Web Projections: Learning from Contextual Subgraphs of the Web

Jure Leskovec *
Carnegie Mellon University
Pittsburgh, PA, USA
jure@cs.cmu.edu

Susan Dumais
Microsoft Research
Redmond, WA, USA
sdumais@microsoft.com

Eric Horvitz
Microsoft Research
Redmond, WA, USA
horvitz@microsoft.com

ABSTRACT

Graphical relationships among web pages have been leveraged as sources of information in methods for ranking search results. To date, specific graphical properties have been used in these analyses. We introduce *web projections*, a methodology that generalizes prior efforts on exploiting graphical relationships of the web in several ways. With the approach, we create subgraphs by projecting sets of pages and domains onto the larger web graph, and then use machine learning to construct predictive models that consider graphical properties as evidence. We describe the method and present experiments that illustrate the construction of predictive models of search result quality and user query reformulation.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Algorithms, Experimentation.

Keywords: web graph, web search, web projection, contextual subgraph, query reformulation

1. INTRODUCTION

Information retrieval methods have traditionally considered documents as independent. A key insight coming to the fore with the pursuit of effective web search is that inferences about relevance can be enhanced by considering the hyperlink relationships among documents [13, 21]. We present a methodology we refer to as *web projections* that centers on the creation and the use of graphical properties of subgraphs of the web. With the approach, we project a set of web pages of interest, such as the results generated by a search engine for queries, on the larger web graph to extract a subgraph, which we call the *web projection graph*. We then identify and exploit graph-centric properties of this subgraph for a variety of search-related tasks. The method can be viewed as a general approach of using context-sensitive collections of web pages to define and focus attention on relevant subsets of the web graph, and then using graph-theoretic features within this subgraph as input to statistical models that can provide predictions about content, relevance, and user behavior.

*Research performed during an internship at Microsoft Research.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

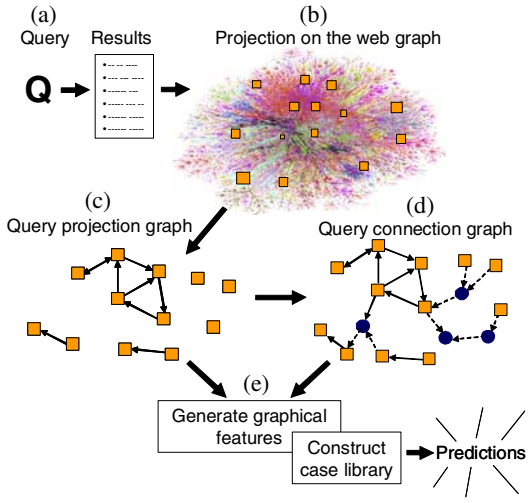
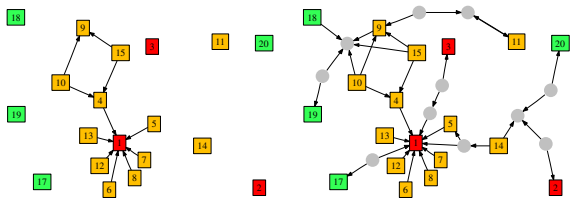


Figure 1: Web projection methodology. Given a query and respective search results, a query projection graph is generated and graph-theoretic features are then used for building predictive models.

We highlight in this paper the use of the subgraphs for analyzing search result quality and for predicting user behavior in reformulating queries. Specifically, we investigate the following questions:

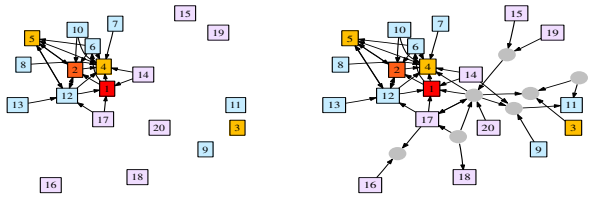
- How do query search results project onto the underlying web graph?
- What can we say about the quality of search results, given the properties of their projection on the web graph?
- Can we predict the difficulty of the query given the projection graph?
- Can we predict users' behaviors with issuing and reformulating queries given the query projection graph?
- How do query reformulations reflect on the projection graphs?

The rest of the paper is organized as follows. In Section 2, we introduce web projections and explain the methodology and the attributes used to model query projection graphs. In Section 3, we describe the data used in our studies. We then describe applications of our approach to predict the quality of sets of search results (Section 4), and to model user behavior when reformulating queries (Section 5). In Section 6, we compare our work to prior research. Finally, we summarize and conclude in Section 7.



(a) Projection graph (b) Connection graph

Figure 2: Query projection graph and query connection graph for the top 20 results of the query *Yahoo search engine* projected on the URL graph.



(a) Projection graph (b) Connection graph

Figure 3: Query projection graph and query connection graph for the top 20 results of the query *Subaru* projected on the domain graph. Notice that projection on the domain graph is denser than projection on the URL graph (figure 2).

2. QUERY PROJECTIONS

We begin by describing the main steps with building web projections and then provide formal definitions of the key components. Figure 1 shows the basic steps of applying the method to analyze search results. We start with a query and a set of results for the query, generated by some procedure, typically via the use of a preexisting search engine (a). We project the search results on the web graph (b), by finding the search results (square nodes) in the larger web graph and then inducing a subgraph based on these identified nodes (c). We name the induced subgraph the *query projection graph*.

Given the typical distances among search results, the query projection graph often contains disconnected components. We connect the nodes of query projection graph to create a *query connection graph* (d). The disconnected components of the query projection graph are connected by identifying web pages that join the components via shortest paths. The *connection nodes* that are introduced during the connecting of the projection graph (circular nodes in Fig. 1) are not drawn from the search result set.

Given the query projection graph and query connection graph we generate a set of evidential features describing the topology of the graph for use in the creation of predictive models via machine learning (e). We provide details about sample topological features in Section 2.2. Finally, we build a case library from a consideration of the topological properties for multiple queries for different outcomes (*e.g.*, high-quality versus low-quality sets of results), and use the case library of graph-theoretic relationships and outcomes to train models that can make predictions about the outcomes. We shall focus in this paper on the tasks that harness graphical properties of web projections generated from sets of results. Two such tasks are the construction of statistical models for predicting quality of a set of search results and modeling user behavior in reformulating queries. As we shall discuss later, there are also opportunities to use the web projection approach to assist with the ranking of individual search results. In such applications, properties and relationships of single results to the subset of pages in the query projection are of interest.

Figure 2 shows an example of a query projection graph and query connection graph for the query *Yahoo search engine*. Square nodes represent web pages and directed edges represent hyperlinks. Circular nodes represent connection nodes. The number in each square represents the rank of the search result in the list returned by a search engine. The color (monochromatic shade) of the node indicates a human-evaluated relevance score of the result to the query. The most relevant results are colored dark (red), the next most relevant orange, then yellow, green, blue and purple.

Figure 3 shows the projection of results for the query *Subaru* onto the domain graph rather than the URL graph. For both projections, the most relevant nodes (colored dark) appear in central locations in the graph and are pointed at by other search results. In contrast, the least relevant nodes (colored bright) are usually not as well connected, often requiring connection nodes to join them to the subgraph.

2.1 Query projection and connection graphs

We now present formal definitions of query projection graph and query connection graph. Consider a *directed* web graph $G(N, E)$ with node set N and directed edge set E, and a given set of search results S. First, we project the results on the web graph G to obtain a set of *projection nodes* N_p, where $N_p = S \cap N$. Note that, ideally, we would like $N_p = S$ but since the coverage of the web graph may not be complete, some search results may not be found in the graph. Thus, in general, $N_p \subseteq S$. We define:

- **Query projection graph** is a subgraph $G_p(N_p, E_p)$ of G induced on N_p nodes, *i.e.*, edge set $E_p = \{(u, v) \in E; u \in N_p \wedge v \in N_p\}$
- **Query connection graph** is a subgraph $G_c(N_c, E_c)$ of G induced on N_c nodes, where $N_c = N_p \cup C$, i.e. edge set $E_c = \{(u, v) \in E; u \in N_c \wedge v \in N_c\}$. Set C is a set of connection nodes, *i.e.*, minimal set of nodes that makes graph G_p connected.

Note that finding the minimal set of connection nodes C is NP-hard, since the problem of finding a Steiner tree [11] reduces to this problem. In our experiments, we used a heuristic policy to find the set C, *i.e.*, to connect the components of G_p. We found that the heuristic policy was reliable and performed well on the datasets that we considered. The policy is as follows:

Let D_i denote the node sets of connected components of G_p ordered by decreasing size ($|D_i| \geq |D_{i+1}|$). We connect each component via the shortest path to the largest component and continue until all components are connected. More precisely, we start with D_2 and connect it via a shortest path on nodes C_2 to D_1. This creates a new largest component $D_{12} = D_1 \cup C_2 \cup D_2$. Now, we proceed and connect D_3 to D_{12} via shortest path C_3, creating $D_{123} = D_{12} \cup C_3 \cup D_3$, and so on until all components are connected. A set of *connection nodes* C is then $C = \cup C_i$. We define a shortest path between the sets of nodes U and V as the *shortest undirected path* over all pairs of nodes (u, v), $u \in U, v \in V$.

GF-PROJ: query projection graph (G_p) features (12)	
G_pNodes	number of nodes in G_p
G_pEdges	number of edges in G_p
G_pComponents	number of connected components
G_pGccNodes	nodes in largest component
G_pGccEdges	edges in largest component
G_pMxDeg	maximal node degree
G_pDeg0Nodes	number of isolated nodes
G_pDeg1Nodes	number of degree 1 nodes
G_pTriads	number of triangles in G_p
G_pDensity	density of G_p ($\|E_p\|/(\|N_p\|(\|N_p\|-1))$)
G_pGccSize	size of largest component ($\|D_1\|/\|N_p\|$)
G_pClustering	clustering coefficient of G_p
GF-CONN: query connection graph (G_c) features (16)	
G_cNodes	number of nodes in G_c
G_cEdges	number of edges G_c
G_cCNodes	number of connector nodes C
G_cCEdges	number of edges incident to C
G_cMxCnDeg	maximal connector node C degree
G_cMxCnOutDeg	maximal connector node C out-degree
G_cMxPnDeg	max projection node (N_p) degree in G_c
G_cAvgPnPath	mean path length of N_p nodes in G_c
G_cMxPnPath	max path length of N_p nodes in G_c
G_cAvgPath	mean path length of N_c nodes in G_c
G_cMxPath	max path length of N_c nodes in G_c
G_cTriads	number of triangles in G_c
G_cDensity	density of G_c ($\|E_c\|/(\|N_c\|(\|N_c\|-1))$)
G_cClustering	clustering coefficient of G_c
GF-COMB: Combined features (17 features)	
DomsToUrls	Ratio of domains to urls in result set
Coverage	Coverage of the projection (N_p/S)
G_pG_cNodes	Node ratio ($\|N_p\|/\|N_c\|$)
G_pG_cEdges	Edge ratio ($\|E_p\|/\|E_c\|$)
G_pG_cAvgPath	Path ratio (AvgPnPath/G_cAvgPath)
G_pG_cMxPath	Path ratio (MxPnPath/G_cMxPath)
F-QUERY: query features (10 features)	
QueryChLen	number of characters in the query
QueryWrdLen	number of query words
QuerySrcRes	number of search results
QueryNDoms	number of domains in result set
QueryNUrl	number of URLs in result set
QueryNRated	number of results with human rating

Table 1: Sample features used to represent query projection, and connection graphs, and the query.

2.2 From graph to evidential features

Given query projection and connection graphs, we seek to extract a set of features that captures key properties of the topology of the graphs. In all, we considered 55 features to describe the characteristics of the projection and connection graphs, and of the query.

Table 1 presents representative features drawn from the larger set of attributes that we used in our experiments. The majority of the features are graphical properties, including the number of nodes and edges, the number and size of connected subgraphs, etc. See [26] for a review of basic graph-theoretic concepts and detailed definitions of the features we use in this work. In one set of experiments, we also considered non-topological properties derived solely from the text of the query and results.

We group the evidential features into four classes: *Query projection graph features* (GF-PROJ, 12 features) are calculated from the projection graph. These features measure various aspects of the connectivity of G_p. Similarly, *query connection graph features* (GF-CONN, 16 features) are obtained from G_c and aim to capture the relations between the projection nodes N_p in the context of connection nodes C. We also consider *combination features* (GF-COMB, 17 features), defined as compositions of features from the other groups. They largely include various ratios and normalizations of more atomic features contained in the other categories. Last, *Query features* (F-QUERY, 10 features) represent non-graphical properties of the result set, calculated from the text of the query and a list of returned search results, including the number of results and domains in the result set.

3. GENERATING CASE LIBRARIES

We now present details on constructing libraries of cases consisting of sets of topological properties that characterize the projections of queries onto the web graph. We used two different representations of the web as a graph. For the study of relevance, we constructed projections from nearly 30 thousand queries, each with a corresponding set of search results. Most of the search results were labeled with a human-assigned relevancy score. For our study of query reformulations, we employed a set of 42 million query-to-query transitions with corresponding lists of search results generated at each step in the search session. For all of the experiments, the search results were obtained from a state-of-the-art ranking algorithm which considers a large number of content features as well as some topological properties on the web as a whole.

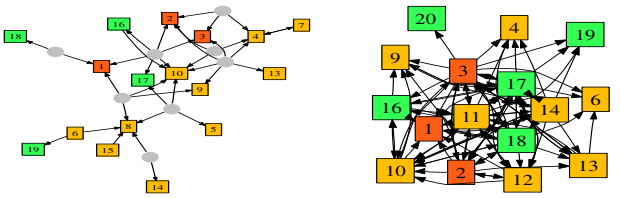
3.1 Web as a graph

We now present the web graphs that provided the substrate for the query-focused projections. We use two variants of the web graph: a URL graph and a domain graph.

3.1.1 URL graph

URL graphs are the most commonly used representation of the web as a directed graph. Nodes represent web pages, and there is a directed edge from node u to node v if there is a hyperlink from web pages u to web pages v. We created our web graph based on a sample of 22 million web pages from a crawl of the web performed in March 2006. We used a sample of the web considered to be of high quality. We started crawling from a seed set of popular, high quality web pages with good reputation. The graph contains 345 million edges and is well connected; the largest weakly connected component contains 21 million nodes, while the second largest has less than a thousand nodes. The strongly connected component is also large, containing 14 million nodes. The second largest component has 7000 nodes. The graph has diameter of 8, and node degrees follow a power-law distribution.

For some prediction tasks, we focused on subsets of these URLs, *e.g.*, those for which we have human relevance judgments. When we project the URLs tagged with relevance judgments onto this URL graph, results may be missing. URLs may be absent for several reasons, including the limited nature of the sample of URLs that we worked with, changes in pages that are returned and judged over time, and the volatile nature of dynamically generated pages. For some tasks, (*e.g.*, predicting the top 20 versus bottom 20 results set, described in detail in Section 4.3), the difference in coverage alone can be a good predictor of class. As we wanted to focus more on the graphical properties than on



(a) URL graph (b) Domain graph

Figure 4: Query projection graph for the top 20 results of the query *encyclopedia* projected on the URL and domain graphs.

coverage per se, we normalized the number of projected results in the graph for the different classes. We did this by first noting how many URLs for the top 20 results were in the projection graph, then considering as many results as needed from the bottom to get the same coverage.

3.1.2 Domain graph

In the domain graph, nodes represent domain names, (*e.g*, `cmu.edu` or `microsoft.com`), and there is a directed edge from node u to node v, if there is are web pages inside domain u that contain a hyperlink to web pages inside domain v. It is important to note that nodes in a domain graph are not arbitrary domains; all sub-domains are collapsed into a second-level domain name. For example, web pages from domains `cs.cmu.edu`, `ml.cmu.edu`, and `lti.cs.cmu.edu` are merged into a single node (domain name) `cmu.edu`.

We considered a complete domain graph of the web from February 2006. The graph contains 39 million domain names and 720 million directed edges. The graph is densely connected, has a diameter of 4, and the largest component contains 99.9% of the nodes. Since this is a complete domain graph we have no problems with projection coverage. The domain of every search result in our dataset can be found in this graph.

Figure 4 shows the differences between projections on URL and the domain graph when projecting the top 20 results for the query *encyclopedia*. Domain graph projections are usually denser and much better connected than the URL graph projections. Domain graphs also have better coverage of the search results, with very few missing nodes.

3.2 Human-rated search results

In one set of experiments, we explored the use of the web-projection methodology for the task of predicting the quality of a set of search results. This task requires assessments of result quality, which we obtained from human judges. For each query, the top k results from one or more systems were presented to the judges for evaluation. The quality of a query-result pair was explicitly labeled by the judges using a six point scale ranging from "Perfect" to "Bad". We note that the labeling was performed over the results already highly ranked by a web search engine, and thus corresponds to a typical user experience. Out of 30,000 total available queries, we focused our experiments on a set of 13,000 queries with at least 40 rated results, with averages of 57 results (URLs) and 46 domains per query.

3.3 Query reformulation corpus

In a second set of experiments, we used web projections to explore patterns of query reformulation. We examined a sample of query logs captured over a six week period by a popular web search engine. We obtained query-query transitions from the logs as described in [23]. For each query q_i, we measured n_i, the number of times the query was observed. For a pair of queries (q_i, q_j), we also measured the probability of reformulation or transition from query i to j, p_{ij}. If we let n_{ij} be the number of times that q_i was followed by q_j within a thirty-minute window, then $p_{ij} = n_{ij}/n_i$ is the probability of q_i being followed by q_j. And similarly, probability p_i of query q_i participating in a transition is defined as $p_i = \sum_j n_{ij}/n_i$.

We started with a set of 35 million queries and 80 million query transitions as defined above. For the analysis described below, we considered only queries and reformulations that appeared at least 10 times in our corpus. Our analyses used 48,458 queries and 120,914 query transitions. We then used the top 20 search results for each of the 48 thousand queries and projected them on the URL and the domain graphs.

4. QUALITY OF SEARCH RESULTS

For predicting the quality of search results, we asked the following questions: By analyzing the projection of a query onto the web graph, what can we tell about the quality of the returned result set? What can we tell about the difficulty of the query? More specifically, we explored the following two tasks:

1. Discriminate good (top 20) versus poor (bottom 20) search result sets.

2. Given a set of results, predict how good the set is, *i.e.*, predict the highest human relevancy rating in the set.

We now describe the problem setting and experimental setup as well as the baseline method.

4.1 Problem definition

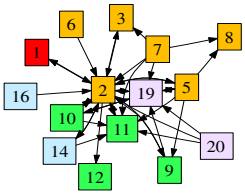
We focus on the following general setting: We are given a query q_i with a set of search results S_i. Each query q_i belongs to class y_i. A class is a categorical value that can be, as an example, the rating of the most relevant search result in the set, or an indicator of whether the result set S_i is composed of the top-ranked or bottom-ranked search results.

We start with S_i and project it on both the URL and the domain graphs (see Section 3.1), create both projection and connection graphs, and extract the attributes as described in Section 2.2. This means that we project every query q_i onto two different graphs of the web, and for each projection, we extract a set of features as defined in table 1. We generate a case library of training examples q_i described with features and we learn a model to predict class y_i via a machine learning procedure.

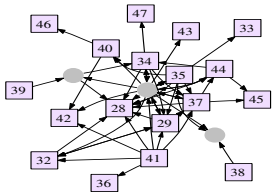
4.2 Experimental setup

For learning the models, we used the WinMine toolkit [3] that uses the GES algorithm [4] in Bayesian structure search to learn a Bayesian network. We model the continuous features as Gaussians and discrete features with a multinomial distribution. For all experiments we report the average classification accuracy over a 10-fold cross validation.

We compare the predictive power of the learned models with two baseline methods. The first baseline model is the marginal model, which predicts the most common class. The

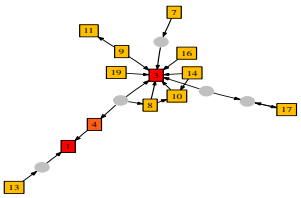


(a) Good result set

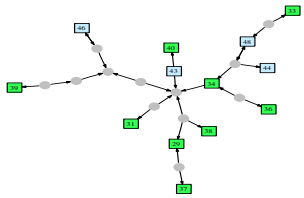


(b) Poor result set

Figure 5: Domain graph projections of good and poor result sets for query *medline*.



(a) Good result set



(b) Poor result set

Figure 6: Projections of good and poor result sets for query *Wisconsin* projected on the URL graph.

second baseline algorithm we use is based on a ranking algorithm that uses a large number of textual and global graph features to rank search results. For the classification tasks, we learn a threshold on the score to predict the class.

The baseline ranking algorithm we used is RankNet [1], a supervised machine-learning technique developed to learn a ranking function. The learning methodology is a neural net algorithm that optimizes feature weights to best match explicitly provided pairwise user preferences. Over 350 input features are used to train RankNet. These features include various aspects of document content, anchor text features, and basic hyperlink features. The output of RankNet is used to rank results. Since the output of RankNet is a combination of state of the art features for ranking, we use it as a discriminatory feature that serves as input to the Bayesian-network classifier. We think this serves as a strong baseline.

4.3 Relative quality of result sets

The first task we consider is the classification of good (top 20) versus poor (bottom 20) result sets. For this task, we used the explicit relevance judgments described in 3.2. For each query, we order the search results from best to worst using the human judgments. We note that this ordering can be different than the output of the search engine.

We then project the top 20 search results, and bottom 20 results ordered by human judgments onto the URL and domain graphs, and compute the features described in table 1 for the two graphs. We learn a model that can predict, for a previously unseen query with a set of search results, whether it is good (top 20) or poor (bottom 20). Given the average number of judged search results per query, we are effectively learning to discriminate the top 20 results versus the search results with ranks 40 to 60. Note that this task involves predicting the relative quality of results for a given query. We examine predicting the absolute quality of the results in the next section.

First, we point to Figure 5 which displays examples of projections of good and poor result sets for the query *medline* on the domain graph. We can see that results in a good set are tightly clustered, while those in the poor result set are more spread out, requiring many connection nodes to connect the components of the projection graph. Also notice that the results marked by humans as most relevant (darkest nodes) appear as the central (high-degree) nodes in the graph (Figure 5(a)). Similarly, Figure 6 shows the good and poor result sets for the query *Wisconsin* projected on the URL graph. The central node in the good result set (panel a) is one of the search results, whereas in poor set (panel b) the central node is a derived connector node.

Table 2 shows the results for the task of predicting good versus poor result sets using several different methods and

Feature set	URL graph	Domain graph
Baseline–Marginals	0.50	0.50
Baseline–RankNet	0.74	0.74
GF-PROJ	0.62	0.82
GF-CONN	0.60	0.86
GF-PROJ+GF-CONN	0.87	0.90
GF-ALL	0.88	0.88

Table 2: Classification accuracy for predicting good versus poor result sets.

feature sets. Results are shown separately for URL and domain graph projections.

The Baseline–Marginals row displays the classification accuracy of predicting the most common class. Baseline–RankNet is the second baseline where only the RankNet score is used for learning. We are using a combination of about 350 textual features to discriminate the good and the poor result sets. GF-PROJ uses the 12 features extracted from the projection graph, GF-CONN uses the 16 connection graph features, GF-PROJ+GF-CONN uses both of these feature sets (28 features), and GF-ALL refers the case where all 55 features, most of which are described in table 1, are used for learning.

We were not surprised to find that RankNet and the new graphical features outperformed the marginal baseline. The RankNet output reflects the extent to which human judgments agree with the output of the learned ranking function. The "GF-" results reflect the extent to which graphical features of the results subset are predictive of human relevance judgments. For the URL graph, the RankNet baseline outperforms models trained only on projection or connection graph features, but the models trained on both sets of features shows substantial improvement over RankNet (18% relative improvement). For the domain graph, all models trained on graphical features outperform RankNet. We obtained the best classification accuracy of 90% when combining projection and connection graph features. We note that we obtained higher classification accuracies when projecting on the domain graph than for URL projections. This is likely due to the sparser coverage of the URL graph.

We found interesting the performance of the "GF-" models, considering only topological features of the web projections, and bypassing analysis of content matches between the query and web pages.

Figure 7 shows a simple model learned from the query projections on the domain graph using the GF-PROJ feature set. The model has a classification accuracy of 0.82. The figure shows the decision tree for the output variable Top 20 vs. Bottom 20. Nodes correspond to input fea-

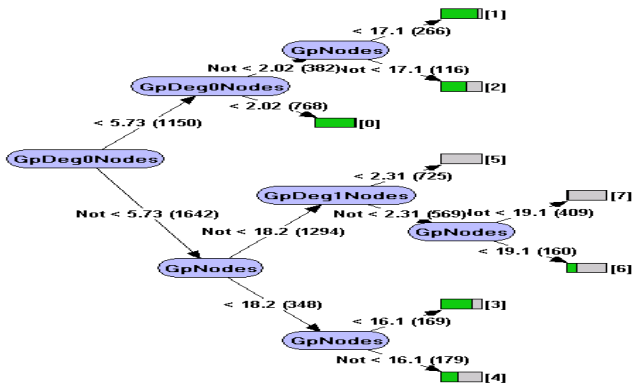


Figure 7: Learned model for discriminating good versus poor search result sets based on query projection on the domain graph.

Feature set	URL graph	Domain graph
Baseline–Marginals	0.36	0.36
Baseline–RankNet	0.48	0.44
GF-PROJ	0.51	0.53
GF-CONN	0.50	0.52
GF-PROJ+GF-CONN	0.54	0.54
GF-ALL	0.55	0.55

Table 3: Result set quality classification accuracy for a 6-way classification problem.

tures, and each leaf node shows the probability distribution for the output variable, which is shown as a histogram. In this case, the variable has only two possible values; the green (darker) area indicates the proportion of good (top 20) sets and the grey area poor (bottom 20) sets. Labels on the edges show the splitting criteria of the parent node variable, and the numbers in parenthesis show the number of training examples routed over the edge. The projection graphs of good result sets (shown as large green (dark) area of the histograms) have few isolated nodes (low values of G_pDeg0Nodes) and results are coming from a few domains (low values of G_pNodes). On the other hand, poor result sets have many domains and many isolated nodes.

4.4 Absolute quality of a result set

In the previous section, we considered the problem of discriminating between good and poor result sets. Now we focus only on top-rated (top 20) results for each query and aim to predict the absolute quality of a query result set. More specifically, we label each query with the highest human-assigned rating for any result in the set. We note that we could use other measures to summarize the quality of result sets. The highest human rating is easy to describe and is of practical importance. Since the human relevance judgments were on a 6-point scale (Section 3.2), we can examine the problem at several granularities. Here, we present results for the 6-class problem (predict top label exactly) and the 2-class problem (predict whether the top label is from the three highest or three lowest rating categories).

First, we consider the 6-class task of predicting the exact rating of the highest rated document in the set of top 20 results. Table 3 shows the classification results. For the URL graph, we obtained a 15% relative improvement over using the RankNet to predict the quality when using all attributes. For the domain graph the improvement was even larger, 25%. Note that all methods using any combination of graphical attributes outperform both baseline methods.

Feature set	URL graph	Domain graph
Baseline–Marginals	0.55	0.55
Baseline–RankNet	0.63	0.60
GF-PROJ	0.80	0.64
GF-CONN	0.79	0.66
GF-PROJ+GF-CONN	0.82	0.69
GF-ALL	0.83	0.71

Table 4: Result set quality classification accuracy for a binary classification problem.

The model (not displayed per space limitations) for the 6-level result set quality classification problem is more complex. The first split of the induced decision tree is on the node ratio of the projection and connection graphs. If the connection graph is much larger than the projection graph, the results are likely to be of poor quality. Moving down the tree, we see that if maximum degree in a graph is relatively small, the results are likely to be of medium quality, with results getting worse as the number of domains in a top 20 set increases. The model revealed that high quality search result sets are associated with projection nodes with large degrees, few domains, small domains to URL ratios, and are well connected.

Next, we examine the same problem at a coarser granularity. The task is to predict whether the set contains a result with the rating in the top or the bottom half of the 6 point rating scale. Table 4 shows the classification accuracies for the classification problem. We note that the difference in performance between the domain and URL graph projections increased even further and that the relative increase in performance over the RankNet baseline increased (31% for the URL and 18% for the domain graph).

This task is similar to that of discriminating the good versus poor result sets (as described in Section 4.3). However, it is also more difficult since we are only working with top 20 results for each query and predicting the absolute quality of the set. The good versus poor prediction requires only a relative judgment.

For the task of distinguishing good versus poor result sets, we found that projections on the domain graph outperformed the projections on the URL graph. For the case of predicting the exact quality of a result set, the projections on the URL graph generally performed better even in cases where the URL graph has the problems with coverage. This may be explained by the difference in the goals and representation. For good versus poor discriminations, the quality of the whole set is important and the domain graph likely represents an appropriate level of abstraction for handling this challenge. In contrast, the quality of a result set is a single result (single node) property. Here the domain graph may be too coarse to capture fine-grained properties of the high quality nodes (search results). A projection on the URL graph may be needed to capture necessary properties.

5. QUERY REFORMULATIONS

As a second illustration of the use of web projections, we explore the learning of models to predict users' query-

Feature set	URL graph	Domain graph
Baseline–Marginals	0.54	0.56
GF-PROJ	0.59	0.58
GF-CONN	0.63	0.59
GF-PROJ+GF-CONN	0.63	0.60
GF-ALL	0.71	0.67

Table 5: Classification accuracy of predicting whether the query is likely to be reformulated.

reformulation behavior and characteristics. Web searchers often refine their queries one or more times, as they seek information on the web. Prior research has explored query reformulations, considering such issues as the timing and type of reformulation seen. For example, Lau and Horvitz [16] build models to predict the likelihood that searchers will specialize, generalize, or reformulate queries within a search session, considering the history and timing of actions. Jones *et al.* [10] examine substitutions that searchers make to their queries.

We explore the use of web projections to build models that predict if and how users reformulate their queries. We used a set of 48 thousand queries that were reformulated at least 10 times. For every query, we took the top 20 search results returned by the search engine, and created the query projection and connection graphs, extracted the graph features, and trained predictive models.

More specifically, we consider the following tasks:

1. Distinguish queries with high versus low reformulation probability.
2. Given a transition from query q_s to query q_d, predict whether it is a specialization or generalization.
3. Given a query that is likely to be reformulated, predict whether it is going to be generalized or specialized.

Next, we describe the experimental setup and give more detailed description of our results and findings.

5.1 Experimental setup

Using the query reformulation data described in Section 3.3, we defined several binary classification tasks. For each selected query, we took the top 20 search results as returned by the search engine, projected them on the domain and URL graphs, and extracted the features. For some of the tasks, the training datasets were quite imbalanced where one outcome was significantly more likely than the other. In order to focus on the key discriminations rather than basic marginal frequencies, we sub-sampled the majority class, so that both classes had roughly the same number of training examples.

5.2 Probability of query reformulation

First, we considered the problem of learning whether a query is likely to be reformulated or not. We split our set of queries into two classes: queries with high reformulation probability ($p_i \geq 0.6$) and queries with low reformulation probability ($p_i \leq 0.15$). We selected these values so that the two classes were about the same size.

Table 5 shows the classification accuracies when projecting on URL and domain graphs. We found gradual improvement with increasing the numbers of topological features under consideration. We also found that cases drawn from projections on the URL graph provided better performance than cases generated from projections on the domain graph. We note the baselines for predicting the most common class are slightly different between the URL and the domain graph since we discarded a few queries that produced very small URL projection graphs.

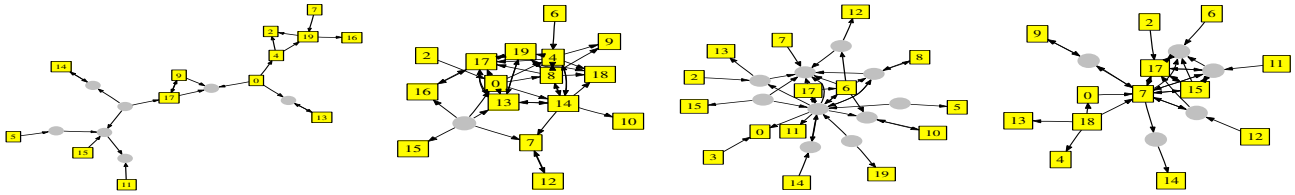
Examining the model (not shown for brevity) we see that, queries that are likely to get reformulated come from many domains, are generally longer than queries that are not reformulated, and have relatively high degree (> 4) connection nodes. Such findings again suggest that queries whose results are tightly knit together on the web are of higher quality, given that such queries are less likely to be reformulated. The findings also suggest result sets with central high degree nodes and a small number of connector nodes are of higher quality.

In another set of experiments, we explored transitions between queries. For this task, we took pairs of queries where there is a strong tendency of transition in only one direction, and then trained a model that learns whether a given query is likely to be the transition *source* or *destination*. Figure 8 shows two examples of source and destination graphs. Our models were able to predict whether a given query is a source or a destination of the transition with an 85% classification accuracy. The learned model provided insights about the relationship between topological properties and the likelihood of the direction of a transition. We saw that sources of query transitions tend to have some isolated nodes, short query strings, many connected components, and nodes that lie far apart in the connection graph, which indicates the returned search results are not satisfactory. In contrast, reformulation destinations (especially specializations) tend to be better connected, and to have higher in-degrees of projection nodes. Intuitively, these results make sense: a searcher probably wants to specify a new query if the search results are somewhat "random", *i.e.*, are scattered widely around on the web. The results of this experiment led another question, which we explore in the following section.

5.3 Query specialization versus generalization

We have just described how we can reliably learn whether a given query is the source or destination of a reformulation. Now, we pursue models that can predict the nature of reformulations. We shall explore in particular whether a reformulation is likely to be a specialization or a generalization of the source query. Given a pair of queries, where q_s is often reformulated into q_d, we want to learn characteristics of projection graphs for queries that are specialized versus generalized.

For this task, we define query specialization as the addition of more words to an existing query, and similarly define generalization as removing words from the query (richer characterizations have been considered Lau and Horvitz [16] and by Jones *et al.* [10].) Given the query transition data, we extracted all pairs of queries where a specialization or generalization transition had occurred at least 10 times. Then we separately projected the source q_s and the destination query q_d and extracted the features. We created transition features by simply taking the difference of the corresponding feature values: $F_i(q_s) - F_i(q_d)$, where $F_i()$ denotes i^{th} feature. Note that in this experiment we do not use the query text attributes (length of the query string) as it would be possible to directly identify the type of transition by the change in the length of the query.



(a) Source of the generalization (b) Destination of the generalization (c) Source of the specialization (d) Destination of the specialization

Figure 8: Sources and destinations of query transitions. Projections (a) and (b) show an example of a generalization from query *free house plans* to the query *house plans*. Projections (c) and (d) show the specialization from *strawberry shortcake* to *strawberry shortcake pictures*. Notice how the reformulated queries result in more connected graphs and bring result nodes into the center.

Feature set	URL graph	Domain graph
Baseline–Marginals	0.50	0.50
GF-PROJ	0.71	0.84
GF-CONN	0.69	0.83
GF-PROJ+GF-CONN	0.71	0.85
GF-ALL	0.80	0.87

Table 6: Classification accuracy of predicting whether a given query transition is a specialization or a generalization.

We show the classification performance in Table 6. Here, using only the projection graph features performs slightly better than using solely the connection graph features. We see consistent increases in performance by combining the projection graph and derived features. We obtain the best accuracy of 87% using projections on the domain graph and all features for learning the predictive models.

The learned decision tree for predicting reformulation using GF-PROJ+GF-CONN features with projections on the URL graph is displayed in Figure 9. Note that the splitting criteria (*e.g.*, GpComponents) here are not the values of the attributes but rather the changes in attribute values, *i.e.*, the difference in the attribute values of the source and the destination of the transition. The model shows that query specializations are characterized by the decrease in the number of connected components of projection graph (first split of the tree). It also shows that the number of nodes and edges in the projection graph increases for generalizations. For specializations, we see that the number of isolated nodes decreases, results are gathered in few connected components, and the size of largest connected component is increases, while the number of connector nodes decreases. These results correspond with the intuition that, when a query is generalized, the list of results will get richer and more diverse. The findings revealed in the learned models suggest that the projection graphs associated with generalizations are sparser and less connected than those associated with specializations, where the projection is likely to be more concentrated, denser, requiring fewer connector nodes.

It may seem that the results here do not go along with those in section 5.2, where we find characteristics of query projection graphs that lead to query reformulation, *i.e.* learn characteristics of badly formulated queries and transitions as query is formulated. On contrary, we see here that in general specializations narrow down the search, while generalizations tend to lead to higher diversity and larger coverage.

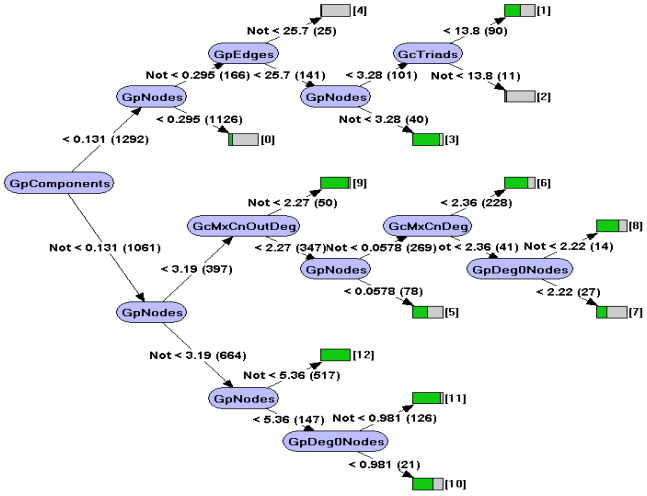


Figure 9: Model learned on URL graph projections for predicting whether transitions between queries are generalizations or specializations.

5.4 Predicting type of query reformulation

Finally, we examine the type of reformulation associated with a query. We seek to predict whether it is more likely to see specific queries generalized or specialized, and how this reflects on the properties of the query projections. For this task, we learn models that consider specific properties of queries that are reformulated in a certain way. Again, we do not use the features derived from the query string (length of the query, number of words, etc.) as the change in the length of the query provides information about the type of reformulation.

Table 7 gives the classification performance for these models. We found that the performance of models learned from cases generated from the URL and domain graph projections is about the same. The projection graphs provided models with better performance than those using only the features of connection graph. Using all features, we obtained a classification performance of 78%.

The learned probabilistic decision tree model shows that the most discriminatory property for this task is the maximum degree of a node in the projection graph. If the maximum degree is low, the query is likely to be specialized. If there is no central node in the projection graph, the user

Feature set	URL graph	Domain graph
Baseline–Marginals	0.50	0.50
GF-PROJ	0.71	0.68
GF-CONN	0.62	0.65
GF-PROJ+GF-CONN	0.70	0.68
GF-ALL	0.78	0.76

Table 7: Classification accuracy of predicting whether a reformulation will likely lead to a specialization or generalization.

will likely specialize the query. On the other hand, generalizations occur when the largest connected component of projection graph is large (more than 10 nodes, for the top 20 results) and where nodes are close together (low average path length in connector graph).

6. RELATED WORK

In prior research, investigators have explored the use of query terms to identify properties and relationships among specific parts of the web graph. In the HITS work by Kleinberg [13], eigenvectors are used to identify authoritative nodes using the notion of focused subgraphs defined by a query and associated links, and mutually reinforcing hubs and authorities. The work is similar to ours in that it extracts and operates on a query-defined subset of the web graph. In contrast to methods we have presented, HITS calculates a single property of a node (the corresponding component of the 1st singular vector of graph adjacency matrix), which is used to rank search results. We use a much wider range of graphical features that characterize whole subgraphs.

Variants of PageRank that work with subgraphs of the web have also been explored in prior research. This work includes explorations of domain-specific or person-specific PageRank [8, 24], and on the use of non-random jump vectors for personalization [9]. Related work on identifying web spam has examined methods for propagating from trusted-pages [27]. Recent work by Nie *et al.* [19] focused on eigenvectors or counts to set node priors.

In the content domain, Cronen-Townsend et al. [5] have looked at techniques for predicting the quality of results (what they call *query difficulty*) by computing the entropy between the language model for the results and the collection as a whole, but they do not consider any graphical properties. Several efforts have combined links and content in different ways. For example, Charkrabarti et al. [2] use link information on classes of neighbors to improve text classification accuracy of a target page. Dean and Henzinger [6] use links and content to find related pages, making use of information about the simple existence of links, rather than the rich topological characteristics of subgraphs that we represent and exploit. There has been research on considering multiple objectives, for example examining relationships among papers, authors, and institutions. In this work, relationships have been computed globally, based on one or more sets of similarity measures [28, 20].

Related work also includes research on citation analysis, including Garfield's early work on the impact factor [7], and later refinements by Pinski and Narin [22]. Vassilvitskii and Brill have recently explored the use of distance (and direction) in the web graph for relevance feedback in web search [25]. Minkov *et al.* [18] have examined contextual search and name disambiguation in email messages using graphs, employing random walks on graphs to disambiguate names. In the context of machine learning on graphs, researchers have sought to predict the labels of vertices in a graph, given the known labels of vertices in training data [14, 17]. A similar formulation has recently been explored in the context of kernel methods [12], and approaches based on extracting subgraphs as features [15].

In contrast to previous efforts, we examine a broad set of graph-theoretic properties of subgraphs, rather than, for example, only examining a single feature such as the eigenvalue associated with individual nodes. The richer characterization of the topological properties of subgraphs as a whole—or of individual nodes relative to the subgraph—allows us to investigate the discriminability of multiple features, to learn models for diverse classification goals, and, more generally, to provide useful analytical tools for exploring the web.

7. SUMMARY AND DIRECTIONS

We presented a methodology for learning predictive models and attributes from a rich set of topological characteristics of sets of web pages, based on their projection on the larger web graph. First, we considered patterns of connectivity among the set of pages returned as search results by projecting them onto the web graph, and analyzed the topological properties of the induced subgraphs. Using these graph-theoretic features, we performed machine learning to build classifiers that discriminate good and poor sets of results. Then, we learned models that predict user behavior when reformulating queries, including whether queries are likely to be reformulated, and the nature of the reformulation. The experimental results for the two problem domains highlight the potential value of employing contextual subgraphs for understanding search-related tasks.

The web-projection method is scalable as demonstrated by the sizes of datasets used in our analyses. Calculating the graph features is fast since projected graphs are fairly small. The most computationally expensive operation is obtaining the query connection graph. Here a shortest-path algorithm is performed to connect the components of query projection graph. If the components are far apart and the graph is densely linked, the shortest-path algorithm will have to traverse most of the graph. In rare cases, this problem arises in the domain graph and, the algorithm can take up a minute to complete. On average, however, less than three seconds were required to project a query and produce the projection and connection graphs. In case of URL graph, which is not as densely connected, we did not see visible delays in the production of the graph projections.

The method of projecting sets of results on the underlying graph of the web and then using machine learning with graph-theoretic features can be applied in many settings. For example, prior work has noted that spam web pages have distinct linkage patterns. We can use the web-projection method to identify either sets of results that are likely to contain many spam pages, or more specifically to identify pages that are likely to be spam (using graphical features of individual nodes as we will discuss next).

Applying our ideas to enhance ranking is especially interesting as it involves looking at features of individual nodes relative to a subset, which is interesting and has wider applicability than ranking itself. There are several ways one could approach this opportunity. One of the approaches we have been pursuing considers the inclusion of additional node-

centric features from the projection and connection graphs, including those describing the position of the node with regard to the rest of the graph. These node-centric features can then be used to train existing ranking algorithms (*e.g.*, RankNet). Another application of our work is to discover missing human relevance scores. Given a projection graph where we have human relevance judgments for a few nodes, we would like to predict the judgments that would be assigned to the rest of the nodes.

Another promising direction for research is exploring the role of the connector nodes. We used these nodes to connect disconnected components of the projection graph. However, we observed that the connector nodes often become hubs holding the network together. As an example, see Figure 5(b) where the central connector node is a hub. However, there are also cases where no such patterns emerge, (*e.g.*, Figure 6(b)). To explore these questions, a better understanding of the quality of our heuristics to connect components of projection graph is needed. In the analyses described, we used a greedy heuristic that randomly chooses connections from the set of competing paths of the same length. We do not yet understand the sensitivity this heuristic in the overall procedure for selecting nodes.

With regard to modeling users and their queries, we are excited about exploring clusters of queries and query transitions based on graphical properties of their projection and connection graphs. The rich evidential patterns provided by graph-theoretic features promise to be valuable in describing different characteristics of queries, determining the classes of queries, and mapping query transitions to these classes.

One could also apply the ideas for modeling web searchers' behaviors in other ways. Web projections might be used to construct predictive models that infer paths that the searchers will likely take when reformulating a query from an initial query. Predictive models could be used to suggest likely query reformulations, specializations, generalizations and avoid transitions that would not likely lead to good sets of results, as captured by the graphical properties of projections associated with the sets.

Other applications include using the graph projections to explore the dynamics and evolution of the web, where models learned from features capturing topology and topological dynamics could help us to understand how sites and pages that created or removed over time relate to the rest of the web. Such models promise to be valuable in predicting the conceptual links and quality of new content and sites.

We believe that the represented work captures a starting point with the use of web projections. We found that the methods complement existing textual and graphical analyses and provoke tantalizing questions and interesting directions for future research in web search and retrieval, including efforts on enhancing result quality and on better understanding and supporting human behavior. We hope that the ideas will be of value to others pursuing insights about the nature and use of graphical properties of the web.

8. REFERENCES

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05*, 2005.

[2] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. In *WWW '99*, 1999.

[3] D. M. Chickering. The winmine toolkit. Technical Report MSR-TR-2002-103, Microsoft, 2002.

[4] D. M. Chickering. Optimal structure identification with greedy search. *JMLR*, 3:507–554, 2003.

[5] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR '02*, 2002.

[6] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. In *WWW '99*, 1999.

[7] E. Garfield. Citation analysis as a tool in journal evaluation. *Science*, 178(60):471–479, November 1972.

[8] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW '02*, 2002.

[9] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03*, 2003.

[10] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW '06*, 2006.

[11] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. 1972.

[12] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *ICML '03*, 2003.

[13] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[14] R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML '02*, 2002.

[15] T. Kudo, E. Maeda, and Y. Matsumoto. An application of boosting to graph classification. In *NIPS '05*. 2005.

[16] T. Lau and E. Horvitz. Patterns of search: analyzing and modeling web query refinement. In *UM '99*, 1999.

[17] J. Leskovec, N. Milic-Frayling, and M. Grobelnik. Impact of linguistic analysis on the semantic graph coverage and learning of document extracts. In *AAAI '05*, 2005.

[18] E. Minkov, W. W. Cohen, and A. Y. Ng. Contextual search and name disambiguation in email using graphs. In *SIGIR '06*, 2006.

[19] L. Nie, B. D. Davison, and X. Qi. Topical link analysis for web search. In *SIGIR '06*, 2006.

[20] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: bringing order to web objects. In *WWW '05*, 2005.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Dig. Lib. Tech. Proj., 1998.

[22] G. Pinski and F. Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing and Management*, 12:297–312, 1976.

[23] F. Radlinski and S. Dumais. Improving personalized web search using result diversification. In *SIGIR '06*, 2006.

[24] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *NIPS '02*, 2002.

[25] S. Vassilvitskii and E. Brill. Using web-graph distance for relevance feedback in web search. In *SIGIR '06*, 2006.

[26] S. Wasserman, K. Faust, and D. Iacobucci. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[27] B. Wu, V. Goel, and B. D. Davison. Topical trustrank: using topicality to combat web spam. In *WWW '06*, 2006.

[28] W. Xi, B. Zhang, Z. Chen, Y. Lu, S. Yan, W.-Y. Ma, and E. A. Fox. Link fusion: a unified link analysis framework for multi-type interrelated data objects. In *WWW '04*, 2004.