, and the distortion suffered by the folder term distributions. We show through experiments that simple term-based content clustering algorithms are not as effective as our proposed algorithms in reducing code length.

We see several directions for further investigation. We would like to support folders which are members of a general hierarchy. Currently our formulation only supports a single level of folders. Our approach may be useful in further attacking the problem of clustering with categorical attributes; for that purpose, we will need to model documents which can belong to multiple hierarchies of folders simultaneously. Our heuristics are slow and greedy; it would be of interest to try to develop provably good algorithms. Finally, we are investigating other applications of the framework, e.g., in identifying the ‘semantic’ boundaries of hypertext aggregates.

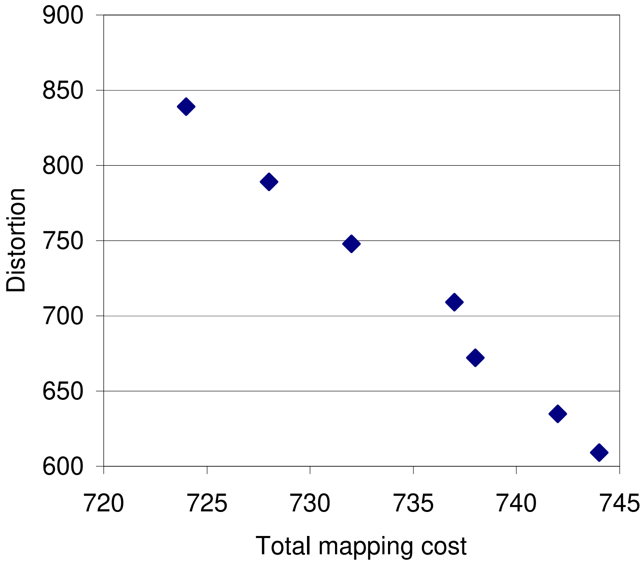

Figure 5: A scatter of distortion vs. mapping cost of tripartite configurations visited by our algorithms clearly show the tradeoff between these two costs.

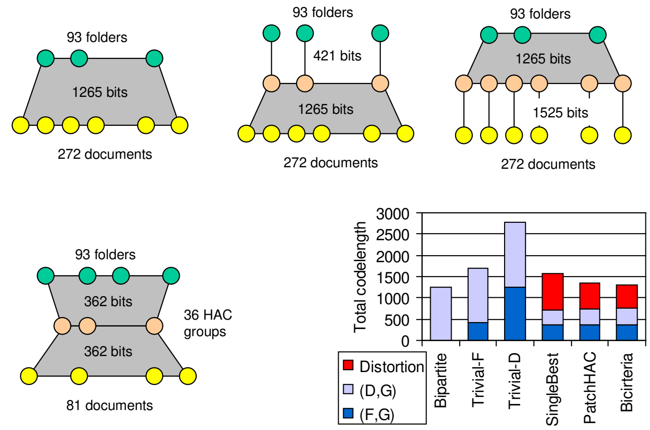

Figure 6: Summary of coding costs of all the algorithms, broken down into the two mapping costs and the distortion.

7. REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD Conference on Management of Data*, Seattle, June 1998. Online at `http://www.almaden.ibm.com/cs/quest/papers/sigmod98_clique.pdf`.

[2] R. A. Botafogo. Cluster analysis for hypertext systems. In *SIGIR International Conference on Information Retrieval*, volume 16, Pittsburgh, PA, June 1993. ACM.

[3] R. A. Botafogo and B. Schneiderman. Identifying aggregates in hypertext structures. In *Hypertext*. ACM, Dec. 1991.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th World-Wide Web Conference (WWW7)*, 1998. Online at `http://decweb.ethz.ch/WWW7/1921/com1921.htm`.

[5] V. Bush. As we may think. *The Atlantic Monthly*, July 1945. Online at `http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm`.

[6] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. In *7th World-wide web conference (WWW7)*, 1998. Online at `http://www7.scu.edu.au/programme/fullpapers/1898/com1898.html`.

[7] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD Conference*. ACM, 1998. Online at `http://www.cs.berkeley.edu/~soumen/sigmod98.ps`.

[8] S. Chakrabarti, S. Srivastava, M. Subramanyam, and M. Tiwari. Archiving and mining community web browsing experience using Memex. In *9th International World Wide Web Conference*, Amsterdam, May 2000.

[9] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[10] D. R. Cutting, D. R. Karger, and J. O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Annual International Conference on Research and Development in Information Retrieval*, 1993.

[11] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Annual International Conference on Research and Development in Information Retrieval*, Denmark, 1992.

[12] G. Das, H. Mannila, and P. Ronkainen. Similarity of attributes by external probes. In *International Conference on Knowledge Discovery and Data Mining*, New York, Aug. 1998. AAAI.

[13] D. A. Gibson, J. M. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In *VLDB*, volume 24, pages 311–322, New York, Aug. 1998.

[14] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, London, 1989.

[15] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[16] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Symposium on Discrete Algorithms*. ACM-SIAM, 1998.

[17] P.-S. Laplace. *Philosophical Essays on Probabilities*. Springer-Verlag, New York, 1995. Translated by A. I. Dale from the 5th French edition of 1825.

[18] E. S. Ristad. A natural law of succession. Research report CS-TR-495-95, Princeton University, July 1995.

[19] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[20] R. Weiss, B. Velez, M. A. Sheldon, C. Nemprempre, P. Szilagyi, A. Duda, and D. K. Gifford. HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Proceedings of the Seventh ACM Conference on Hypertext*, Washington, DC, Mar. 1996.