Navigating the Web with Query Tags

Ioannis Antonellis
Computer Science Dept
Stanford University
antonell@cs.stanford.edu

Jawed Karim
Computer Science Dept
Stanford University
jawed@cs.stanford.edu

Hector Garcia-Molina
Computer Science Dept
Stanford University
hector@cs.stanford.edu

ABSTRACT

We propose to integrate various pieces of information about a web page (search queries, social annotations, terms extracted from the pagetext) into a navigational menu. This menu displays an auxiliary set of tags (navigational tags) selected with the goal of helping user navigation. We propose a novel framework (navigational utility) for comparing different tag selection methods. We also investigate which source of tags is more suitable for our scenario and we conclude that tags extracted from search queries (query tags) are more appropriate.

1. INTRODUCTION

Searching and browsing are the two predominant ways to locate information of interest in any hypertext environment such as the Web. However, both paradigms are problematic. When browsing, users must guide themselves using browsing cues: textual or graphical indications of the content reachable via a link. When searching, users lose the important context present in the pages leading to the search result.

The reason both searching and browsing persist is due to their complementary advantages. With searching, users can identify pages containing specific information quickly. With browsing, users can find information even when appropriate search keywords are either nonexistent or unavailable. Also, browsing is appropriate in cases where a great deal of information and context is obtained along the browsing path itself, not just at the final page.

In our work, we treat browsing as a first class citizen of the information discovery process and we focus on enhancing navigation on the Web. We bring search into context by providing a context-aware *navigation menu* to users who navigate the Web. This menu displays *navigational tags*: intelligently selected terms from a variety of sources (e.g., pagetext, search queries, social bookmarking sites, anchor text). These terms are related to what is being displayed in the page and describe potential information needs of a page visitor. In addition, each tag links to a set of destination pages via *navigational links*: automatically created links that are maintained so that the destination page always contains the most up-to-date, relevant to the tag and interesting for the user, content. The ultimate goal of the navigational tags and links is to help users perform search tasks through navigation and efficiently explore the content on the Web.

WWW2010, April 26-30, 2010, Raleigh, North Carolina.

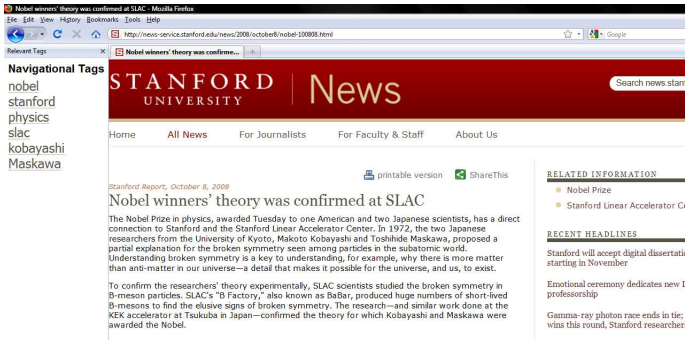

Figure 1: Navigational menu example

We illustrate the idea of navigational menu using a screenshot from the system we are building (Figure 1). The navigational menu is implemented as a browser plugin. In the example, a user is visiting a page from the Stanford news website related to the 2008 physics Nobel Prize. The menu (tab on the left) decides that the tags Nobel, Stanford, physics, SLAC (Stanford Linear Accelerator Center) and the names of the two prize winners can be helpful to users. By clicking, on the tag Nobel, for example, a user will be offered a list of links to pages. Two such pages can be the official Nobel awards website as well as a page from the whitehouse.org website announcing Barack Obama's award. The first page can be matched with the tag Nobel since we expect nobelprize.org to contain occurences of the term Nobel. In addition, we expect a high volume of search queries related to the recent peace Nobel Prize to be associated with the whitehouse.org page. Notice that the Stanford news webpage (along with many other popular websites, e.g. nytimes.com) also contains a section "Related Information" with relevant to the page, but manually selected links. Our navigational menu, however, is (a) designed to be automatically maintained and (b) always present at a specific place, making it easily accesible to users.

Tags are already in use at various sites and have been proven useful. However, users need to go to the various social bookmarking sites (del.icio.us, stumbleupon, etc.) to see tags. The goal of our navigational menu is to (a) gather tags from multiple sources (including query logs) and (b) display them right next to the browsed page. Similarly, web directories and archives (e.g. Google, Yahoo!, DMOZ, etc.) provide links to useful pages for a given "term": a search query in the case of web archives, or a hierarchy description

in the case of web directories. In our navigational menu, we want to show some of these links.

In this paper, we do not attempt to evaluate the utility of our navigational menu, since we first need to know how to populate the menu. Instead, we focus on studying how to build such a menu. We consider three different sources of page tags: (a) tags extracted from the pagetext (pagetext tags), (b) tags from social bookmarking sites (bookmarking tags), and (c) tags extracted from query logs (query tags). We then address two main questions: (i) Which source of page tags is the most appropriate source for navigational tags and (ii) how should we select and rank navigational tags for a given page.

In the following three sections we introduce the navigational utility metric and present the formulation of our navigational tag/link selection problem. We then discuss our three tag and link selection methods (Sec. 5). In Sections 8, 9, and 10 we compare the different tag sources and tag selection methods and present a set of six main findings. We discuss related work in Section 11 and conclude in Section 12.

2. NAVIGATIONAL UTILITY

In this section we introduce the notion of the *navigational utility* of a page as a conceptual metric for understanding and evaluating different navigational tag/link selection methods.

Suppose a user u visits a page p with a known information need (e.g., after submitting a search query q). We expect that there are additional pages, reachable from p that are relevant to q. However, these pages might be many clicks away from p and thus difficult for the user to reach. We can quantify the effort required by the user to navigate to all "important" pages by measuring p's average distance to all of those pages. Since the "important" pages are not known in advance we can instead just compute p's average distance to all pages reachable from p with some number of clicks. We define this quantity as the navigational utility of a page (Section 4).

We can then measure how our navigational menu affects the utility of a page by comparing the utility of the original page (without our menu) and the page's utility with the navigational menu. We can also compare different menu generation methods using the same process. For example, suppose two different link selection methods m_1, m_2 add an extra link to a page p. As a result, the navigational utility of p increases to $u_1(p)$ or $u_2(p)$ respectively. If $u_1(p) < u_2(p)$, then method m_2 will be prefered over m_1.

3. NAVIGATIONAL TAG SOURCES

The three navigational tag sources we consider in our work are: (a) pagetext (pagetext tags), (b) query logs (query tags), and (c) social annotations (del.icio.us tags). Although tags extracted from anchor text (anchor tags) could also be useful, we did not considered them in this paper.

Pagetext. The first source of navigational tags for a page is the actual pagetext. It is the most natural source for navigational tags, and pagetext tags will always reflect the latest page content. For example, if the text in a page changes very often (e.g. if a page has news articles), pagetext tags will always be consistent with the latest content.

Query Logs. Search engine query logs contain information about the search queries people issue before clicking on a web page. In our previous work ([2]), we found that terms used to query for a webpage (query tags) contain extra information compared to the pagetext (and social annotations), and can provide substantially many new tags for a very large fraction of the Web. In addition, as we showed in [2], it is easy for people not afiiliated with a search engine to gain access to search engine logs using the referer field present in every webserver's access log. Finally, query tags provide the users' perspective on what is important in a web page, in contrast to pagetext tags that only capture the point of view of the creator of the page.

Social Annotations. Tags from social bookmarking sites (such as del.icio.us and stumbleupon) provide a user-generated description of a page. Social annotations are an inherently different source of information compared to the pagetext or the query tags since people use these tags to organize their bookmarks in a way that will help them remember and share links to pages.

4. PROBLEM FORMULATION

In this section we present our formal model for selecting navigational tags and links.

Notation: We model a webgraph as a directed graph $G = (\mathcal{V}, \mathcal{E})$ where each node $v \in \mathcal{V}$ corresponds to a webpage and each directed edge $e = (v, u) \in E$ corresponds to a hyperlink from page v to page u. Although we will be referring to G as "the webgraph", it does not mean that G has to be the whole Web; it usually corresponds to a smaller part of the Web (e.g. the stanford.edu domain, or the .edu domain).

Let $\mathcal{T}$ denote a universe of tags, e.g., all words appearing in all web pages or all words ever used in a search query. A source of page tags is a function $\tau : V \rightarrow 2^{\mathcal{T}}$; for every node $v \in \mathcal{V}$, $\tau(v)$ is a set of tags $\{t_1, t_2, \ldots\}$, with $t_i \in \mathcal{T}$. Also, a source of link destinations is a function $\ell : \mathcal{T} \rightarrow 2^{\mathcal{V}}$; for every tag $t \in \bigcup_{v \in \mathcal{V}} \tau(v)$, $\ell(t)$ is a set of page nodes $\{v_1, v_2, \ldots\}$ with $v_i \in \mathcal{V}$. These are pages with information relevant to the tag t. For a set of tags $A \in 2^{\mathcal{T}}$, we define the function $\ell_s : 2^{\mathcal{T}} \rightarrow 2^{\mathcal{V}}$ as $\ell_s(A) = \bigcup_{t \in A} \ell(t)$. The function ℓ_s is the set equivalent of ℓ.

For every pair of pages $p, v \in \mathcal{V}$ in the graph G, we define as $\mathrm{d}_G(p, v)$ the length of the shortest path from p to v in G, e.g. $\mathrm{d}_G(p, p) = 0$. Also, for every page $p \in \mathcal{V}$, we define as $\mathrm{nodes}_G(p)$ the set of reachable nodes from p in G, that is, all nodes $v \in V$ with $\mathrm{d}_G(p, v) > 0$. Note that $p \notin \mathrm{nodes}_G(p)$ since $\mathrm{d}_G(p, p) = 0$.

Navigational utility: We are now ready to give the formal definition of the *navigational utility* $u(p, G)$ of a page $p \in \mathcal{V}$ in the graph G. First, if $\mathrm{nodes}_G(p) = \emptyset$, i.e. p has no outgoing edges in G, we define $u(p, G) = 0$. For all $p \in \mathcal{V}$ with $\mathrm{nodes}_G(p) \neq \emptyset$, we define the navigational utility $u(p, G)$ as the product of two factors F_1, F_2:

$$u(p, G) = F_1 \cdot F_2 = \left(\frac{|\mathrm{nodes}_G(p)|}{|V| - 1} \right) \cdot \left(\frac{|\mathrm{nodes}_G(p)|}{\sum_{v \in \mathrm{nodes}_G(p)} \mathrm{d}_G(p, v)} \right) \quad (1)$$

Both F_1 and F_2 get values from the domain $[0, 1]$ and thus $u(p, G) \in [0, 1]$.

The factor F_1 measures the fraction of G that is reachable from p. For example, if $F_1 = 1$, then all the nodes in the graph are reachable from p. If $F_1 = 0$, then the only node reachable from p is p itself. If $F_1 = 0.5$, then half of the

Table 1: Navigational utility examples.

URL	F_1	F_2	u
www.stanford.edu	0.987296	0.225481	0.222617
www.stanford.edu/home/administration/policy.html	0.987296	0.224583	0.22173
www.stanford.edu/home/atoz/letters.html	0.987296	0.224452	0.221601
www.stanford.edu/home/atoz/lettera.html	0.987296	0.224179	0.221331
www.stanford.edu/home/atoz/letterr.html	0.987296	0.224163	0.221315
www.stanford.edu/home/administration/dept.html	0.987296	0.224017	0.221171

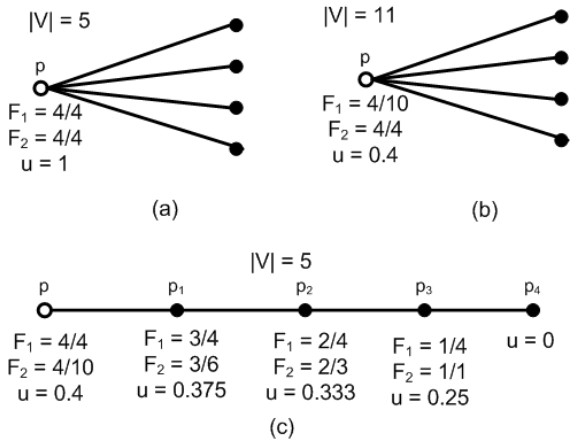

Figure 2: The navigational utility of sample nodes in graphs.

nodes in G are reachable from p.

The factor F_2 measures how "close" to p the nodes reachable from p are. If $F_2 = 1$, then every node in the set $\text{nodes}_G(p)$ is just one hop away from p. If $F_2 = 0.5$, then, on average, every node in the set $\text{nodes}_G(p)$ is $1/F_2 = 2$ hops away from p. The closer to 1 the F_2 value is, the better navigationally p is. The quantity $1/F_2$ measures p's average distance to all its reachable nodes, i.e., it measures how many steps are required to access every reachable node starting from p. $1/F_2$ corresponds to the closeness centrality measure of a vertex within a graph. Vertices that tend to have short distances within the graph have higher closeness.

Figure 2 shows the navigational utility of sample nodes in three graphs. In the graph G_a (Fig. 2(a)), $u(p, G_a) = 1$ since all nodes in the graph are reachable from p with just one hop (both the F_1 and the F_2 factors are equal to one). However, in the graph G_b (Fig. 2(b)) only a portion of graph nodes are reachable from p (since $|\mathcal{V}| = 11$). Thus, $u(p, G_b) = 0.4 < u(p, G_a)$. Finally, the sample graph G_c (Fig. 2(c)) illustrates how the navigational utility can be used to rank nodes based on their navigational utility. In the example, we get $u(p, G_c) > u(p_1, G_c) > u(p_2, G_c) > u(p_3, G_c) > u(p_4, G_c)$, i.e., the node p is a better browsing node than say node p_2 for example. Moreover, we can see how different graph structures can result in two nodes having the same navigational utility value. For example, notice that $u(p, G_c) = u(p, G_b)$. In the G_b graph, the node p has a perfect F_2 factor but a less good F_1 factor. Reversely, the node p of the G_c graph has a perfect F_1 score but a not so good F_2 score.

Overall, the higher the navigational utility of a page is, the higher the benefit for a visitor who browses through that page is. A visitor that browses through pages with high navigational utility can reach more pages (measured through the F_1 factor) with fewer clicks (measured through the F_2 factor). As an example, Table 1 shows the top 6 Stanford pages with the highest navigational utility values (u column) along with the values for the F_1 and F_2 factors. All of these pages are indeed central browsing points for Stanford visitors.

Navigational gain: Finally, for each page $p \in \mathcal{V}$ we define a *navigational gain function* $u_p : 2^{\mathcal{V}} \to \mathbb{R}$ that captures the navigational gain for p if we add links from p to all nodes in a set of nodes A from G. Let $A \subseteq \mathcal{V}$. Then:

$$u_p(A) = \frac{1}{u(p, G)} - \frac{1}{u(p, G_A)} \tag{2}$$

where $G_A = (\mathcal{V}, E_A)$ is the graph with the same nodes as G but with an edge $e = (u, v) \in E_A$ if and only if $e \in E$ or $(u \equiv p$ and $v \in A)$.

Intuitively, the navigational gain measures the decrease in the average distance of a node to all of its reachable nodes, caused by the addition of new links. The larger the decrease, the higher the increase in the navigational utility of the page. In other words, we are interested in linking p to nodes that will maximize p's navigational utility.

Problem description: A user is browsing a page p and we want to show him or her an auxiliary set of tags $\mathcal{Q}$ to help with user navigation. When the user clicks on a tag t, he is offered a set of "relevant" links $\mathcal{W}_t$. One way to select $\mathcal{Q}$ and the $\mathcal{W}_t$'s is by maximizing p's navigational utility.

We can then formally state the navigational tag/link selection problem (k, λ)-NTLSP as:

Definition 1 (Navigational tag/link selection). *Given a page $p \in \mathcal{V}$, a tags budget k and a links budget λ, find a set of tags $\mathcal{Q} \subseteq \tau(p)$ with $|\mathcal{Q}| \leq k$ and for each selected tag $t \in \mathcal{Q}$ find a set of link destinations $\mathcal{W}_t \subseteq \ell(t)$ with $|\mathcal{W}_t| \leq \lambda$ to maximize*

$$U(p) = u_p(\bigcup_{t \in \mathcal{Q}} \mathcal{W}_t) \tag{3}$$

Hardnes: The (k, λ)-NTLSP is NP-hard. We give a full proof in Appendix A. Even simple special cases of the (k, λ)-NTLSP are NP-hard; we give a full description of such cases in Appendix B. However, the good news is that, as we show in Appendix C, the objective function $U(p)$ is a monotonically increasing, submodular function. This implies that the Greedy heuristic, which we present in the following sections, is guaranteed to give a good approximation to the optimal solution.

5. TAG SELECTION METHODS

In this Section we present the three basic tag selection methods we use. In Section 7 we show how we combine them with the link destination selection algorithms of Section 6 to create heuristic algorithms for the general tag/link selection problem.

Algorithm 1 gives the skeleton of the tag selection algorithms. For a given page p, the algorithm considers the set of candidate tags $\tau(p)$ and the set of all associated link destinations $\ell(t)$ for every tag $t \in \tau(p)$, and selects the top k tags with the maximum value of a function $f(t, p, \ell(t))$. Based on the definition of f we have the following three heuristics (also summarized in Table 2).

(a) TF: The first tag selection method is based on the tag frequency. Let $t \in \tau(p$ be a candidate tag for a node p. Then the tag count of t in the given page node is simply the number of times a given tag appears in that page. This count is usually normalized to prevent a bias towards longer pages (which may have a higher tag count regardless of the actual importance of that tag for the document) to give a measure of the importance of t within p. Thus, we have the tag frequency defined as follows.

$$\mathrm{tf}_{t,p} = \frac{n_{t,p}}{\sum_{o \in \tau(p)} n_{o,p}} \tag{4}$$

where $n_{t,p}$ is the number of occurrences of the tag t in page p, and the denominator is the sum of number of occurrences of all tags in page p.

Algorithm 1 uses the function f defined as $f(t, p, \ell(t)) \equiv f(t, p) = \mathrm{tf}_{t,p}$ to simply select the top k tags, among the candidate tags, with the largest tag frequency. As we can see from Equation 4, the function f used by the algorithm does not depend on candidate link destinations $\ell(t)$ of a tag t.

(b) TFIDF: The second tag selection algorithm combines the tag frequency with the inverse page frequency. The inverse page frequency measures the general importance of the tag in the graph G. We have:

$$\mathrm{idf}_t = \log \frac{|V|}{|\{v : t \in \tau(v)\}|} \tag{5}$$

Then, the tfidf score of a tag t for a page p is defined as:

$$\mathrm{tfidf}_{t,p} = \mathrm{tf}_{t,p} \cdot \mathrm{idf}_t \tag{6}$$

Based on Equation 6 Algorithm 1 uses the function f defined as $f(t, p, \ell(t)) \equiv f(t, p) = \mathrm{tfidf}_{t,p}$ to select the top k tags among the candidate tags with the largest tfidf score. Again, the function f used by the algorithm does not depend on the candidate link destinations $\ell(t)$ of a tag t.

(c) Greedy: The greedy tag selection algorithm (Algorithm 1) uses the function $f(t, p, \ell(t)) = u_p(\ell(t))$. At each step, it selects the tag that increases the navigational gain of the page the most. As we show in Appendix C, the Greedy algorithm is guaranteed to give a good approximation to the optimal solution.

Table 2: Variations of the general tag selection algorithm.

Variation	$f(t, p, L)$
TF tag selection	$\mathrm{tf}_{t,p} = \frac{n_{t,p}}{\sum_{o \in \tau(p)} n_{o,p}}$
TFIDF tag selection	$\mathrm{tfidf}_{t,p} = \mathrm{tf}_{t,p} \cdot \mathrm{idf}_t$
Greedy tag selection	$u_p(\ell(t))$

Algorithm 1 General Tag Selection Algorithm

Input: integer k, node $p \in V$, candidate navigational tags $C = \tau(p)$, link destinations function ℓ
Output: navigational tags W
91: $W \leftarrow \emptyset$
92: **while** $|W| \leq k$ **do**
93: Find $t \in C \setminus W$ that maximizes $f(t, p, \ell(t))$
94: Set $W \leftarrow W \cup \{t\}$
95: Set $C \leftarrow C \setminus \{t\}$
96: **end while**
97: **return** W

6. LINK SELECTION METHODS

Similar to the tag selection methods, we present three basic link selection algorithms. Algorithm 2 gives the skeleton of the link selection algorithms. For a given page p and a tag t, the algorithm considers all candidate link destination nodes $v \in \ell(t)$ and at each step selects a node that maximizes the value of a function $f(p, t, v, W)$ where W is the set of nodes already selected by the algorithm. Based on the definition of f we have the following three variations (also summarized in Table 3).

(a) TF: The first link selection method is based on the tag frequency. Let $v \in \ell(t)$ be a candidate link destination node for a tag t. Then, we compute the tag frequency of t for v according to Equation 4. Algorithm 2 computes the tf score of tag t for all candidate link destination nodes and selects the λ nodes with the highest score. **(b) TFIDF:** The TFIDF link selection algorithm 2 works similarly to the TF Link Selection algorithm, with the only difference being that it selects nodes based on the tfidf score (Equation 6). **(c) Greedy:** Finally, the greedy link selection algorithm 2 at each step selects the node that increases the navigational gain of the page the most. As we show in Appenxid C, the Greedy algorithm is guaranteed to give a good approximation to the optimal solution.

In order to get link destinations for a page p that are semantically very relevant to p, we can consider as candidate destinations for the above-described algorithms only nodes that share many tags with p. For example, one could consider only pages that have at least 4 tags in common with p. However, if none of the link destination candidates satisfy this condition, we will have no pages to link to.

An alternative approach would be to count the common tags of every link destination candidate with p and boost candidates with many such common tags. Algorithm 3 generalizes Algorithm 2 by adding the term $|\tau(p) \cap \tau(v)|$ in the

Table 3: Variations of the general link selection algorithm.

Variation	$f(p, t, v, W)$
TF link selection	$\mathrm{tf}_{t,v} = \frac{n_{t,v}}{\sum_{o \in \tau(p)} n_{o,v}}$
TFIDF link selection	$\mathrm{tfidf}_{t,v} = \mathrm{tf}_{t,v} \cdot \mathrm{idf}_t$
Greedy link selection	$u_p(W)$

Algorithm 2 General Link Selection Algorithm

Input: integer λ, node $p \in V$, tag t, candidate link destination nodes $C = \ell(t)$
Output: navigational link destinations W
$W \leftarrow \emptyset$
while $|W| \leq \lambda$ **do**
  Find $v \in C \setminus W$ that maximizes $f(p, t, v, W \cup \{v\})$
  Set $W \leftarrow W \cup \{v\}$
  Set $C \leftarrow C \setminus \{v\}$
end while
return W

Algorithm 3 General Link Selection with Tag Overlap Algorithm

Input: integer λ, node $p \in V$, tag t, candidate link destination nodes $C = \ell(t)$, tag overlap threshold r, weights $\alpha, \beta \in [0, 1]$
Output: navigational link destinations W
$W \leftarrow \emptyset$
while $|W| \leq \lambda$ **do**
  Find $v \in C \setminus W$ that maximizes $\alpha \cdot \frac{f(p,t,v,W \cup \{v\})}{\sum_v f(p,t,v,W \cup \{v\})} + \beta \cdot |\tau(p) \cap \tau(v)|$
  Set $W \leftarrow W \cup \{v\}$
  Set $C \leftarrow C \setminus \{v\}$
end while
return W

function optimized. At each step, the algorithm selects the node v that maximizes the function:

$$\alpha \cdot \frac{f(p, t, v, W \cup \{v\})}{\sum_v f(p, t, v, W \cup \{v\})} + \beta \cdot |\tau(p) \cap \tau(v)| \qquad (7)$$

We can control the amount of boosting that the new term adds to nodes by properly adjusting the weights $\alpha, \beta \in [0, 1]$, with $\alpha + \beta = 1$. In addition, we now have to normalize the original term $f(p, t, v, W \cup \{v\})$ of Algorithm 2 by dividing it with the sum of the function values among all candidate nodes.

For example, if $\alpha = \beta = 0.5$ then the algorithm favors equally the value of f and the overlap term. If $\beta = 0$ then Algorithm 3 becomes equivalent to Algorithm 2, whereas if $\alpha = 0$ then Algorithm 3 takes into account just the tag overlap of the candidate nodes with the page p and ignores the value of f. In our experiments, when we used Algorithm 3 we set $\alpha = \beta = 0.5$.

7. TAG/LINK SELECTION HEURISTICS

Thus far we have presented tag selection and link selection heuristics. Using the algorithms of the previous two sections as basic blocks, we are now ready to build algorithms for the tag/link selection problem. There are two ways we can combine the tag and link selection algorithms as illustrated in Algorithms 4 and 5.

Algorithm 4 first selects λ links for every candidate tag and drops all the remaining links. Then it takes into account only the chosen links to select k tags. On the other hand, Algorithm 5 first uses all existing links to select k tags and it then selects λ links only for each of the k chosen tags.

Due to lack of space, in our experiments we focus on analyzing the performance of Algorithm 4. In addition, we restrict the SelectLinks function to the TF and TFIDF link selection heuristics only.

Algorithm 4 General Tag/Link Selection Algorithm

Input: integers k, λ, node $p \in V$, candidate navigational tags $C = \tau(p)$, link destinations function ℓ
Output: navigational tags W and $\forall t \in W$ link destinations L_t
$A \leftarrow C$
while $A \neq \emptyset$ **do**
  Pick $t \in A$
  $L_t \leftarrow$ SelectLinks$(\lambda, p, t, \ell(t))$
  $A \leftarrow A \setminus t$
end while
Let ℓ_λ be a links destination function defined by the sets $L_t, \forall t \in C$
$W \leftarrow$ SelectTags(k, p, C, ℓ_λ)
return W, L_t $\forall t \in W$

Algorithm 5 General Tag/Link Selection Algorithm

Input: integers k, λ, node $p \in V$, candidate navigational tags $C = \tau(p)$, link destinations function ℓ
Output: navigational tags W and $\forall t \in W$ link destinations L_t
$W \leftarrow$ SelectTags(k, p, C, ℓ)
while $W \neq \emptyset$ **do**
  Pick $t \in W$
  $L_t \leftarrow$ SelectLinks$(\lambda, p, t, \ell(t))$
  $W \leftarrow W \setminus t$
end while
return W, L_t $\forall t \in W$

8. EXPERIMENTS

The evaluation of navigational tags is inherently subjective, as is typical in the area of web search. This paper focuses on selecting the navigational tags and links and not on the different ways they can be presented to users. Hence in our evaluation we (a) use our navigational utility framework to show that navigational tags have the potential to improve web navigation, and we (b) use Mechanical Turk to evaluate the quality of the pages linked by our navigational tags.

In our experiments, we compare (a) the performance of query tags, pagetext tags and del.icio.us tags as navigational tag sources and (b) the performance of the Greedy tag selection algorithm against the TF and TFIDF tag selection baselines. From a practical standpoint, it is probably desirable to combine tags selected from many sources or methods. However, our focus is to understand the inherent characteristics of the different tag sources and as a result we do not consider blending tag sources or tag selection methods. We leave such analysis as an important part of our future work.

Our experimental evaluation consists of two components: The first component is the navigational utility framework. We study the effect of different tag sources and tag/link selection methods on the navigational utility of a page. The second component is human evaluation: We use Mechanical Turk to perform human evaluation of navigational tag sources.

Overall, we are interested in answering two basic ques-

tions: (a) Which source of tags is the most suitable source for navigational tags, and (b) how should we select and rank navigational tags.

In the next sections we first describe in detail the datasets we used in our evaluation and then we present our experimental findings.

9. DATASETS

Using the method described in [2], we collected a dataset from the stanford.edu domain. This dataset (Dataset Q) consists of all queried web pages from the three major search engines that appeared in the access logs of the main stanford web server during a period of 12 months beginning March 2007. Each entry in the dataset is a pair $\langle x, y \rangle$ where x is a URL and y is a query tag: an n-gram of size one (unigram), extracted from a search query. The dataset consists of 359,749 unique URLs, and 10,997,818 unique queries, out of which we extracted 937,075 unique query tags (unigrams). Although all possible n-grams extracted from a query can be candidate query tags, we limit our experiments in this work only to unigram query tags.

We also used a subset of the del.icio.us dataset used in [9] by keeping only URLs from the www.stanford.edu domain. Each entry in this subset is a a pair $\langle x, y \rangle$ where x is a URL and y is a tag for that URL by some delicious user. The collection period of this dataset is March-April of 2007 and as a result it has temporal overlap with Dataset Q. The subset we extracted (Dataset D), consists of 2,965 unique URLs and 5,670 unique tags.

We collected a crawl (Dataset C) of all pages (duplicates eliminated) whose URLs appear in either Dataset Q or Dataset D. The crawl was performed in early September 2008 and thus all pagetext for URLs in Dataset C corresponds to the web pages' version available online at that tim. We used the crawl to extract pagetext tags for the URLs in the dataset. We also found that 12,611 URLs were no longer available online; we did not consider those URLs in our experiments.

Finally, we used a Stanford web graph (Dataset G) obtained from the Infolab WebBase project. The webgraph was based on a December 2008 crawl and was used in all navigational utility computations.

10. FINDINGS

What is better to use as navigational tags: query tags or del.icio.us tags? Or perhaps pagetext tags? Also, how much do we gain/lose if we use one source of tags instead of another? Can we substitute many tags from a "bad" source for fewer tags from a "better" source, if we do not have access to the better source? Furthermore, can we quantify some sort of "exchange rate" between tag sources (e.g. one query tag is "equivalent" to three pagetext tags)?

Suppose now that we have decided which source of tags is the best. How do we select and rank tags from that source? Is looking at maximizing the navigational gain of a page the right thing to do? How does the Greedy algorithm compare to the TF/TFIDF baselines?

We answer questions like these in the next sections. We organize our findings as a set of six results. At the beginning of each result, we highlight the main result in a frame and we then present the experimental methodology, and the details of our finding.

10.1 Tag sources

Table 4: Parameters used in our experiments, along with their values domain used.

Parameter	Values domain
Tag Source	pagetext, query, del.icio.us tags
Link Source	query tags
Tag Selection	TF, TFIDF, Greedy
Link Selection	TF, TFIDF
# of tags	1-100
# of links	1-100

Result 1: Query tags are a better source for navigational tags than tags extracted from the pagetext. In addition, pagetext tags are better than del.icio.us tags.

Methodology: In our first experiment, we use the navigational utility framework to compare the different tag sources, i.e., we look at which source of tags is inherently more likely to increase the navigational utility of a page. Of course, the tag source is just one parameter that can affect the navigational utility; there exist numerous others. For example, simply by adding more navigational tags on a page, one expects the navigational utility of the page to increase. Similarly, by showing more navigational links per tag, one also expects the navigational utility of the page to increase.

Thus, we focus on studying the effects of the following six parameters (also shown in Table 4) to the navigational utility of a page: (a) tag source, (b) link source, (c) tag selection method, (d) link selection method, (e) number of tags selected, and (f) number of links per tag selected. For every parameter, we predefine a values domain (Table 4) and experimentally study how different combinations of parameter values affect the navigational utility of pages.

Experiment Description: We first computed the navigational utility of a sample of two hundred, pages from the Dataset Q. We used the webgraph from Dataset G and implemented the navigational utility computation using the Boost C++ graph library.

In order to compare the tag sources, we fixed the tag and link selection methods and asked how the different tag sources affect the navigational utility (NU) of a page if we vary the number of tags used or the number of links per tag selected.

Findings: The typical navigational gain (NG) vs number of tags graph (NG-tags graph) looks like the top graph in Figure 3. This graph corresponds to one page, the main Stanford web page (www.stanford.edu.). In the graph, we plot the navigational gain generated by different tag sources (y-axis) as we vary the number of tags selected (x-axis). For this graph, the tags and links are selected using the TF tag and link selection methods, and we set the number of navigational links per tag to ten.

As we increase the number of tags selected (x-axis), all three tag sources cause an increase in the page's navigational gain (y-axis). However, we can clearly see that query tags quickly (five tags and more) contribute the most to the navigational gain and consistently dominate the pagetext and del.icio.us tags thereafter. Moreover, pagetext tags generate higher navigational gain compared to del.icio.us tags.

Recall that higher navigational gain means that the added tags are more useful in leading to other content (in this case content in the Stanford domain). Thus, query tags are more effective in covering this domain.

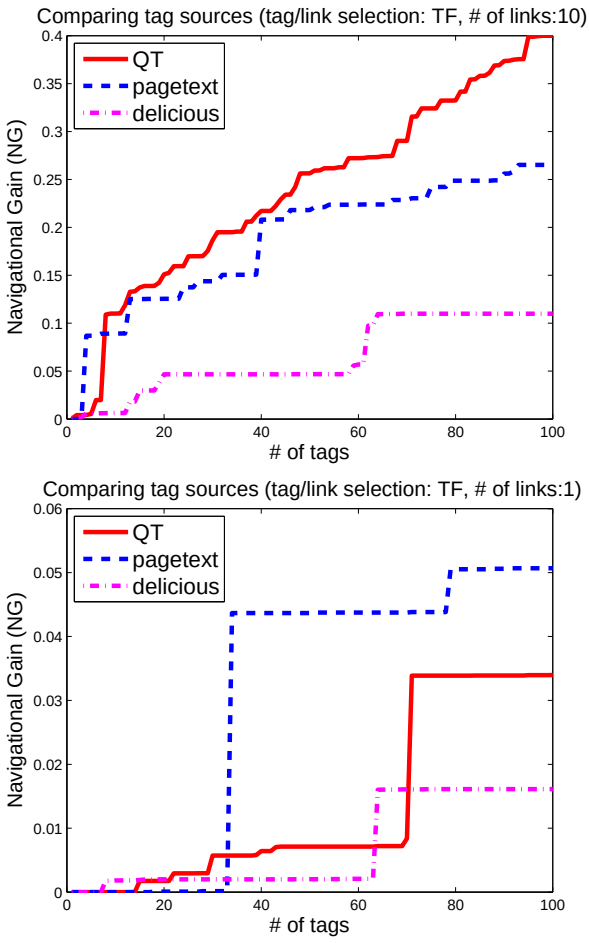

Figure 3: Comparing tag sources. Fix tag/link selection methods. How tag sources compare if we vary the # of tags and # of links used?

Also, as we can see, query tags provide almost twice the navigational gain compared to pagetext tags and at least four times more than deli.icio.us tags. Furthermore, pagetext tags generate at least twice the navigational gain as del.icio.us tags.

Overall, we found that on average (among all two hundred pages in our sample, all different values of the number of tags selected and all different values of the number of links per tag selected), pagetext tags generate about 1.8 times higher navigational gain than pagetext tags, and 4.3 times higher navigational gain than del.icio.us tags. Of course, there are exceptions to this observation. For example, in the bottom graph of Figure 3 we show the navigational gain vs number of tags graph for www.stanford.edu in the corner case where we only select one link per tag. Query tags dominate pagetext and delicious tags for the first 35 tags. After that the picture changes and pagetext ends up generating the largest navigational gain, with the query tags following. There is also a short range in the number of tags parameter ([65-70] tags) where del.icio.us tags dominate query tags.

> **Result 2:** Human evaluators find pages linked by query tags more relevant than pages linked by pagetext or del.icio.us tags.

Methodology: Using Mechanical Turk we designed an experiment that involved human evaluation of the navigational tag sources. In this experiment, we used the Greedy tag selection method. This decision was based on our analysis using the navigational utility (Result 4) which showed that the Greedy tag selection method outperforms the TF and TFIDF baselines. We also fixed the link selection method to "TF with tag overlap" (Algorithm 3) and focused on varying the source of query tags (query (QT), pagetext(PT) and delicious(DT) tags). Finally, we fixed both the number of selected tags and links per tag to ten.

Data: We generated navigational links for the main Stanford webpage (`http://www.stanford.edu/`) from the three navigational tag sources. We then did the same for every destination page of the generated navigational links to generate new navigational links. By repeating this process, we collected a list of all pages that someone could reach had he had started from the main Stanford page and had he followed at most three navigational links (i.e., pages that were at most three navigational links away).

This process left us with three sets of pages: each set containing pages reachable by navigational links generated from a different tags source. We then randomly selected 100 pages from each set. For every pair of methods (query-pagetext tags (QT-PT), query-del.icio.us tags (QT-DT), pagetext-del.icio.us tags (PT-DT)), we randomly matched pages and generated three sets (QT-PT, QT-DT and PT-DT) with a hundred pairs of pages each. The total number of pairs was 300.

Experiment design: We showed random page pairs from all three sets (QT-PT, QT-DT and PT-DT) to different Mechanical Turk workers and asked them to perform a task with the following description:

Experiment Description 1. *Suppose you are the web designer of the main web page of Stanford University. This page contains information about the university and its departments, the student life, student resources, news, etc. Your task as a web designer is to decide which pages to link to from the university's page.*

Below, we are showing you two possible destination pages for a link. We are asking you to examine both pages and then click on the one that you think is the most appropriate destination for a link.

To ensure high quality results, we generated 10 extra pairs of pages with a trivial winner page per pair. We then randomly inserted those pairs into all of our Mechanical Turk tasks and only accepted ratings from workers that gave the correct answer to all 10 injected pairs (certified workers).

When a worker is asked to choose between two pages, he is shown screenshots from both pages side by side so he can examine them. After indicating the best of the pair, the worker moves on to the next pair. The worker finishes the task after evaluating 110 pairs (100 test pairs plus 10 injected ones). Since we discarded all of the results from non-certified workers, we repeateadly generated Mechanical

Table 5: Human Evaluation: Number of URLs that have 5 or more judgment agreements.

QT > DT		DT > QT	
agreements	URLs (%)	agreements	URLs (%)
5	100 (100%)	5	0
6	100 (100%)	6	0
7	83 (83%)	7	0
8	75 (75%)	7	0
9	69 (69%)	7	0
QT > PT		**PT > QT**	
agreements	URLs (%)	agreements	URLs (%)
5	95 (95%)	5	5 (5%)
6	95 (95%)	6	3 (3%)
7	80 (80%)	7	0 (0%)
8	58 (58%)	8	0 (0%)
9	53 (53%)	9	0 (0%)

Turk tasks untill we got ratings from 9 different certified workers for every set of pairs (QT-PT, QT-DT and PT-DT).

Findings: Table 5 summarizes the results of the experiment. We only show the results from the comparisons of QT with PT (top of figure) and QT with DT (bottom). We ommit the comparison between PT and DT since the experiment showed that there was not enough statistical significance to decide which of the two gives better results.

In Table 5, we show the number of URL pairs that have 5 or more judgment agreements. For example, looking at the QT > DT portion of the table, the second line is read as follows: For all 100 out of the 100 URL pairs of the QT-DT set, at least 6 out of the 9 total judges decided that the URL selected by QT was more relevant compared to the URL selected by DT. Similarily, looking at the QT > PT portion of the table, we see that for 95 URL pairs from the QT-PT set, at least 6 of the 9 judges agreed that the QT-selected URL was more relevant than the PT one. We also show the DT > QT and PT> QT comparison results. For example, in the PT > QT portion of the table, we see that for 3 pairs, at least 6 people agreed that the PT-generated URL was more relevant than the QT one.

Overall, using the data of Table 5, we found that with very high statistical significance (p-value $\ll 0.001\%$) query tags select more interesting pages than del.icio.us or pagetext tags.

> **Result 3:** We can substitute query tags with pagetext or del.icio.us tags at a greater "price", in cases where query tags are not available.

Methodology: In our first experiment (Result 1) we compared the different tag sources using the navigational utility framework. Such analysis allows as to estimate the navigational utility loss, if we decide to use pagetext tags instead of query tags. In this experiment, we use the navigational utility framework to establish an "exchange rate" between query tags and pagetext tags. Since we don't expect query tags n to always be available for every page, we want to be able to decide whether and how we can compensate for the lack of query tags with pagetext tags. We looked at the same sample of 200 pages used in Result 1. For a given number of query tags, we asked how many pagetext tags are required to generate the same navigational gain in a page.

For every page, we can present this relationship between tag sources using a tag sources equivalence graph. In a graph like Fig. 4, the x-axis corresponds to query tags and the y-axis to pagetext tags. Drawing the point (x, y) in this graph (red, solid line) means that x query tags generate the same navigational gain as y pagetext tags for a page. The green dashed line corresponds to the $y = x$ graph and is provided for reference. It represents cases where the two sources are exactly equivalent, i.e., k query tags generate the same gain as k pagetext tags for all k. Points in the region above the $y = x$ line correspond to cases where query tags provide higher navigational gain, whereas points in the region below that line represent cases where pagetext is more beneficial.

Findings: The typical tag sources equivalence graph is shown in Figure 4. This graph corresponds to the main Stanford web page (www.stanford.edu.). The red, solid line represents the exchange rate between tag sources. For example, we see that 15 query tags are equivalent to 26 pagetext tags or that 25 query tags are equivalent to 40 pagetext tags. The flat line in the [20-38] range of query tags corresponds to the jump of the NG-tags graph (top graph in Figure 3) at around 40 tags.

Overall, by averaging over the 200 pages we looked at and over all different values for the number of tags used, we found that one query tag is equivalent to 1.8 pagetext tags.

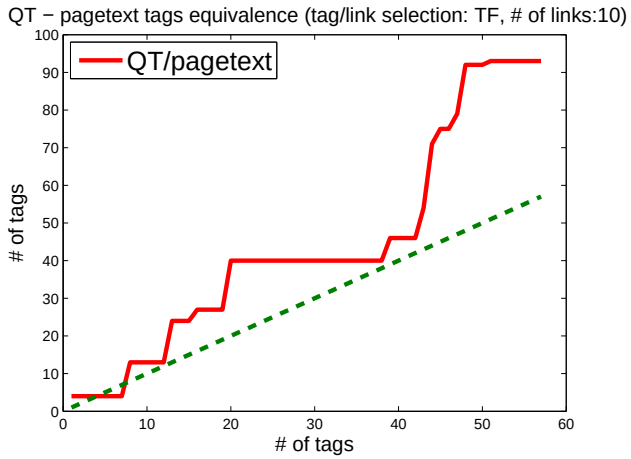

Figure 4: query tags - pagetext tags equivalence

10.2 Tag selection methods

> **Result 4:** The Greedy tag selection method outperforms the TF and TFIDF baselines. Furthermore, the TF and TFIDF heuristics give results of similar quality.

Methodology: In this experiment, we use the navigational utility framework to compare the different tag selection methods, i.e., we look at which tag selection method is inherently more likely to increase the navigational utility of a page.

Experiment Description: We first computed the navigational utility of a sample of two hundred pages from the

Dataset Q. In order to compare the tag selection methods, we fixed the tag source and the link selection method and asked how the different tag selection methods affect the navigational utility of a page if we vary the number of tags used or the number of links per tag selected.

Findings: The typical navigational gain (NG) vs number of tags graph (NG-tags graph) looks like the top graph in Figure 5. This graph corresponds to the main Stanford web page (www.stanford.edu). In the graph, we are plotting the page's navigational gain caused by the use of different tag selection methods (y-axis), as we vary the number of tags selected (x-axis. We are using query tags as the source of tags, links are selected using the TF baseline, and we set the number of navigational links per tag to seven.

As we increase the number of tags selected (x-axis), all three tag selection methods cause an increase in the page's navigational gain (y-axis). However, we can clearly see that Greedy dominates TF and TFIDF right from the first tag selected. Moreover, TF and TFIDF contribute almost identically to the page's navigational utility. Also, as we can see, Greedy provides almost four times the navigational gain compared to TF and TFIDF.

Overall, we found that on average (over all two hundred pages in our sample, all different values of the nunber of tags, and all different values of the number of links per tag selected), Greedy selects tags that generate 4 times higher navigational gain than the tags selected by TF or TFIDF. Moreover, Greedy is always superior to TF and TFIDF in all cases.

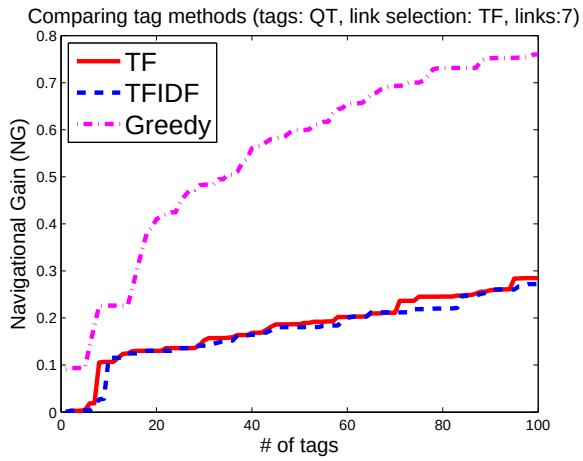

Figure 5: query tags - pagetext tags equivalence

> **Result 5:** Human evaluators find pages selected by the Greedy tag selection algorithm more relevant than pages selected by the TF or TFIDF tag selection methods.

Methodology: Similar to Result 3, we used Mechanical Turk to design an experiment that involved human evaluation of the navigational tag sources. In the experiment, we used query tags as the navigational tags source. This decision was based on our previous analysis using the navigational utility (Result 1), which showed that query tags are prefered over pagetext and del.icio.us tags. We also fixed the link selection method to "TF with tag overlap" (Algorithm 3) and focused on varying the tag selection methods (TF, TFIDF and Greedy). Finally, we fixed both the number of selected tags and links per tag to ten.

Table 6: Human Evaluation: Number of URL pairs that have 5 or more judgment agreements.

Greedy > TF		**TF > Greedy**	
agreements	pairs (%)	agreements	pairs (%)
5	100 (100%)	5	0
6	49 (49%)	6	0
7	0	7	0
Greedy > TFIDF		**TFIDF > Greedy**	
agreements	pairs (%)	agreements	pairs (%)
5	61 (61%)	5	39 (39%)
6	61 (61%)	6	29 (29%)
7	48 (48%)	7	20 (20%)
8	34 (34%)	8	10 (10%)
9	7 (7%)	9	3 (3%)

Data: We used the three tag/link selection methods to generate navigational links for the main Stanford webpage (`http://www.stanford.edu/`). Using a similar data generation process as in Result 3, we created three sets (Greedy-TF, Greedy-TFIDF, TF-TFIDF) with a hundred pairs of pages each.

Experiment design: Again, our experiment was similar to the one used in Result 3. Evaluators were shown pairs of pages and were asked to select the both appropriate link destination for the Stanford page.

Results: Table 6 summarizes the results of the experiment. We only show the results from the comparisons of Greedy with TF and Greedy with TFIDF. We ommit the comparison between TF and TFIDF since the experiment showed that there was not enough statistical significance to decide which of the two gives better results.

In Table 6, we show the number of URL pairs that have 5 or more judgment agreements. For example, looking at the Greedy > TF portion of the table, the second line is read as follows: For 49 out of the 100 URL pairs of the Greedy-TF set, at least 6 out of the 9 total judges decided that the URL selected by Greedy was more relevant compared to the URL selected by TF. Similarily, looking at the Greedy > TFIDF portion of the table, we see that for 61 URL pairs from the Greedy-TFIDF set, at least 6 of the 9 judges agreed that the Greedy-selected URL was more relevant than the TFIDF one. We also show the TF > Greedy and TFIDF > Greedy comparisons. For example, in the TFIDF > Greedy portion of the table, we see that for 39 pairs, at least 5 people agreed that the TF-generated URL was more relevant than the Greedy one.

Overall, using the data of Table 6, we found that with very high statistical significance (p-value $\ll 0.001\%$) Greedy selects more relevant pages than both TF and TFIDF.

> **Result 6:** If we cannot afford to run Greedy, we can get similar results by displaying more TF or TFIDF selected tags.

Methodology: In our fourth experiment (Result 4) we compared the different tag selection methods using the

navigational utility framework. Such analysis allows us to estimate the navigational utility loss, if we decide to run TF instead of Greedy. In this experiment, we use the navigational utility framework to establish an "exchange rate" between Greedy and TF. Since running Greedy requires obtaining the webgraph (or at least part of it), it might not always be feasible to run Greedy. In such cases, we want to be able to decide whether and how we can compensate for running Greedy with the simpler TF method.

We looked at the same sample of 200 pages used in Result 4. For a given number of Greedy-selected tags, we asked how many TF-selected tags are required to generate the same navigational gain in a page.

For every page, we can present this relationship between tag selection methods using a tag methods equivalence graph, similar to the tag sources equivalence graph we described in Result 3.

Findings: The typical tag methods equivalence graph is shown in Figure 6. This graph corresponds to the main Stanford web page (www.stanford.edu). The green, dashed line corresponds to the $y = x$ graph and is provided for reference. The red, solid line represents the exchange rate between tag selection methods. For example, we see that 15 Greedy-tags are equivalent to 95 TF-tags or that 5 Greedy-tags are equivalent to 10 TF-tags. The flat line in the [8-14] range of query tags corresponds to jump of TF in the NG-tags graph (Fig. 5) at about 70 tags.

Overall, by averaging over all 200 page we looked at, all different values for the number of tags used and all different values for the number of links per tag selected, we found that one Greedy-selected tag is equivalent to 4 TF-selected tags.

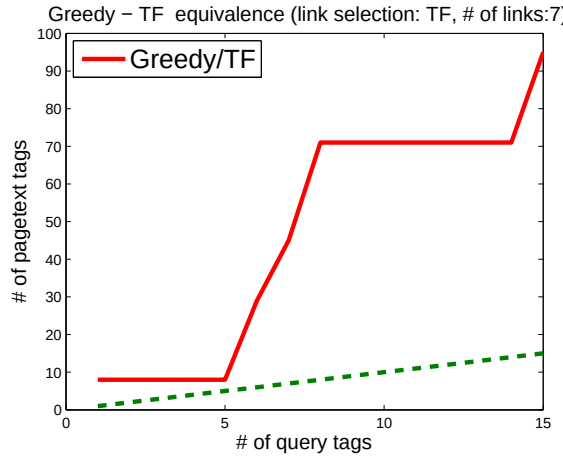

Figure 6: query tags - pagetext tags equivalence

11. RELATED WORK

The related work falls into three main categories.

Browsing and Searching. The importance of supplying context with query results is well recognized. To deal with the problem of no-context when searching, search engines are usually providing query suggestions to the user [12].

Mining query and activity logs. Previous work has looked at how query logs can be utilized by a search engine (e.g., [1], [4]) to improve ranking of search results or to generate query rewrites and query suggestions. In [2], we showed how web users can extract portion of the search engine query logs from webserver access logs, and introduced the notion of query tags: automatically generated web page tags from query logs. In [10] we study how to use web activity logs to recommend links to web users.

Navigational tag/link selection problem. Finally, the formulation of the navigational tag/link selection problem is related to the optimal bookmark selection problem [8], the optimal selection of HTTP proxies ([13]) and the recently studied quicklink selection problem ([5]).

12. CONCLUSIONS

We proposed a navigational menu as a tool to improve navigation on the Web. When a user is browsing a page, we show an auxiliary set of tags to help his navigation. When the user clicks on a tag, he is offered a set of relevant links. We defined the navigational utility of a page as a conceptual metric useful for understanding and evaluating different navigational tags sources and tag/link selection methods. Based both on the navigational utility framework and human-involved evaluation, we studied (a) which source of tags is the most appropriate for navigational tags and (b) how should we select and rank navigational tags for a given page. We found that search queries significantly improve the navigational utility of pages and help people discover very interesting content.

13. REFERENCES

[1] I. Antonellis, H. Garcia-Molina, and C.-C. Chang. Simrank++: Query rewriting through link analysis of the click graph. In *PVLDB, Vol 1, No 1.*

[2] I. Antonellis, H. Garcia-Molina, and J. Karim. Tagging with queries: How and why? In *Proc. WSDM 2009.*

[3] M. Bilenko and R. W. White. Mining the search trails of surfing crowds: Identifying relevant websites from user activity. In *Proc. WWW, 2008.*

[4] D. Chakrabarti, R. Kumar, and K. Punera. Quicklink selection for navigational query results. In *Proc. WWW, 2009.*

[5] C. Chekuri and S. Khanna. A ptas for the multiple knapsack problem. In *Proc. SODA, 2000.*

[6] R. Cohen and L. Katzir. The generalized maximum coverage problem. In *IPL, 2008.*

[7] J. Czyzowicz, E. Kranakis, D. Krizanc, A. Pelc, and M. V. Martin. Optimal assignment of bookmarks to web pages. In *Ars Combinatoria, 2007.*

[8] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social-bookmarking improve web search? In *Proc. WSDM 2008.*

[9] J. Karim, I. Antonellis, V. Ganapathi, and H. Garcia-Molina. A dynamic navigation guide for webpages. In *TR, http://ilpubs.stanford.edu:8090/946/.*

[10] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. In *IPL, 2008.*

[11] R. Kraft, C. C. Chang, F. Maghoul, and R. Kumar. Searching with context. In *Proc. WWW, 2006.*

[12] B. Li, M. J. Golin, G. F. Italiano, and X. Deng. On the optimal placement of web proxies in the internet. In *Proc. INFOCOM, 1999.*

[13] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. In *Mathematics of Operations Research, 1978*.

[14] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. In *Mathematical Programming, 1978*.

APPENDIX

A. HARDNESS OF NTLSP

In this section we show the hardness of the navigational tag/link selection problem (NTLSP).

LEMMA 1. *The navigational tag/link selection problem is NP-hard.*

PROOF. We will show that the navigational tag/link selection problem generalizes the NP-hard bookmark assignment problem (BAP) [8]. An instance of the k_B-BAP problem assumes a directed graph $G_B = (V_B, E_B)$, a node $p \in V_B$ and a positive integer $k_B \leq |V_B|$. The task is to select a set of nodes $B \subseteq V_B$ of size k_B to maximize $u_p(B)$.

Consider the following instance I of the navigational tag/link selection problem. Let a graph $G = (V, E)$ and a node $p \in V$. Let $|\tau(p)| = |V|$ and (i) $\forall t \in \tau(p),\ |\ell(t)| = 1$ and (ii) $\forall t_i, t_j \in \tau(p),\ \ell(t_i) \neq \ell(t_j)$. We are interested in selecting a set of k tags $\mathcal{Q} \subseteq \tau(p)$ and for each selected tag $t \in \mathcal{Q}$ select a set of nodes λ $\mathcal{W}_t \subseteq \ell(t)$ to maximize $u_p(\bigcup_{t \in \mathcal{Q}} \mathcal{W}_t)$.

The instance I reduces to the following instance of k-BAP: $G_B(V_B, E_B) = G$ and $k_B = k$.

Thus, the navigational tag/link selection problem generalizes the bookmark assignment problem and is NP-hard. □

Even simple special cases of the (k, λ)-NTLSP are NP-hard; we give a full description of such cases in the following section.

B. SPECIAL CASES OF NTLSP

Even simple special cases of the (k, λ)-NTLSP are NP-hard, e.g. if we completely ignore the tag selection subproblem and focus only on selecting at most λ link destination we still get the λ-BAP which is NP-hard [8]. If now, we ignore the tag selection subproblem and furthermore we relax the objective function, i.e. assume that $u_p(A) = \sum_{a \in A} u_p(a)$, we end up with the 0-1 knapsack problem which again is NP-hard.

In this section we analyze special cases of the (k, λ)-NTLSP and give greedy approximation algorithms. Given the hardness of the general problem, we will later develop heuristics that utilize as solution components the greedy algorithms from the special cases of the general optimization problem.

Based on the number of available tags/links and several relaxations of the objective function we have the following special cases of (k, λ)-NTLSP.

B.1 One Tag

If there is only one available tag for the node p, i.e., $|\tau(p)| = 1$, then the (k, λ)-NTLSP becomes the bookmark assignment problem (λ-BAP) [8]. The λ-BAP asks for selecting λ link destinations for a node p of a graph so that the average distance from p to all nodes in the graph is minimized. The λ-BAP has been shown to be NP-hard for a general graph [8] but an $O(|V|^3\lambda^2)$ dynamic programming algorithm solves the problem if G is a tree [13].

We use the greedy Algorithm 6 to solve the λ-BAP. As we show in the Appendix, the objective function of the (k, λ)-NTLSP is a non-decreasing and submodular function and thus the greedy Algorithm 6 gives a provably good approximation to the optimal solution [14, 15].

If we relax the objective function of the (k, λ)-NTLSP and assume that $u_p(A) = \sum_{a \in A} u_p(a)$, i.e., the utility of a set of nodes is the sum of the node weights, we end up with an instance of the $0-1$ knapsack problem which is NP-hard. A dynamic programming solution for the $0-1$ knapsack problem runs in pseudo-polynomial time $O(|V|\lambda)$. Since in practice $\lambda \simeq 10$ this algorithm has practically linear time complexity.

Algorithm 6 BAP-Greedy

Input: integer λ, graph $G = (V, E)$, node $p \in V$, link destinations function ℓ
Output: link destinations W
91: $W \leftarrow \emptyset$
92: **while** $|W| \leq \lambda$ **do**
93: Find $v \in C \setminus W$ that maximizes $u_p(W \cup \{v\}) - u_p(W)$
94: Set $W \leftarrow W \cup \{v\}$
95: **end while**
96: **return** W

B.2 Less than k tags

Suppose there are less than k tags available for the node p, i.e., $|\tau(p)| \leq k$. Then, if we relax the objective function of the (k, λ)-NTLSP and assume that $u_p(A) = \sum_{a \in A} u_p(a)$, i.e., the utility of a set of nodes is the sum of the node weights, we end up with an instance of the multiple knapsack problem which is NP-hard. The multiple multidimensional knapsack problem is a generalization of the $0-1$ knapsack problem. It assumes we are given a set of items (nodes in our case) and knapsacks (tags in our case) such that each item has a profit and a size, and each knapsack has a capacity. The goal is to find a subset of items of maximum profit such that they have a feasible packing in the knapsacks. A polynomial time approximation scheme (PTAS) exists for the multiple knapsack problem [6].

B.3 One link per tag

If there is only one available link node per tag for a node p, i.e. $|\ell(t)| = 1, \forall t \in \tau(p)$ then it is easy to see that the (k, λ)-NTLSP becomes the bookmark assignment problem (k-BAP) [8]. Everything discussed in Section B.1 applies here as well.

B.4 Less than λ links per tag

Now, suppose there are less than λ link nodes available per tag for a node p, i.e. $|\ell(t)| \leq \lambda, \forall t \in \tau(p)$. Then, if we relax the objective function of the (k, λ)-NTLSP and assume that $u_p(A) = \sum_{a \in A} u_p(a)$, i.e., the utility of a set of nodes is the sum of the node weights, we end up with an instance of the budgeted maximum coverage problem (BMC, [11]). In [11], the authors give a greedy algorithm which gives provably good approximation to the optimal solution.

B.5 Relaxed objective function

If we relax the objective function of the (k, λ)-NTLSP and assume that $u_p(A) = \sum_{a \in A} u_p(a)$, i.e., the utility of a set of nodes is the sum of the node weights, then we end up with an instance of the generalized maximum coverage problem (GMCP, [7]). A modified greedy algorithm gives a provably good approximation to the problem.

C. SUBMODULARITY OF NAVIGATIONAL UTILITY

Let $G = (V, E)$ be a directed graph and let $p \in V$. We denote as nodes$(p) \subseteq V$ the subset of nodes reachable from p. Without loss of generality, we assume that nodes$(p) = V$. Let $S_p = (V, E_p)$ be a shortest path tree for G rooted on p. S_p is a subgraph of G constructed so that the distance between p and all other nodes in V is minimal. Note that S_p is not unique; there might be more than one shortest paths from p to a node. Let depth(q) be the depth of a node $q \in V$ in the tree S_p, i.e., depth$(q) = d(p, q)$. An edge $e = (r, s) \in E$ is called a *red edge* if and only if $e \in E_p$.

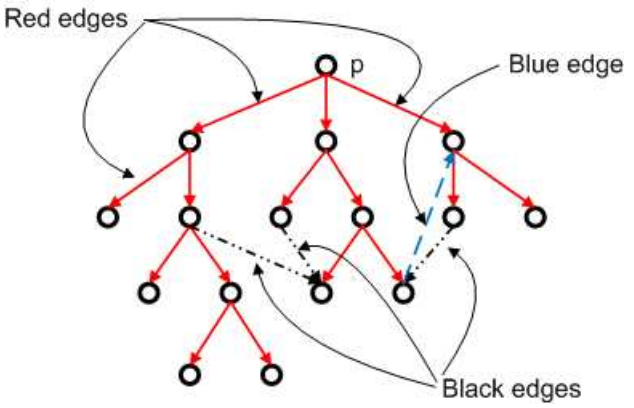

Figure 7: Red, blue and black edges

Let $e = (r, s) \in E$ be an edge of G. Then, the following two are true:

(a) $e \in E_p$, (i.e., e is red), $\Rightarrow$ depth$(r) =$ depth$(s) - 1$

(b) depth$(r) \geq$ depth$(s) \Leftrightarrow e \in E \setminus E_p$

Note that the converse of (a) above does not hold, i.e. depth$(r) =$ depth$(s) - 1 \Rightarrow e \in E_p$ or $e \in E \setminus E_p$. An edge $e = (r, s)$ with $e \in E \setminus E_p$ and depth$(r) =$ depth$(s) - 1$ is called a *black edge*. Since a shortest path tree S_p is not unique, an edge in G might be red or black depending on which tree S_p is chosen. Finally, an edge $e = (r, s)$ with depth$(r) \geq$ depth(s) is a *blue edge*. Figure 7 shows examples of red, blue and black edges of a graph.

For every $q \in V$ we define a set of paths interesting_paths(q). This set contains any path in G with q as its start vertex. We denote as interesting_nodes(q) a set that contains any node that appears in any path $P \in$ interesting_paths(q). Obviously, $q \in$ interesting_nodes(q). Intuitively, interesting_nodes(q) are the nodes whose shortest distance to p *might* decrease if we add an edge (p, q) to the graph. If $A \subseteq 2^V$ is a set of nodes, we overload the definition of interesting_nodes and define interesting_nodes$(A) = \bigcup_{q \in A}$ interesting_nodes(q).

Also, if A is a set of nodes in V, i.e. $A \subseteq 2^V$, we denote as interesting_paths$_A(q)$ the set which contains any path in G_A with q as its start vertex. Similarily, interesting_nodes$_A(q)$ is a set that contains any node that appears in any path $P \in$ interesting_paths$_A(q)$.

Finally, we denote as path(q) the unique path in G that starts from p, ends in q and contains only red edges. Also, if A is a set of nodes in V, i.e. $A \subseteq 2^V$, path$_A(q)$ denotes the unique path in $G' = (V, E \cup (\cup_{i \in A}(p, i)))$ that starts from p, ends in q and contains only red edges.

Below, we will show that the function $u_p : 2^V \rightarrow \mathbb{R}$ with $u_p(A) = u(p, G) - u(p, G_A)$ is monotonically increasing and submodular. This implies that the approximation ratio of the Greedy algorithm is $1 - \left(\frac{k-1}{k}\right)^k$ which as $k \rightarrow \infty$ approaches $1 - 1/e = 0.63$.

Throughout our analysis we assume for simplicity that $u(p, G) = \sum_{v \in V} d_G(p, v)$. Although this definition of $u(p, G)$ lacks the multiplication factor $\frac{|V|}{|\text{nodes}_G(p)|^2}$ from the definition we use in Equation 1, the analysis remains the same in both cases.

We first show that u_p is monotonically increasing.

THEOREM 1. *The function $u_p : 2^V \rightarrow \mathbb{R}$ is a monotonically increasing function, i.e. $\forall A \subseteq B \subseteq 2^V$, $u_p(A) \leq u_p(B)$*

PROOF. We have:

$$\begin{aligned} u_p(B) - u_p(A) &= u(p, G) - u(p, G_B) - u(p, G) + u(p, G_A) \\ &= u(p, G_A) - u(p, G_B) \\ &= \sum_{v \in V} d_{G_A}(p, v) - \sum_{v \in V} d_{G_B}(p, v) \\ &= \sum_{v \in V} [d_{G_A}(p, v) - d_{G_B}(p, v)] \end{aligned}$$

But since $A \subseteq B$, G_B contains all edges of G_A plus some additional ones. So, $d_{G_A}(p, v) \geq d_{G_B}(p, v)$.

Thus, $u_p(B) - u_p(A) \geq 0 \Leftrightarrow u_p(A) \leq u_p(B)$. □

Before we prove that u_p is submodular we need to prove the following Theorem for the *interesting_nodes* set.

THEOREM 2. *Let $A \subseteq B \subseteq 2^V$. Then, interesting_nodes$_B(\epsilon) \subseteq$ interesting_nodes$_A(\epsilon)$, $\forall \epsilon \in V$.*

PROOF. Consider $a \in B \setminus A$ and let $C =$ interesting_nodes$_A(\epsilon) \cap$ interesting_nodes$_A(a)$. If $C = \emptyset$ then interesting_nodes$_A(\epsilon) =$ interesting_nodes$_{A \cup \{a\}}(\epsilon)$.

However, if $C \neq \emptyset$ then interesting_nodes$_{A \cup \{a\}}(\epsilon) \subseteq$ interesting_nodes$_A(\epsilon)$ since all nodes in C will no longer be interesting nodes for ϵ in the graph $G_{A \cup \{a\}}$ but instead will become interesting nodes for the node a.

By repeating the same reasoning for all $a_i \in B \setminus A$ we conclude that interesting_nodes$_B(\epsilon) \subseteq$ interesting_nodes$_A(\epsilon)$, $\forall \epsilon \in V$. □

COROLLARY 1. *Let $A \subseteq B \subseteq 2^V$. Then, $|$interesting_nodes$_B(\epsilon)| \leq |$interesting_nodes$_A(\epsilon)|$, $\forall \epsilon \in V$.*

We are now ready to show that u_p is submodular.

THEOREM 3. *The function $u_p : 2^V \rightarrow \mathbb{R}$ is submodular.*

PROOF. We will show that for any $A \subseteq B \subseteq 2^V$ and $\epsilon \in 2^V \setminus B$:

$$u_p(B \cup \{\epsilon\}) - u_p(B) \leq u_p(A \cup \{\epsilon\}) - u_p(A)$$

We have:

$$u_p(A \cup \{\epsilon\}) = u_p(A) + \sum_{x \in \text{interesting\_nodes}_A(\epsilon)} (|\text{path}_A(x)| - |\text{path}_{A \cup \{\epsilon\}}(x)|)$$

or

$$u_p(A \cup \{\epsilon\}) - u_p(A) = \sum_{x \in \text{interesting\_nodes}_A(\epsilon)} (|\text{path}_A(x)| - |\text{path}_{A \cup \{\epsilon\}}(x)|)$$

Similarily,

$$u_p(B \cup \{\epsilon\}) - u_p(B) = \sum_{x \in \text{interesting\_nodes}_B(\epsilon)} (|\text{path}_B(x)| - |\text{path}_{B\cup\{\epsilon\}}(x)|)$$

Using Corollary 1 we get:

$$u_p(B \cup \{\epsilon\}) - u_p(B) \leq \sum_{x \in \text{interesting\_nodes}_A(\epsilon)} (|\text{path}_B(x)| - |\text{path}_{B\cup\{\epsilon\}}(x)|) \quad (8)$$

Now, in order to compare the quantities $u_p(A \cup \{\epsilon\}) - u_p(A)$ and $u_p(B \cup \{\epsilon\}) - u_p(B)$ we will just compare $|\text{path}_B(x)|$ to $|\text{path}_A(x)|$ and $|\text{path}_{B\cup\{\epsilon\}}(x)|$ to $|\text{path}_{A\cup\{\epsilon\}}(x)|$ for $x \in \text{interesting\_nodes}_A(\epsilon)$. This requires us to consider all possible ways the sets $\text{interesting\_nodes}_A(\epsilon)$, $\text{interesting\_nodes}(B)$ and $\text{interesting\_nodes}(A)$ can overlap. Since $\text{interesting\_nodes}(A) \subseteq \text{interesting\_nodes}(B)$ there are only nine different ways for the sets to overlap, as shown in Figure 8.

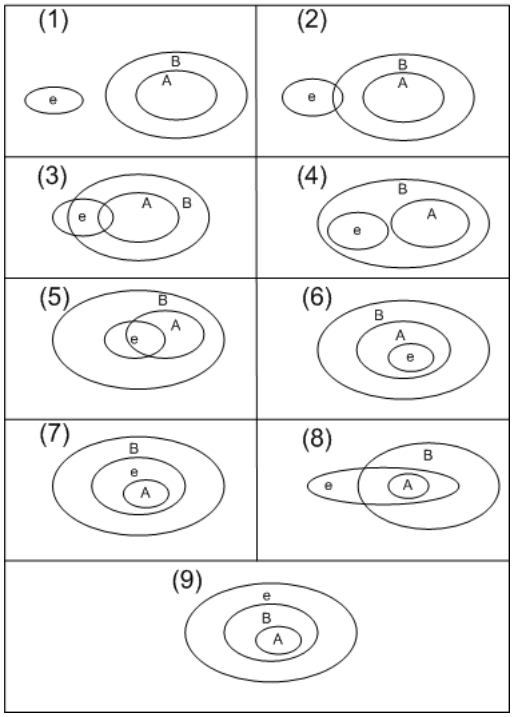

Figure 8: Possible ways for the sets $\text{interesting\_nodes}_A(\epsilon)$, $\text{interesting\_nodes}(B)$ and $\text{interesting\_nodes}(A)$ to overlap. For simplicity, in the figure we use A to refer to $\text{interesting\_nodes}(A)$, B to $\text{interesting\_nodes}(B)$ and e to $\text{interesting\_nodes}_A(\epsilon)$

Consider the running example of Figure 9 and suppose $A = \{4, 10\}$ and $B = \{4, 10, 7\}$. Then $\text{interesting\_nodes}(A) = \{4, 10, 11, 14, 15\}$ and $\text{interesting\_nodes}(B) = \{4, 10, 11, 14, 15, 7, 12, 13\}$. Also, $\text{interesting\_nodes}(7) = \{7, 12, 13, 15\}$.

It is convenient for the proof to organize the nine different ways for the sets to overlap from Figure 8 into the following four cases.

(a) (case (1) of Figure 8):

$\text{interesting\_nodes}_A(\epsilon) \cap \text{interesting\_nodes}(A) = \emptyset$

In our example of Figure 9 this happens if for instance $e = \{9\}$.
In this case, we have $|\text{path}_B(x)| = |\text{path}_A(x)|$, and $|\text{path}_{B\cup\{\epsilon\}}(x)| = |\text{path}_{A\cup\{\epsilon\}}(x)|, \forall x \in \text{interesting\_nodes}(\epsilon)$. Thus,

$$\sum_{x \in \text{interesting\_nodes}_A(\epsilon)} (|\text{path}_B(x)| - |\text{path}_{B\cup\{\epsilon\}}(x)|) = \sum_{x \in \text{interesting\_nodes}_A(\epsilon)} (|\text{path}_A(x)| - |\text{path}_{A\cup\{\epsilon\}}(x)|)$$

and using Equation 8 we get $u_p(B \cup \{\epsilon\}) - u_p(B) \leq u_p(A \cup \{\epsilon\}) - u_p(A)$.

(b) (cases (2) and (4) of Figure 8):
$\exists\, y \in B$ such that:

$\text{interesting\_nodes}_A(\epsilon) \cap \text{interesting\_nodes}(y) \neq \emptyset$

and $\forall x \in A$

$\text{interesting\_nodes}_A(\epsilon) \cap \text{interesting\_nodes}(A) = \emptyset$

In our example of Figure 9, this can happen if $e = \{13\}$.
Let C be the set of all $y \in B \backslash A$ with $\text{interesting\_nodes}(\epsilon) \cap \text{interesting\_nodes}(y) \neq \emptyset$.
Then, $\forall x \in \text{interesting\_nodes}(\epsilon)$, $|\text{path}_B(x)| < |\text{path}_A(x)|$ since $\text{path}_B(x)$ contains at least a node in C for all $x \in \text{interesting\_nodes}(\epsilon)$.
Also, $\forall x \in \text{interesting\_nodes}(\epsilon)$, $|\text{path}_{B\cup\{\epsilon\}}(x)| \leq |\text{path}_{A\cup\{\epsilon\}}(x)|$. The strict inequality holds when $\text{path}_B(x)$ contains a node in C and the equality holds otherwise.
Thus, $u_p(B \cup \{\epsilon\}) - u_p(B) \leq u_p(A \cup \{\epsilon\}) - u_p(A)$.

(c) (case (6) of Figure 8):
$\exists\, x \in A$ such that:

$\text{interesting\_nodes}_A(\epsilon) \cap \text{interesting\_nodes}(x) \neq \emptyset$

and $\forall y \in B \setminus A$:

$\text{interesting\_nodes}_A(\epsilon) \cap \text{interesting\_nodes}(y) = \emptyset$

In the running example of Figure 9, if $e = \{11\}$ then $\text{interesting\_nodes}(4)$ includes node 11 but node 11 is not in $\text{interesting\_nodes}(B \setminus A) = \text{interesting\_nodes}(7)$.
In this case, we have $|\text{path}_B(x)| = |\text{path}_A(x)|$, and $|\text{path}_{B\cup\{\epsilon\}}(x)| = |\text{path}_{A\cup\{\epsilon\}}(x)|, \forall x \in \text{interesting\_nodes}(\epsilon)$. Thus,

$$\sum_{x \in \text{interesting\_nodes}_A(\epsilon)} (|\text{path}_B(x)| - |\text{path}_{B\cup\{\epsilon\}}(x)|) = \sum_{x \in \text{interesting\_nodes}_A(\epsilon)} (|\text{path}_A(x)| - |\text{path}_{A\cup\{\epsilon\}}(x)|)$$

and using Equation 8 we get $u_p(B \cup \{\epsilon\}) - u_p(B) \leq u_p(A \cup \{\epsilon\}) - u_p(A)$.

(d) (cases (3), (5), (7), (8) and (9) of Figure 8):
$\exists\, x \in A$ such that:

$\text{interesting\_nodes}_A(\epsilon) \cap \text{interesting\_nodes}(x) \neq \emptyset$

and $\exists\, y \in B \setminus A$ such that:

$\text{interesting\_nodes}_A(\epsilon) \cap \text{interesting\_nodes}(y) \neq \emptyset$

In the running example of Figure 9, if $e = \{15\}$ then interesting_nodes(4) includes node 15 and in addition node 15 is also in interesting_nodes$(B\backslash A)$ = interesting_nodes(7) because of the black edge $(12, 15)$.

Let C be the set of all $y \in B\backslash A$ with interesting_nodes$(\epsilon) \cap$ interesting_nodes$(y) \neq \emptyset$.

Then, $\forall x \in$ interesting_nodes(ϵ), $|\text{path}_B(x)| < |\text{path}_A(x)|$ since $\text{path}_B(x)$ contains at least a node in C for all $x \in$ interesting_nodes(ϵ).

Also, $\forall x \in$ interesting_nodes(ϵ), $|\text{path}_{B\cup\{\epsilon\}}(x)| \leq |\text{path}_{A\cup\{\epsilon\}}(x)|$. The strict inequality holds when $\text{path}_B(x)$ contains a node in C and the equality holds otherwise.

Thus, $u_p(B \cup \{\epsilon\}) - u_p(B) \leq u_p(A \cup \{\epsilon\}) - u_p(A)$.

This conludes the proof. □

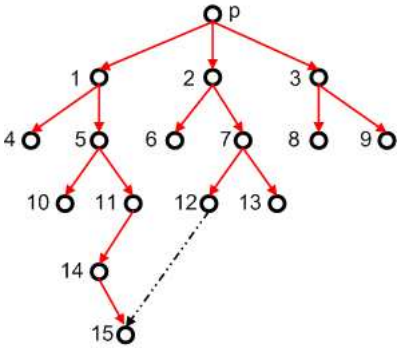

Figure 9: Running example for submodularity proof.

D. WORST CASE GRAPHS FOR GREEDY

In this section, we constuct worst-case graphs for the Greedy algorithm. These graphs force Greedy to give solutions that are $1 - \left(\frac{k-1}{k}\right)^k$ times away from the optimal solution.

D.1 k=2

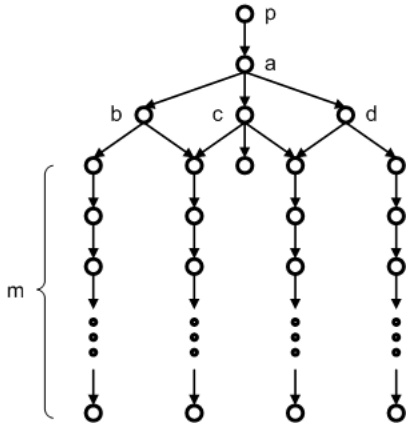

Figure 10: A worst case graph for Greedy

Let us begin with a wost-case graph for $k = 2$. The graph of Figure 10 is such a graph. The ratio of Greedy/Optimal for $k = 2$ and large m becomes 3/4, but no worse. To see why, observe that nodes b and d are the roots of similar trees while c is the root of a tree that shares half of the b-rooted tree, half of the d-rooted tree and another node. This extra node forces the first choice of the Greedy to be node c and then Greedy selects either b or d. However, the optimal solution is to select the nodes b, d.

To compute the optimality ratio, we first compute the average distance of p to every node:

$$\begin{aligned} u(p,G) &= 2\cdot[1+2+\cdots+(m+2)]+2+ \\ &\quad [2+3+\cdots+(m+2)] \\ &\quad [3+4+\cdots+(m+2)] \\ &= 2\frac{(m+2)(m+3)}{2}+2+\frac{(m+1)(m+4)}{2}+ \\ &\quad \frac{m(m+5)}{2} \\ &= 2m^2+10m+10 \end{aligned}$$

Suppose now we add links from p to nodes b and d. The new average distance for p becomes:

$$\begin{aligned} u(p,G_{\{b\}\cup\{d\}}) &= 2\cdot[1+2+\cdots+(m+1)]+6+ \\ &\quad 2\cdot[2+3+\cdots+(m+1)] \\ &= 2\frac{(m+1)(m+2)}{2}+6+2\cdot\frac{m(m+3)}{2}= \\ &= 2m^2+6m+8 \end{aligned}$$

However, the optimal solution is to add links from p to c

and d. In this case, the average distance for p becomes:

$$\begin{aligned} u(p, G_{\{c\}\cup\{d\}}) &= [1+2+\cdots+(m+2)] + \\ &\quad 2\cdot[1+2+\cdots+(m+1)] + \\ &\quad 2+[2+3+\cdots+(m+1)] \\ &= \frac{(m+2)(m+3)}{2} + 2\cdot\frac{(m+1)(m+2)}{2} + \\ &\quad 2+\frac{m(m+3)}{2} \\ &= 2m^2+7m+7 \end{aligned}$$

So, the optimality ratio r for Greedy is:

$$r = \frac{u(p,G) - u(p, G_{\{c\}\cup\{d\}})}{u(p,G) - u(p, G_{\{b\}\cup\{d\}})} = \frac{3m+3}{4m+2} \tag{9}$$

For (not so) large m, the ratio r becomes close to $3/4$. For instance, if $m = 10$ we have $r = 0.79$ and for $m = 100$ we have $r = 0.754$.

D.2 k>2

The construction of the worst-case graph for arbitrary k is similar to the $k = 2$ case. We start with the node p linking to a node a as in Figure 10. Node a links to k nodes $B = \{b_1, b_2, \ldots, b_k\}$, each of which is the root of k identical chain subtrees with m nodes. The set B will be the optimal solution.

We now add $k-1$ new links from the node a to new nodes $C = \{c_1, c_2, \ldots, c_{k-1}\}$. The nodes in C along with one of the nodes in B will be the solution that Greedy will give. For this to happen, node c_1 links to the first chain from the subtree of each $b \in B$ plus to a new node d_1. The next node c_2 links to the second chain from the subtree of each $b \in B$ plus to a new node d_2. We do the same for the remaining nodes of C, i.e. for $i = 1$ to k, node c_i links to the i-th chain from the subtree of each $b \in B$ plus to a new node d_i.

We can show that the approximation ratio of Greedy in the graph we just constructed is $1 - \left(\frac{k-1}{k}\right)^k$.