Persona: a contextualized and personalized web search

Francisco Tanudjaja — fcstanud@mit.edu

Lik Mui — lmui@mit.edu

Laboratory of Computer Science at MIT, Cambridge, MA 02139

June 1, 2001

Abstract

Recent advances in graph-based search techniques derived from Kleinberg's work [1] have been impressive. This paper further improves the graph-based search algorithm in two dimensions. Firstly, variants of Kleinberg's techniques do not take into account the semantics of the query string nor of the nodes being searched. As a result, polysemy of query words cannot be resolved. This paper presents an interactive query scheme utilizing the simple web ontology provided by the Open Directory Project to resolve meanings of a user query. Secondly, we extend a recently proposed personalized version of the Kleinberg algorithm [3]. Simulation results are presented to illustrate the sensitivity of our technique. We outline the implementation of our algorithm in the *Persona* personalized web search system.

1 Overview

Search engines index large numbers of documents and let users query desired documents. However, most search engines are not tailored to meet individual user preferences. [6] noted that almost half of the documents returned by search engines are deemed irrelevant by their users. There are several aspects to the problem. First is the problem of synonyms and homonyms. Synonyms are two words that are spelt differently but have the same meaning. Homonyms are words that are spelt the same but have different meanings. Without prior knowledge, there is no way for the search engine to predict user interest from simple text based queries. Secondly, search engines should be deterministic in that it should return the same set of documents to all users with the same query at a certain time. Therefore it is inherent that search engines are not designed to adapt to personal preferences.

Current information retrieval and data mining research tries to enhance user's web experience from several directions. One direction is to create a better structural model of the web, such that it can interface more efficiently with search engines. Another approach is to model user behavior as to predict users' interests better.

Along the lines of the former are efforts at better defining the meaning of queries themselves. The *Wordnet* project at Princeton University is an online lexical reference system that organizes English words into synonym sets [7]. A similar approach is to build a taxonomy of words. A taxonomy comprises of a tree structure in which a word belongs to a certain node, each with parents and children. A node's parent serves a general category that encompasses all of its children. A node may have children that are sub-categories of itself. An example of such word taxonomies are the *Open Directory*

Project [http://dmoz.org] and the *Magellan* hierarchy [http://magellan.excite.com]. Yet another approach is to create a semantic structure in machine readable format. As opposed to classifying content from a person's point of view, this method embeds meta data for classification, allowing document content to be machine readable. There are currently efforts at standardizing these classification, for example OIL (*Ontology Interchange Language*) and DAML (*DARPA Agent Markup Language*). Haystack [4] is an ongoing project in semantic meta data indexing.

Along the lines of the latter approach, various research in data mining and knowledge representation have build models to record user interest and predict user behavior. Ultimately, these user models interface with a system so as to give it *a priori* knowledge regarding user preferences.

Clearly, work in user profiling is closely related to building better personalized systems. Different methods of gathering user data is often coupled with various personalization systems. We found that the combinations that are available in the context of personalized search are unsatisfactory. We propose a novel approach in building a better system with the following. First, we extend existing theory with regards to personalized search. Second, we propose to model users interest using an interactive query scheme utilizing the web ontology provided by the *Open Directory Project.*

To support our argument, we have built an implementation of a personalized search engine. The system wraps a personalization module onto an existing search engine, and refines search results using the proposed extension of the graph based algorithm.

At its core, the proposed system utilizes a taxonomy of user interest and disinterest. We use a tree coloring method to represent user profiles. Visited nodes are 'colored' by the number of time it is visited, whether the user rate it as positive or negative, and

URLs that it associates to.

In addition, we run sets of controlled experiments to analyze the performance of each of the existing variants. The experimental results verify our predictions and confirm that the proposed extension performs better.

We offer a roadmap of this document. Section 2 outlines related work in personalized web browsing and reviews existing methods using graph based search algorithms. Section 3 describes our extension to existing theory. Section 4 describes the user modeling technique. Section 5 outlines the implementation of *Persona*. Section 6 describes the simulation results. We conclude in section 7 with some direction for future work.

2 Related Works

2.1 Examples of personalization applications

Personalization applications cover a range of spectrum. At one end of the spectrum, we have filtering systems, which filter input from an information resource. Information of possible interest are marked. An example of such a filtering system, SmartPush [8] combines several novel ideas together. The system finds information by means of semantic meta data to filter news articles. In addition, it builds the user profile using a simple hierarchical concept model. For example, under the category news, there are the categories sports, literature, economics, etc. The model records user preference by giving weightings to these nodes. This idea seeks to improve from the common *bag of words* approach in storing user profile. However, SmartPush requires news providers to provide the semantic meta data. The concept hierarchy is also determined by the content provider.

In the context of web browsing, there are several examples with regards to personalization systems. For example [9] uses implicit feedback to profile users' browsing behavior. In particular, the system analyzes activity logs of a proxy server that intercepts requests coming out of a gateway and logs browsing information. Topics of interest are calculated using a page interest estimator coupled with vector space techniques. From these, the system extracts an *n-gram* set of words, namely bigrams and trigrams, that represents user interest. The idea is to capture in the user profile sets of words that may have different meaning when coupled together (e.g bar tender). The paper also offers suggestions on which sets of words should be given more emphasis, such those within the bold tag and italic tag in an HTML page.

Another work by [6] describes personalized search based on a taxonomy. It uses an existing taxonomy from *Magellan*, which classifies documents into approximately 4400 nodes. Profiles are stored as concept hierarchies, as in the SmartPush system. Each node is associated with a set of documents. Each document is represented by a weighted keyword vector, determined by the number of occurrences of keywords. User browsing behavior is analyzed from web server log, and documents that are browsed are calculated into keyword vectors, and matched against documents associated with nodes. The node that corresponds to the closest match is stored in the user's profile. The system also takes into account temporal effects, i.e how long a user browses each document.

There are currently many available systems that allow for personalization to some extent. Some systems allow users to specify page content [e.g *My Yahoo*], while others recommends web pages, books, music, etc. [5] contains a survey of many of the available systems and their methodologies.

2.2 Graph based search techniques

The Hyperlink Induced Topic Selection (HITS) algorithm is a well known approach in information retrieval. We can transform the relationship between a set of documents into a directed graph, in which a node represent a document and a link from node i to node j represents a reference from document i to document j. If such a graph can be represented by an adjacency matrix $\mathbf{M}$, the authority vector $\mathbf{a}$, and the hub vector $\mathbf{h}$, we can find a converging, steady state solution using the *power method* from linear algebra [1, 2]. The i^{th} value of vector $\mathbf{a}$ represents the authority value for node i. The adjacency matrix $\mathbf{M}$ has value M_{ij} equals to one if there is a link from node i to node j, and zero otherwise.

In the context of personalization, we would like to indicate a preference on certain documents. HITS assumes all nodes are equal; their authority and hub measure are determined essentially by the number of in-degrees and out-degrees. Consider an example in which there are two clusters of document, and we indicate that we like document j in the second cluster better. Ideally, we want to 'lift' the relative authoritativeness and hubness of documents in the second cluster in relation to the whole document collection. Two variants of HITS that deals with such 'lifting' are introduced and described in [3]. The following describes each in turn.

2.2.1 Single node lifting

In single node lifting, the authority and hub measure of document j is lifted by augmenting the j^{th} component of either the authority vector $\mathbf{a}$ or the hub vector $\mathbf{h}$ at each iteration. By directly changing the value of the j^{th} element, the nodes that are connected to it as are also affected. We start with an initial vector $\mathbf{a}_o$ and add $\boldsymbol{\delta}$ at each iteration loop. Because the steady state value of $\mathbf{a}$ only depends on the eigenvectors of $\mathbf{M}^T\mathbf{M}$ [1], this amplification of the value a_j has to happen at each step of the iteration as to affect the original steady state. Intuitively, this makes sense: since we are interested in node j, we want to amplify its value at each iteration such that the original HITS steady state solution is somewhat shifted in favor of a_j.

2.2.2 Gradient ascent HITS

The gradient ascent HITS has a different approach in lifting the authority and hub measures for a node. The main idea behind gradient ascent is to find the values of M_{ki} in the adjacency matrix $\mathbf{M}$ that maximizes a_j or h_j. At each iteration, the algorithm takes a small step $\gamma \cdot \Delta M_{ki}$ for all k and i in a manner that increases the relative value of a_j or h_j. Accordingly, we have a new resultant matrix $\mathbf{M}^* = \mathbf{M} + \gamma \cdot \Delta\mathbf{M}$ at each iteration! This entails to *shifting the principal eigenvector* at each iteration in the direction of our document of interest.

This variant of HITS suffers from the common malaise of gradient ascent algorithms, namely the trap of local maxima. At each step of the iteration, the algorithm tries to maximize *locally* what step to take. The value of $\mathbf{M}^*$ that produces a possible increase in a_j is maximized with respect to the previous $\mathbf{M}$, not the *original* $\mathbf{M}$. Therefore, it only has a local view and optimize in that sense. Nevertheless, the algorithm does shift the solution away from the original eigenvector of $\mathbf{M}^T\mathbf{M}$ towards a new value that tries to increase a_j in the resultant vector $\mathbf{a}$.

2.2.3 Comparison and analysis

Both the single node lifting and gradient ascent provide a way to 'lift' an *individual* node in a cluster of documents. [3] claims that gradient ascent is superior to single node lifting. The single lifting approach looks at a node of interest j and directly changes the adjacency matrix so as to affect all the nodes that link to it directly.

In contrast, the gradient ascent node tries to readjust the values of *all* the nodes in the matrix $\mathbf{M}^T\mathbf{M}$ so as to lift a_j. It looks at all the values in the matrix and decides which ones should increase or decrease or stay the same so as to maximize the authoritativeness or hubness of a certain document.

Intuitively, the second approach is more elegant and robust and should therefore perform better. As our experiments in section 6 show, this is indeed the case.

3 Extensions to current techniques

We would like to extend the above algorithm with the following heuristic: *decrease the value of all nodes $l \neq j$.* The motivation is that we would like to see a *faster rate of change* for readjusting the relative rankings of authorities and hubs, by doing both gradient ascent and descent at the same time. Adding this simple heuristic is a natural extension of existing theory. [3] mentions the use of gradient descent to reduce the authority of irrelevant documents, but claims that negative weight values of a_j complicates the analysis. The following sections introduce two methods that explores this heuristic a step further.

3.1 Combination one

In this combination we take into account the contribution of lifting the node of interest j as well as the average contribution of pushing nodes $l \neq j$. However, we note that the average contribution of the nodes $l \neq j$ may be greater the contribution from lifting node j, for which the total step ΔM_{ki} for a certain i, j in the matrix $\mathbf{M}$ may be negative. We note that given an adjacency matrix $\mathbf{M}$ with values of ones and zeros, it is not possible to have a negative entry M_{ki} using the gradient ascent method. Now that we are pushing irrelevant nodes, these values may be negative.

Noting the contribution from lifting node j to be ΔM_{ki_j} and the contribution from lifting node $l \neq j$ to be ΔM_{ki_l}, we apply the following rule: *if* $-\Delta M_{ki_j} \leq \frac{1}{N-1} \sum_{l\,,l \neq j} \Delta M_{ki_l}$, *then* $\Delta M_{ki} = \Delta M_{ki_j}$ *else* $\Delta M_{ki} = \Delta M_{ki_j} + \frac{1}{N-1} \sum_{l\,,l \neq j} \Delta M_{ki_l}$. That is to say: if the effects of ΔM_{ki} from lowering the authorities of all nodes $l \neq j$ is such that the authority of node j is lowered, we *ignore* their contributions to ΔM_{ki}.

3.2 Combination two

In this second extension, we loosen our previous restriction, and for all cases, let $\Delta M_{ki} = \Delta M_{ki_j} + \frac{1}{N-1} \sum_{l\,,l \neq j} \Delta M_{ki_l}$.

Now it is possible to have negative values for ΔM_{ki}. The justification for this method is the fact that we essentially care only about the relative rankings of the node, and even if lowering node l contributes to lowering the *nominal* value of a_j or h_j, we allow it to happen because the *relative* value of a_j is still greater than the lowered node a_l.

For a gradient ascent and descent algorithm running at n iteration, we can do the following to attenuate possible negative values of ΔM_{ki}:

- At *odd* values of n, calculate ΔM_{ki} as $\Delta M_{ki} = \Delta M_{ki_j}$.
- At *even* values of n, calculate ΔM_{ki} as $\Delta M_{ki} = \Delta M_{ki_j} + \frac{1}{N-1}\sum_{l\,,l\neq j} \Delta M_{ki_l}$.

Alternating between these two give the algorithm some time to readjust its values in the cases where M_{ki} is negative.

In general, we expect both combinations to have a faster rate of readjustment than the normal gradient ascent HITS. We analyze the performance of each of these variants in section 6.

4 Ontology based user profile modeling

4.1 Design criteria and choice

Gradient ascent HITS provides a method to bias certain documents in a graph based structure. What is lacking, however, is a method of keeping track the history of user interests. In this respect, we propose a model of user profiling to complement our theoretical extension of graph based search.

We found that most user models lie in the common *bag of words* approach. Strings of words that represent user interests are kept in the form of web browser *cookies* or files. Moreover, most user models do not take into account what users *dislike*.

In the context of our search engine, this approach is inadequate for several reasons. Firstly, the bag of words approach does not consider the semantics of words. For example, when users indicate liking for 'cars', this approach does not consider other words such as 'automobile.' Likewise, when users indicate liking for 'rose' in the sense of 'wine,' as opposed to 'flower,' the bag of words approach lacks an efficient method to make a proper distinction. In other words, we run into the problem of homonyms and synonyms. Secondly, by ignoring what users dislike, the system does not learn from past mistakes. Though this approach of using only positive feedback is safer, it does not put the set of dislikes in proper perspective.

We propose an approach that uses a tree coloring technique. The tree is an *Open Directory Project* (ODP) taxonomy, which contains nodes that represent semantic contexts of web pages. We keep a record of visited nodes, and 'color' each by the number of times it has been visited, the number of times the user rates it positive or negative, and URL's that the node associates with.

ODP is a multiply connected tree. In the tree itself, ODP has only one parent, but its format allows multiple aliasing, so in effect a node may have multiple parents. In addition, each node is associated with an ID number and a set of 'leaves' which are the web pages associated with the node.

The user profile is a mapping from a context to a set of ODP nodes. A context is defined as a user query. For each query, the system generates a set of pages. Users can rate pages as being 'good' or 'unrelated.' Since each page is associated with a node in ODP, this feedback is updated into the user profile. Each entry in the mapping has the number of times the node has been visited, the number of positive and negative feedback for the node and the set of URL associated with it. Figure 1 (a) displays a schema of the user profile.

From our discussion in section 2, we found several methods that also uses this tree coloring scheme. Note, however, that this approach, although along the same lines, are distinct from these other methods. [6] uses a tree weighting scheme to calculate the vector space model of documents associated with each node. Documents of interests are processed and its vector space value matched against the vector space values of the nodes. The weights - in our case, colors - of the tree does *not* change. The other example, SmartPush [8], uses a taxonomy provided by news provider to determine which news articles to reccomend. This tree is dynamic, but does not cover the breadth and depth of our proposed system. Hence, although the idea of tree coloring is not new in itself, the way we combine it with the system is quite distinct.

4.2 Use cases

The following summarizes our user modeling technique:

- *Data gathering*

  Our model gathers data using explicit feedback. Users are allowed to rate a certain context positively and negatively. Feedback will be recorded in the user's profile.

- *Representation*

  A user profile is a mapping from contexts to nodes. One context may map to several nodes. For example, the context 'car' may map to nodes that represent 'Honda', 'Volvo', etc. Each node has a 'color' that encodes the number of times it has been visited, rated positive, negative, and associated URL's.

- *Interpretation*

  The table is kept as the user profile. When a user submits a query, the system does a table lookup to find the context. The following happens:

  - If found, the system looks at the mapping of nodes, and accordingly give more or less weighting to its associated URL's to be filtered.

- If not found, the system tries to associate that context with an ODP node. There may be several nodes that can be associated with the context. For each of these nodes, look up all the nodes in the table and check if either:

  1. The node associated with the query has the same parent as any node in the table.
  2. The node associated with the query is a child of any of the node in the table.

  If any of the above is true, return the associated URL's and their respective weights to be filtered; else, return nothing.

The results are passed back to the graph based search algorithm from the previous section. Nodes with positive weights are given positive bias, while nodes with negative weightings are given negative bias. Note that the current prototype implementation simply filters out the negatively weighted URL's. It searches only up to one depth up and down the tree to look for parents and children when comparing nodes.

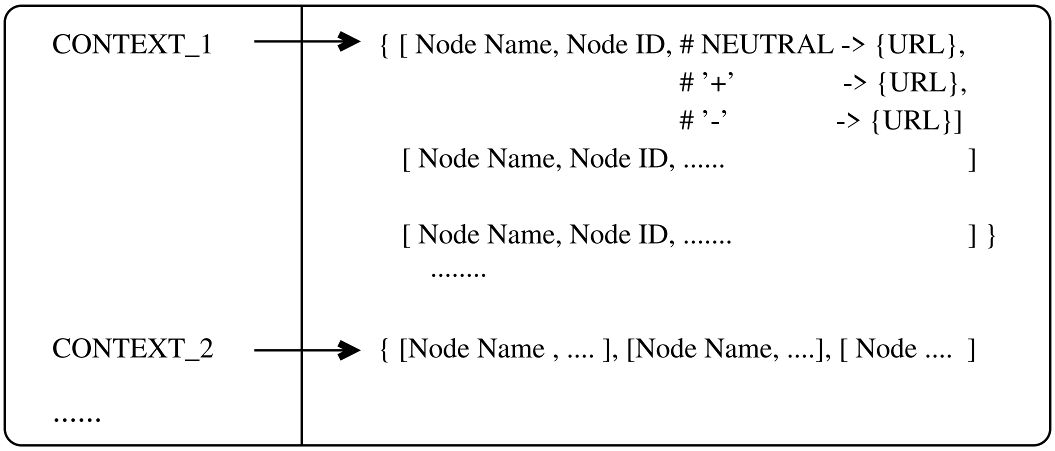
We note that there is much room for improvements. For example, generalizing the node searching mechanism up to n nodes up and down the classification tree, we observe that the nodes are more generic as they reach the root node. We can add the following simple heuristic: *the closer a node is to the root, the less depth the tree will be searched.* So instead of finding up to depth n for each node, the depth should be a function of how close a node is to a tree.

To illustrate clearly our profiling system, we give a simple example as drawn in Figure 1 (b). In the past, a user had queried 'vision' and was given two set of results, one relating health, another regarding computer science (machine vision). The user indicated that he or she preferred vision in the health sense, and rated vision in the computer science sense as negative. Next the user queries the word 'virus'. The system does not have any information regarding the user's preference on 'virus' and therefore looks at all the nodes in the profile table. Searching for 'virus' in ODP, the system finds two nodes, one in *health/virus* and another in *computer_science/virus*. Matching the user's profile shows that the user has indicated interest in the node *health/vision*, a negative weighting on *computer_science/machine_vision*, finds that *health/vision* has the same parent as *health/virus* and incoorporates this fact in returning the results.

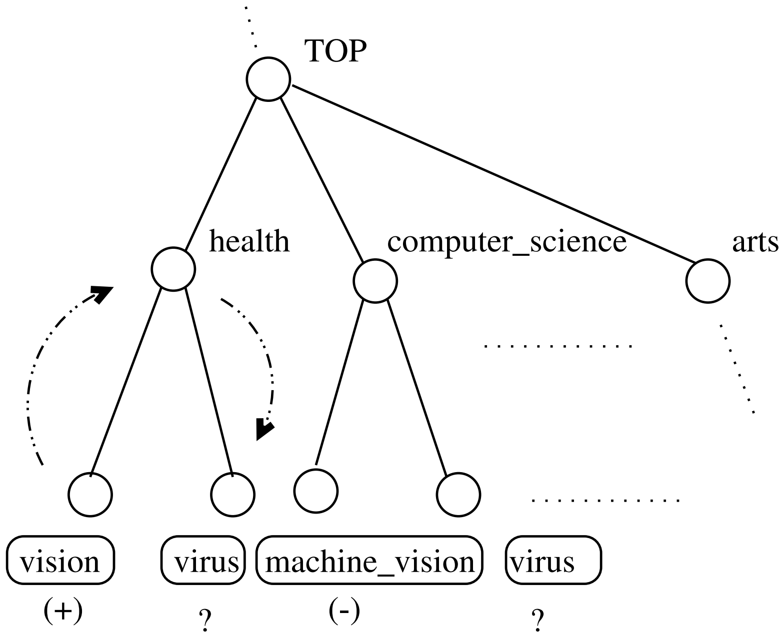
Clearly, this ontological approach is beneficial in the sense that the system can better predict user interest and further classify them into separate categories - leveraging on the semantics inherent in an ontology. Instead of trying to capture the meaning of words *per se*, an ontological profile captures the semantics of user queries, thus enabling it to find synonyms or related contexts as well as hierarchical relationships. Therefore we overcome many limitations of the standard bag of words approach.

Moreover, recent work by [10] introduces a technique for static matching of several users who had two 'colored' ontologies. This method has possible applications in the area of interest matching and collaborative filtering.

In summary, the proposed user model introduces improvements over the common bag of words approach and various other techniques. Given the design constraints, we feel that it is the most feasible technique. The model is quite scalable and modular and is easily extendible to other applications.

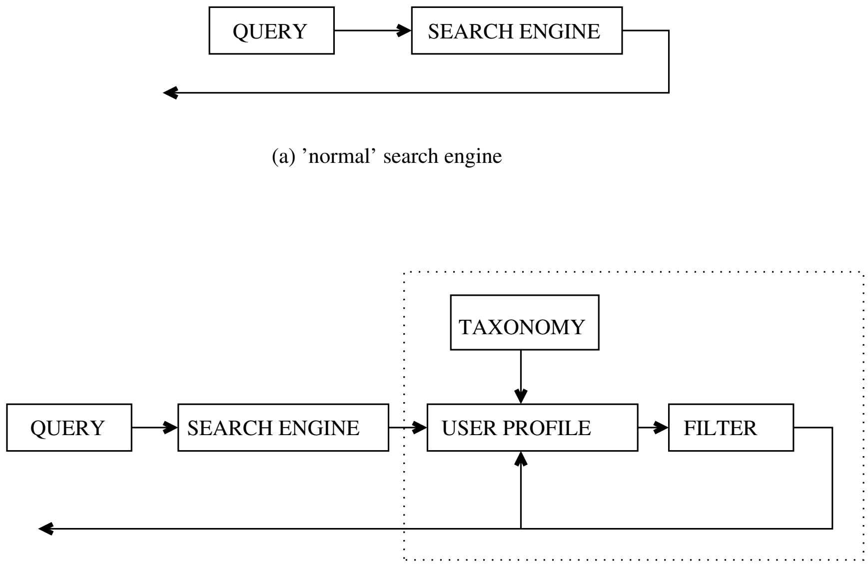


(a) User Profile



(b) Sample case

Figure 1: Building and using the user profile (a) depicts a schema of the user profile, (b) is an example of a use case



(a) 'normal' search engine

(b) Persona system wraps around existing search engine

Figure 2: System diagram

5 System description

The *Persona* system is a personalization module that wraps around a search engine. It lies between the search engine and the end user, refining the results coming out of a 'normal' search engine. Figure 5 shows a diagram of the system.

The system consists of a front end and a back end. The front end interfaces with the user, accepting queries and user feedback. These are then passed to the back end, which processes these queries and builds the user profile. The back end core of *Persona* consists of two main modules: a filtering mechanism and a user profiling system. The following sections describe each in turn.

5.1 Overview of filtering mechanism

The brief overview of the filtering mechanism is as follows:

1. *Query input*

   The user inputs a query, which will then be outsourced to *dmoz*, an existing search engine - the result of which will be filtered to leave the top n results. These n documents corresponds to nodes in the ODP taxonomy.

2. *Personalization Agent*

   First, the system consults the user profile to check the user's history. If there is a match, the system incorporates past user preference. If not, a 'normal' filtering module then processes these results. The HITS algorithm as described in [1] dictates the following:

   (a) Call the initial set of results the *root set*.

   (b) Expand the *root set* by order one distance, such that all web pages that point from and to the root set are included. This set of node is the *base set*.

   (c) Treating each web page as a node and each URL in that web page as a link, create a directed graph structure consisting of all members of the base set. In this calculation, the nodes that are from the same domain are not taken into account and are thus filtered out.

   (d) Using the number of in-degrees and out-degrees from each node, the algorithm calculates the *authoritativeness* and *hubness* of each node. Based on a node's authoritativeness, the results are ranked accordingly.

   (e) The ranked results are then updated against the user profile.

3. *Feedback-based Result Refinement*

   The system returns the filtered results to the user. The user may give positive or negative feedback on the returned set of documents. The system will then *refine* the current results based on these preferences, giving more weighting to the positively rated pages and less to the negatively rated pages.

In addition, these user feedback are incorporated into the user profile. The user profiling is discussed in the next section.

5.2 User Profile

The user profile relies on relevance feedback. Each positive and negative feedback serves two function. First is to refine the set of searches and re-rank the results. Second is to build the user's profile. The user feedback is updated by 'coloring' nodes as we described in the previous section.

Each entry in the user profile consists of a context word, which consists of queries that the user types in. Each of this entry is associated with a set of nodes. Each node in the ODP has a unique identifier. The user profile keeps track of how many times each node in the profile is visited, the number of positive ratings, negative ratings, as well as the set of URLs associated with it.

In this manner, we do not keep the whole ODP taxonomy inside the user profile. We only keep track of the nodes that has been colored. The user profile then can be kept small, allowing for scalability.

The system consults the user profile at every new query. If a query exists in the user profile, it returns the set of URL's associated with the colored nodes. If there is no data on the query, the user profile finds the set of ODP nodes that closely matches the query and tries to find nodes in the profile that may be its parents or children.

In the case of user feedback, the user profile simple colors related nodes and passes on the bias information to the variants of HITS. These variants will take the bias into account and return the set of most related documents.

We note that most search engines have the feature that lets users browse through 'similar pages.' We claim that this refining technique is different from ours. Finding 'similar pages' usually entails returning the closest document set that the search engine indexes. In contrast, our filtering mechanism expands a set of URL's and emphasize those with positive feedback. It expands up to depth three down to create a new graph structure, and lift those documents which are positive.

6 Experimental results

6.1 Controlled experiments

In the controlled experiment, we generate clusterings of documents. Each cluster consists of a fixed number of documents, and they are bind to a certain context. Each document has a fixed number of links. There are two types of links, ones by which a document points to a document in a the same cluster, and ones by which a document points to another document in a different context. To generate a statistically reliable data set, the way each node interacts with one another is determined stochastically. To following summarizes the experiment parameters:

- *Number of clusterings or context.*

  Each generated set contains a fixed number of contexts or topics. Each node k in the set is attached to a context $c_k \in \{C_1, C_2, ..., C_M\}$ for M contexts.

- *Number of documents in a cluster: D.*

  To create a set of balanced clusters, the number of documents per cluster is fixed.

- *Number of links per document: N.*

  Again, to create a set of balanced clusters, the number of links per document is fixed. This parameter is upper bounded by the number of documents per cluster.

- *Probability that a link connects two documents in the same cluster: p*

  Define p to be Prob($l_{ij} \mid c_i = c_j$) and $1 - p$ to be Prob($l_{ij} \mid c_i \neq c_j$), where l_{ij} is a link from node i to node j. In other words, p denotes the probability of a document pointing to another document of the same context, and $1 - p$ denotes the probability of a document pointing to another document with a different context.

- *Gamma (γ)*

  γ is the step size of the gradient ascent, and the perturbation size in single node lifting.

- *Number of iteration*

  Even though in theory would like to find the steady state value of the authority and hub vectors, in practice we are only interested in the *relative* rankings of the nodes, which stays relatively the same after four or five iterations.

Figure 3 illustrates an example of a possible clusterings of documents - represented as nodes - as well as their interaction. Since links are probabilistic, each generated nodes-set has varying graph structures.

Now that we have this platform, we want to create test harnesses based on our algorithms. In particular, we would like to see the effects of lifting using all four variants of HITS.

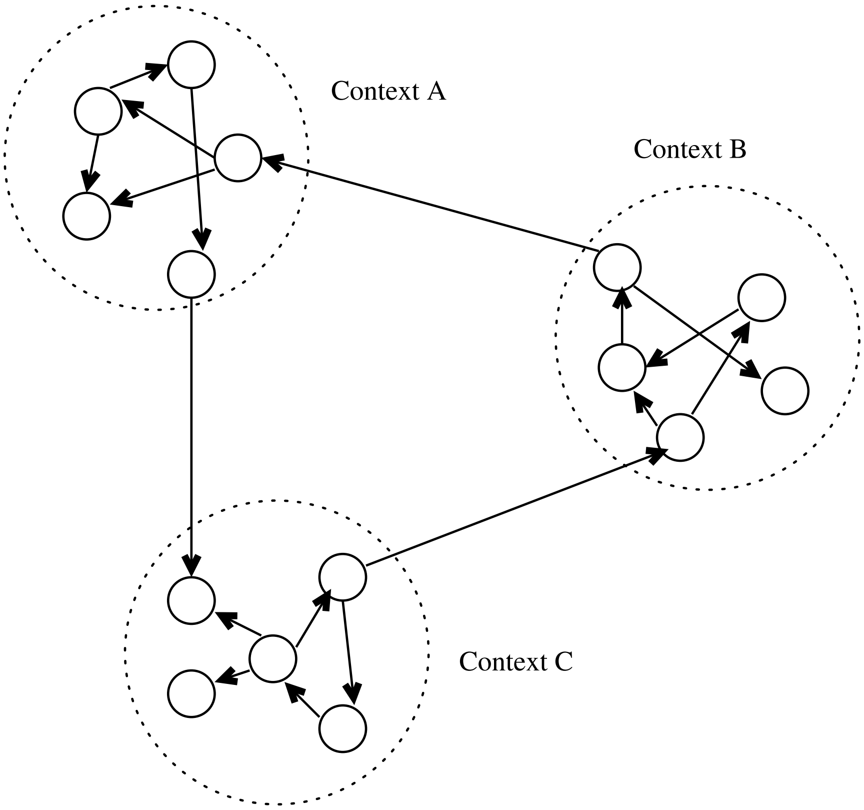


Figure 3: View of clusters of documents tagged to different context cross referencing each other

Parameter	Variable	Value
Number of categories	M	3
Number of documents per category	D	10
Number of links per document	N	8
Prob($l_{ij} \mid c_i = c_j$)	p	-
Prob($l_{ij} \mid c_i \neq c_j$)	$1-p$	-
Weighting factor	γ	0.5
Number of iterations	-	5

Table 1: Parameters for a sample data point

6.1.1 Performance Metric

We would like to have a performance metric to quantify these results. We want to measure performance on two criteria: how much the average relative rankings of relevant nodes are increased by the technique, and how much of the average relative rankings of irrelevant nodes are decreased. By relevant nodes, we mean nodes that has the same context as the lifted node. To quantify this fact, we use a performance metric described as follows:

- Given D number of documents per category, and node i with rank R_i and context $c_i = c_{lift-node}$, we measure the average 'lift' L to be:

$$L = \frac{1}{D} \cdot \sum_{\forall i \mid c_i = c_{(lift-node)}} (R_{i_{old}} - R_{i_{new}})$$

- The average 'suppress' S of all nodes j with context $c_j \neq c_{lift-node}$ is defined to be:

$$S = -\frac{1}{D \cdot (M-1)} \cdot \sum_{\forall j \mid c_j \neq c_{(lift-node)}} (R_{i_{old}} - R_{i_{new}})$$

Note that L and S are not independent metrics. For example, if all the rankings stay the same, that is, if $L = 0$ then there is very high probability that S is also equal to zero. If most of the rankings change - i.e L is relatively high - then S is very likely to be relatively high as well.

Using these performance metrics, we can perform several runs and accumulate data for statistical analysis.

6.1.2 Simulation Results

We are interested in the relationship between probability of a node pointing to another node of the same context p to the performance metrics L and S. Using the parameters described in Table 1, we vary the value of p from 0 to 1, with increments of 0.1. At each interval, we calculate the L and S metrics for five different trials. With higher p, we expect the number of documents lifted or suppressed to be larger. The relationship should be somewhat linear. In the following, we will run two sets, one for authoritativeness measure, another for hubness measure. We refer to our theoretical extension from section 3 as gradient ascent combination one, or gradient ascent^{+}, and gradient ascent combination two, or gradient ascent^{++}.

The discussion is broken down into the two subsections. First we distinguish between *authority* and *hub* measures. Second we separate the L metric from the S metric.

6.1.3 Authority simulation results

First we analyze the results for the L metric. We graph the set of points to better visualize the results. Figure 4 plots this graph in its point representation and its least square estimate for each cluster of data. In this simulation, the line that represents single node lifting is very close to the line that represents gradient ascent^{+}. The line that represents gradient ascent is very close to the line that represents gradient ascent^{++}. The shape of all the lines, however, are linearly increasing, which is what we should expect.

In section 3, we discuss that both extension of the gradient ascent method should perform better. However, the graph merely shows that gradient ascent^{+} performs at least as well as single node lifting, and gradient ascent^{++} performs at least as well as the original gradient ascent. We had expected both variants of gradient ascent should converge faster to a solution that differentiates between lifted and suppressed nodes.

What we can infer from this graph is the difference between the performance of gradient ascent in comparison to the single node lifting variant of HITS. Single node lifting at *most* lifts as much documents than the gradient ascent variants. With these tentative results, we look at the next graph.

Figure 5 plots the data points and its least square estimation for the S metric. Here the results are more encouraging. We see that the gradient ascent^{++} does confirm our theoretical expectations. However, gradient ascent^{+} performs in between single node lifting and the original gradient ascent.

This substantial difference suggests that both extensions of gradient ascent works better in *suppressing* irrelevant documents than *lifting* relevant documents. With this fact in mind, we move on to the next analysis.

6.2 Hub simulation results

This section analyzes the results from running the simulation to calculate *hub* measure. As before, the raw data and least square estimate for the L metric is calculated. Figure 6 plots these values.

As in the previous L metric analysis, the original gradient ascent is very closely

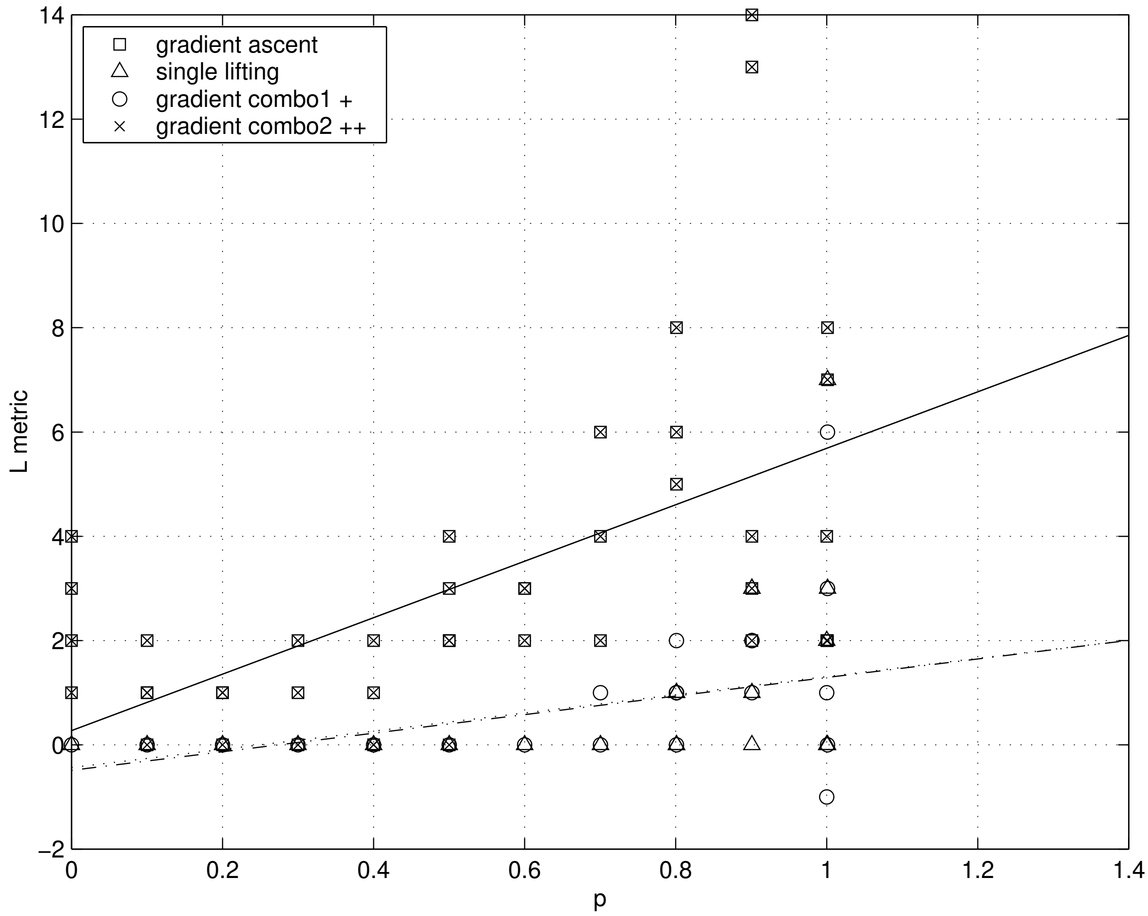


Figure 4: Data points for authoritativeness measure using metric L. The top line represents the least square estimate for gradient ascent and gradient ascent^{++}. The bottom line represents the estimate for gradient ascent^{+} and single node lifting, the other two merged into one.

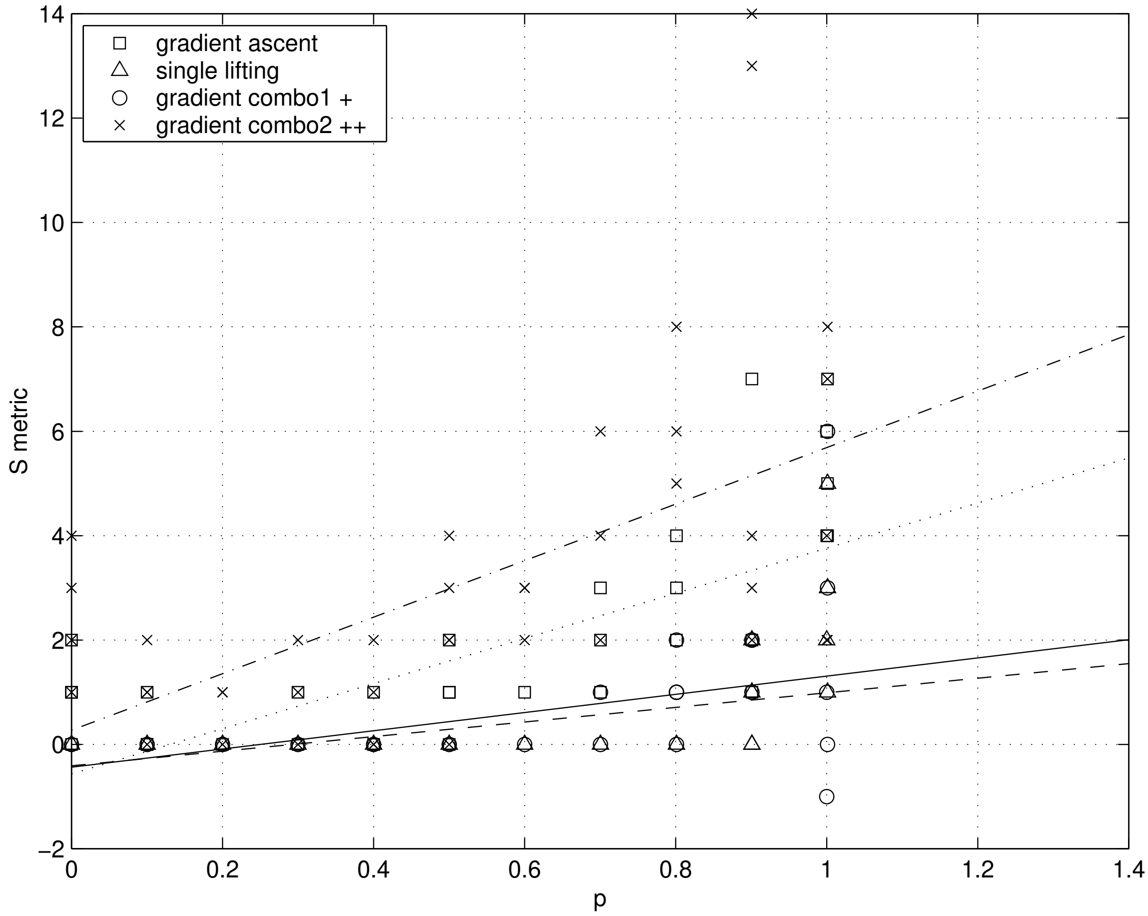


Figure 5: Data points for authoritativeness measure using metric S. The dashed line at the top is the least square estimate for gradient ascent^{++}. The dotted line represents normal gradient ascent. The solid line represents gradient ascent $^{+}$. Lastly, the bottom line represents single node lifting.

connected to gradient ascent^{++} in that they are practically collinear. The other gradient ascent variant, however, fare better than in the previous simulation, though it is still distinctly below the original gradient ascent.

We note that the metric values are negative in this simulation for p less than 0.4 The reason is that for low p, the lifted document will be linked to many irrelevant documents. The lifted hub will pull up these documents along with it.

The S metric analysis, however, shows interesting results. In this simulation, as depicted in Figure 7, *both* variants of the gradient ascent perform better than the original gradient ascent and the single node lifting variant. Again, this follows the trend that we see in our previous S metric analysis: *both extensions of gradient ascent suppress irrelevant document better than they lift relevant documents.*

Figure 7 validates our expectation that combining both gradient ascent and descent results in a *faster rate of change.* The slope is steeper for both gradient ascent^{+} and gradient ascent^{++} than it is for the other two techniques.

6.3 Summary of analysis

Though the results are somewhat mixed, we can conclude the following results:

- The first combination of gradient ascent, or gradient ascent^{+} *at least* performs as well as single node lifting.
- The second combination of gradient ascent, or gradient ascent^{++} *at least* performs as well than the original gradient ascent.
- Both combinations does better at *suppressing* irrelevant documents rather than *lifting* relevant documents.

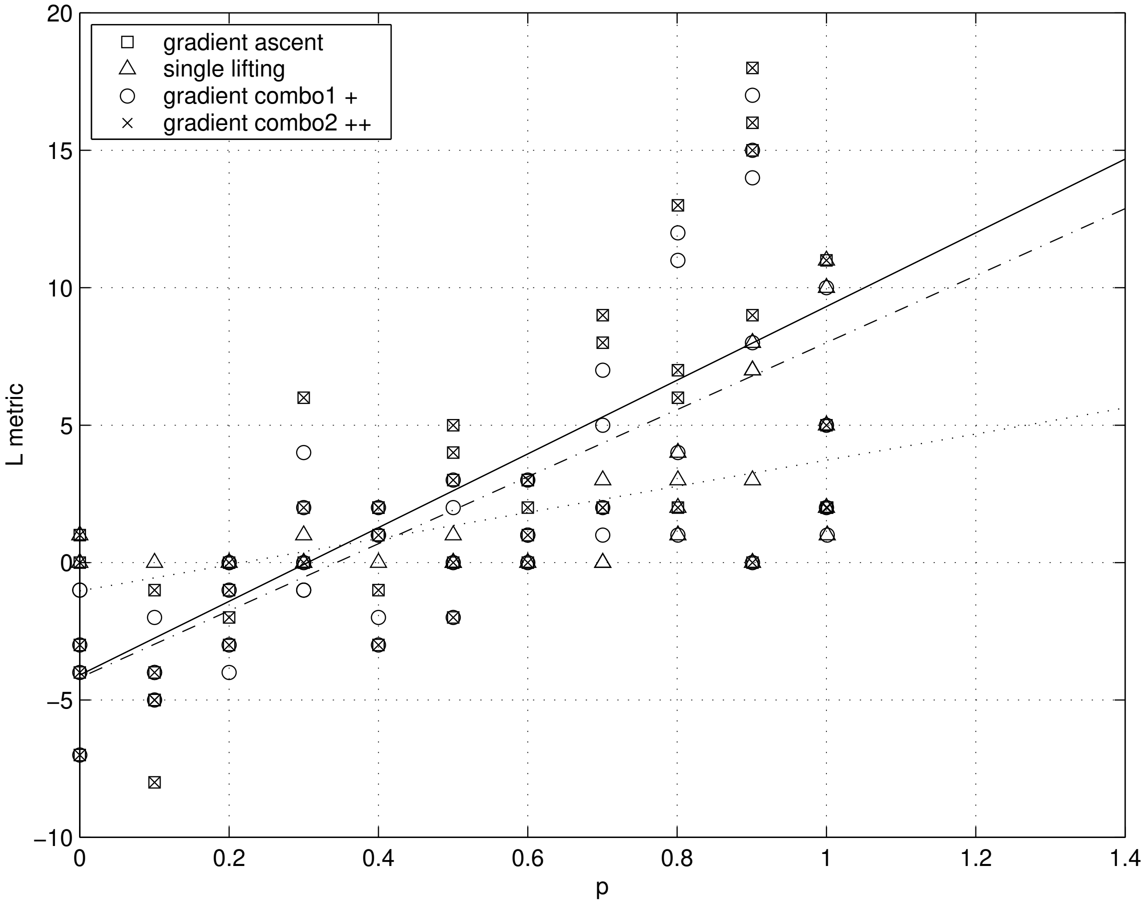


Figure 6: Data points for hubness measure using metric L. The dotted line is the least square estimate for single node lifting, the dashed line for gradient ascent combo 1, the other two merged into one.

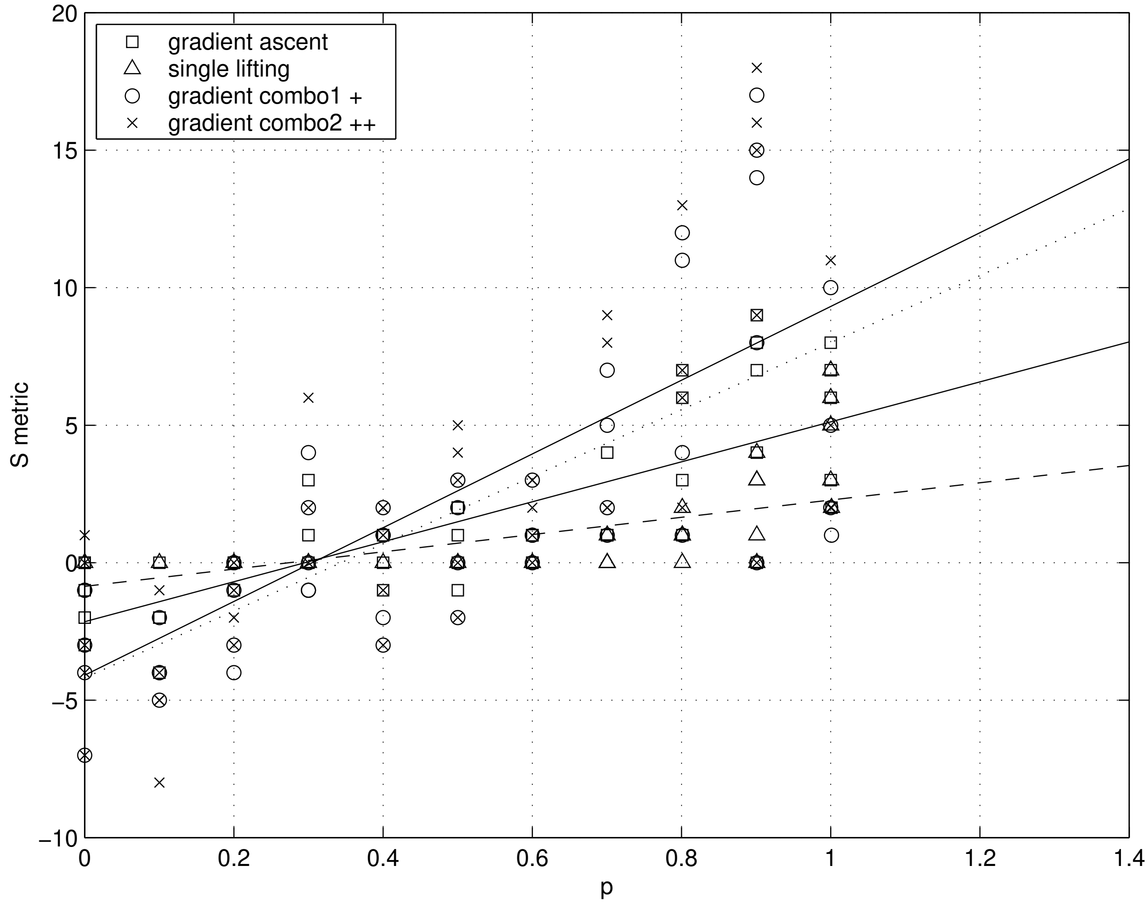


Figure 7: Data points for hubness measure using metric S. The top solid line is the least square estimate of gradient ascent^{++}. The dotted line is the estimate for gradient ascent^{+}. The second from the bottom solid line is for normal gradient ascent. The dashed line is for single node lifting.

The reason why our first combination does not work as well may be that the approach is 'half hearted' in that instead of totally maximizing the contribution of preferred document j and minimizing the contribution of other documents $l \neq j$, we only do so when the *nominal* value of j at least increases. The result is a gradient step size that does neither. Our second approach is more effective because we only concern ourselves with the *relative* value of document j in comparison to other nodes. In the end, this does indeed produce better results.

6.4 Real-time analysis

The purpose of the set of validation and testing above is to complement the system implementation, since it is not possible to run controlled experiments in this real time system. However, we did test the *Persona* prototype to check if it produces reasonable results. We tried using several queries, namely: machine learning and virus. For the machine learning query, we indicate that we like a page that relates machine learning to games, and indicate all other unrelated pages as negative. After submitting our feedback, *Persona* returns seven out of ten pages relating to machine learning, as opposed to one out of ten pages in the previous case. As for the virus query, we got back two sets of pages, one pointing to health related virus, and another to computer related virus. We indicate as positive all the pages that are health related, and indicate as negative all the pages that are computer related. *Persona* returns a set of ten pages all relating to health. We explained in section 5 how our method is different than the 'click to find similar pages' that we find in other search engines.

However, the prototype as it stands needs much improvement, particularly in terms of speed. Right now, for every page, it will open all the URL's in real time, causing the system response to be very slow. We need to add a caching and a multi threading model to hide latency. The filtering process itself is relatively fast; the system can process a few hundred nodes in less than one second.

A snapshot of the *Persona* user interface is shown in Figure 8.

7 Conclusion

In this paper, we perform analyses of several methodologies in graph based search algorithms and extend existing theory. In addition, we develop a robust, scalable user model using a taxonomy structure as provided by the *Open Directory Project*. Our goal is to create a system that better personalizes users' web experience. In hoping to achieve this goal, we implemented *Persona*, a personalization module that wraps around an existing search engine.

Our extension of current theory is derived from the gradient ascent variant of HITS. We experiment with two possible combinations of gradient ascent. From running sets of simulations, we verified that gradient ascent is superior to single node lifting. One of our proposed combination managed to perform at least as well as the gradient ascent variant. We come up with robust quantitative results regarding the performance of these techniques.

We build a user model that improves upon current existing methods. Though it is not feasible to test the effectiveness of our user profiling technique, several tests with the *Persona* system has shown positive results. Moreover, recent models has been developed to perform static matching among user profiles to determine similarities between users [10]. We find this very encouraging, and may extend further work in

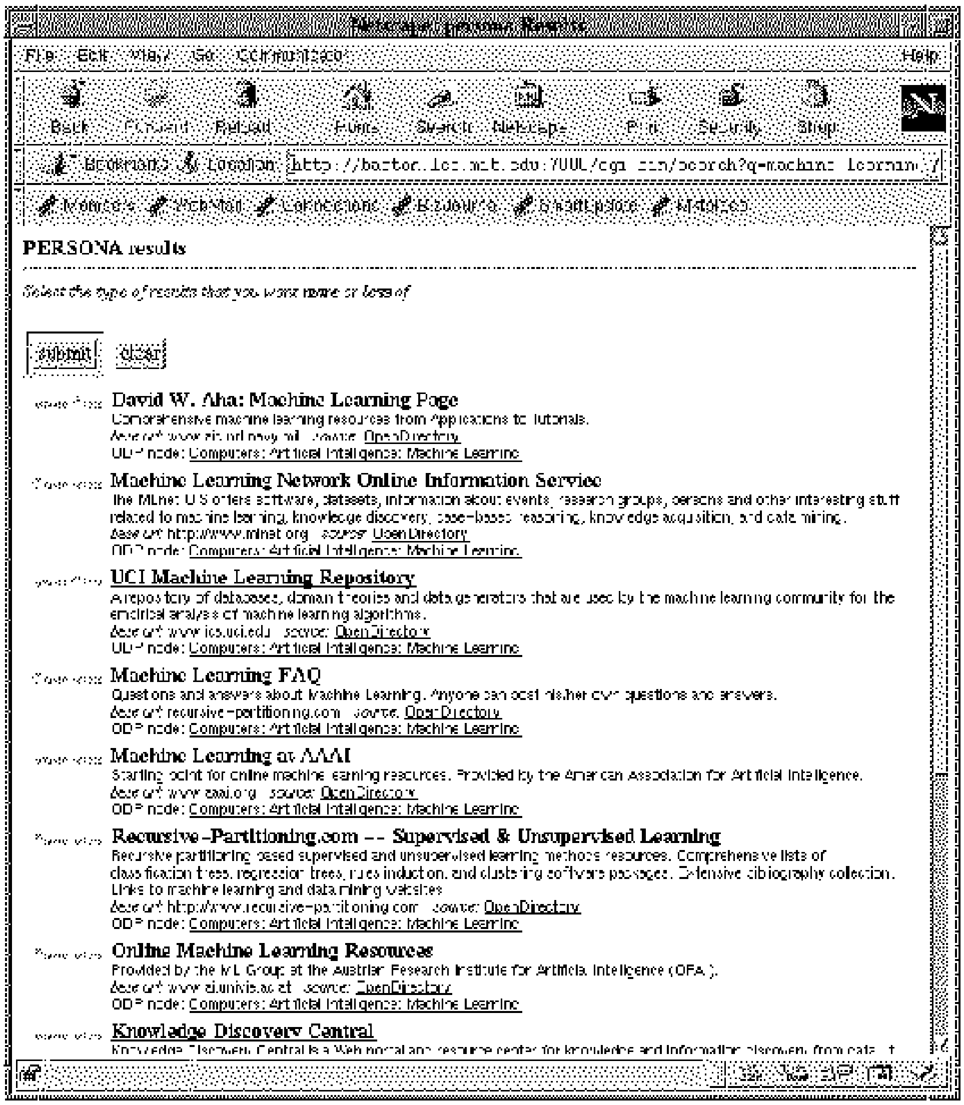


Figure 8: Snapshot of the *Persona* system

this direction.

References

[1] Kleinberg, J., *Authoritative Sources in a Hyperlinked Environment*, in Proceedings of the ACM-SIAM symposium on Discrete Algorithms, 1997.

[2] Kleinberg, J.; Kumar, R.; Raghavan, P.; Rajagopalan, S.; and Tomkins, A., *The Web as a graph: measurements, models, and methods*, invited survey at the International Conference on Combinatorics and Computing, 1999.

[3] Chang, H.; Cohn, D.; and McCallum, A., *Creating Customized Authority Lists*, in Proceedings of the Seventeenth International Conference of Machine Learning, 2000.

[4] Adar, E; Karger, D; and Stein, L., *Haystack: Per-User Information Environments*, in Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, 1999.

[5] Pretschner, A. and Gauch, S., *Personalization on the Web*, University of Kansas, ITTC-FY2000-TR-13591-01, 1999.

[6] Pretschner, A. and Gauch, S., *Ontology Based Personalized Search*, in Proceedings of the Eleventh IEEE International Conference on Tools with Artificial Intelligence, 1999.

[7] Fellbaum, C., *WordNet: An Electronic Lexical Database*, MIT Press, 1998.

[8] Kurki, T.; Jokela, S.; Sulonen, R.; and Turpeinen, M., *Agents in Delivering Personalized Content Based on Semantic Metadata*, in Proceedings of AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace, 1999.

[9] Chan, P K., *A non-invasive learning approach to building web user profiles*, in Proceedings of ACM International Conference on Knowledge Discovery and Data Mining, 1999.

[10] Koh, W., *An information theoric approach to interest matching*, MIT, 2001.