Generalized Link Suggestions via Web Site Clustering

Jangwon Seo†, Fernando Diaz‡, Evgeniy Gabrilovich‡, Vanja Josifovski‡, Bo Pang‡
†University of Massachusetts Amherst, 140 Governors Drive, Amherst, MA 01003
‡Yahoo! Research, 4301 Great America Parkway, Santa Clara, CA 95054
†jangwon@cs.umass.edu, ‡{diazf, gabr, vanjaj, bopang}@yahoo-inc.com

ABSTRACT

Proactive link suggestion leads to improved user experience by allowing users to reach relevant information with fewer clicks, fewer pages to read, or simply faster because the right pages are prefetched just in time. In this paper we tackle two new scenarios for link suggestion, which were not covered in prior work owing to scarcity of historical browsing data. In the web search scenario, we propose a method for generating quick links—additional entry points into Web sites, which are shown for top search results for navigational queries—for tail sites, for which little browsing statistics is available. Beyond Web search, we also propose a method for link suggestion in general web browsing, effectively anticipating the next link to be followed by the user. Our approach performs clustering of Web sites in order to aggregate information across multiple sites, and enables relevant link suggestion for virtually any site, including tail sites and brand new sites for which little historical data is available. Empirical evaluation confirms the validity of our method using editorially labeled data as well as real-life search and browsing data from a major US search engine.

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

General Terms: Algorithms, Experimentation, Measurement

Keywords: Quick links, Tail sites, Link suggestion, Website clustering, Assisted browsing

1. INTRODUCTION

Finding information on the web often amounts to finding the right URL. Proactively suggesting links that are relevant to users' current information needs is therefore likely to lead to higher user satisfaction and allow the users to accomplish their goals faster. In this paper we propose two novel link suggestion approaches for the two main ways to find information online, namely, web search and browsing.

In response to a navigational query [3], search engines strive to provide the URL to which the user likely wants to navigate. However, many navigational queries still have some amount of ambiguity. For example, when submitting the query "P.F. Chang's" (a chain of Chinese restaurants in the U.S.), the user may be interested in finding the closest

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

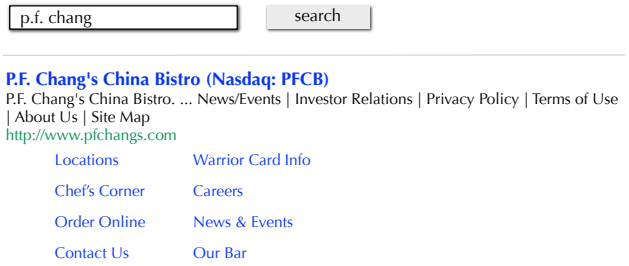


Figure 1: Example of link suggestions in a search result page for the navigational query 'P.F. Chang'.

restaurant, checking the menu, booking a table, or ordering a take-away. The search engine cannot possibly determine the right alternative given the very short query. What it can do, however, is to surface direct links to the most likely of these options, by showing them beneath the main URL, www.pfchangs.com (cf. Figure 1). These suggested links are known as quick links, and today all major search engines offer this functionality.

Link suggestion was also found to be useful in many scenarios beyond web search, including web page pre-fetching [14] and site navigation [20]. As a result, link suggestion systems have received considerable attention in a variety of commercial systems, from major search engines to companies distributing pre-fetching technology[1].

Previous work on link suggestion mainly focused on exploiting logged user browsing behavior to achieve strong performance. Quick link suggestion uses clicks logged in toolbar data to determine relevance [5]. Similarly, pre-fetching systems often use site-level access logs to suggest links for pre-fetching [14]. Although these techniques are adequate for sites with sufficient traffic, performance can suffer when such data is scarce or does not exist at all. For example, quick links are available for popular restaurant chains such as "P.F. Chang's", but not available for "Tarzana Armenian Deli". Unfortunately, sufficient traffic is often a luxury possessed by a relatively small number of sites; low traffic is the norm.

To address this limitation, we broaden the scope of link suggestion techniques beyond traffic-based solutions. In the context of quick links, we extend traffic-based models to include non-traffic signals based on page and site layout. We also cluster sites to leverage similarities between categories

[1] `http://en.wikipedia.org/wiki/Web_accelerator`

of sites. For example, restaurant web sites should include a 'menu' quick link. Together, our techniques allow us to perform quick link suggestion for a much larger set of sites; in principle, we can provide a quick link to virtually any page.

We also introduce a task similar to pre-fetching, which we call *dynamic quick links*. While the use of static quick links is limited to Web search, in dynamic quicklinks, we condition the recommendation of quick links on the page the user is currently reading. One way to present dynamic quick links is by providing a tool (e.g., a pop-up window) suggesting links to browse next. Traffic-based models similar to those used in pre-fetching can be used for dynamic quick link suggestion. We also exploit both site-level and link-level clustering to improve performance.

The contributions of this paper are threefold. First, we extend the existing quick links paradigm and propose an approach to compute quick links for any web site, including tail sites as well as brand new sites. Second, we formulate the notion of dynamic quick links, and propose an approach for assisted browsing, effectively anticipating which link the user will choose next from any given page. In both of this scenarios, we propose methods that are applicable to any site, including those for which little or no historical browsing data is available. This becomes possible due to our use of non-traffic-based data based on page and site structure, as well as clustering of Web sites to address data sparsity. Finally, our experimental evaluation confirms the validity of our method using editorially labeled data as well as real-life search and browsing data from a major US search engine.

2. PROBLEM DEFINITION

The **static quick links task** refers to the problem of selecting and ranking links for a user entering a web site. It is defined by a set of sites, $\mathcal{S}$. Each site, $s \in \mathcal{S}$, has a set of *candidate quick links*, $\mathcal{U}(s)$. In our work, $\mathcal{U}(s)$ consists of all the links contained on the web site's homepage p. For each $u \in \mathcal{U}(s)$, there is an unobserved binary relevance $r_s(u) \in \{0, 1\}$. Given $s \in \mathcal{S}$, we would like to select and rank a set of k urls from $\mathcal{U}(s)$ so as to maximize the relevance of this set. We remain agnostic about performance measures of relevance, studying several in our experiments.

The **dynamic quick links task** refers to conditioning our selection and ranking of k urls on the url $u' \in \mathcal{U}(s)$ that the user is currently browsing.

A key question in developing link suggestion algorithms is whether to make them query dependent or not. Choosing the query dependent route is beneficial for Web search, as it uses additional information contained in the query. Yet this also comes at a cost, as this approach increases the amount of computation to be done for each query, a critical consideration for search engines handling hundreds of millions of queries each day. Query independent approaches are also more general, as they can equally apply to browsing scenarios, where no explicit query is available. For the sake of uniformity and computational efficiency, in this paper we opted to focus on query independent approaches.

3. ALGORITHMS

3.1 Static Quick Links

We adopt a machine learning approach to address the static quick links task. In general, machine learning approaches learn a relationship between task instances and a desired target value. In our situation, each $u \in \mathcal{U}(s)$ is an instance and the desired target value is its relevance, $r_s(u)$. In order to generalize between instances, machine learning approaches compute abstract features of each instance and learn the relationship between these features. Learning is performed by using a small set of training instances which have labeled target values; in our case, we assume access to a small set of sites $\mathcal{S}^t \subset \mathcal{S}$ whose URLs have the relevance values, $r_s(u)$, provided. Such an approach requires that we define two things: how to compute instance features and how to model the relationship to the target.

3.1.1 Features

We followed two principles when designing features. First, obviously we would like features of u which correlate with $r_s(u)$. Second, because we are interested in strong performance on both head and tail sites, we would like features well-represented in both head and tail sites. For example, restricting ourselves to click-based features results in poor representations for tail sites. We refer to features well-represented in both head and tail sites as *common features*, and features better represented in head sites (tending to be sparse in tail sites) as *head features*. It is worth noting that head features carry critical information about how popular links are or how often links are used [5].

For each $u \in \mathcal{U}(s)$, we generate three sets of common features. *URL-based features* are extracted from a URL address of u, for example, the depth of the URL path, the type of URL file extensions (e.g., html, jpg, php), etc. *Anchor text-based features* are extracted from anchor text used for u in the homepage p of that site, for example, how many named entities are in anchor text w, how many nouns or verbs are in anchor text, etc. Notice that these are functions of the text not term features often used in the information retrieval and text classification literature; we do this to provide generalization across different types of sites. *DOM block-based features* are extracted from the Document Object Model (DOM) block b (of homepage p) that u belongs to; for example, the ratio of bytes of text to the number of links in b, the position of b in DOM block order, etc.

For each candidate quick link, we generate two sets of head features. *Link structure-based features* are extracted from hyper-link structures of the whole Web graph, for example, the number of incoming links to u. *User behavior-based features* are extracted from user behavior data such as toolbar logs, for example, the number of visits to u over a certain period of time. We expect head features to be extremely sparse or nonexistent for tail sites.

In subsequent sections, we will refer to the features of u as ϕ_u^s.

3.1.2 Modeling Relevance

We would like to model the relationship between a candidate quick link's features, ϕ_u^s and its relevance, $r_s(u)$. To accomplish this, we cast our task as a regression problem. That is, we would like to learn a function h whose domain is the site and candidate quick link and whose range is rel-

evance. We measure the training set error of h as,

$$\mathcal{E}(h, \mathcal{S}^t) = \sum_{s \in \mathcal{S}^t} \sum_{u \in \mathcal{U}(s)} (h(s,u) - r_s(u))^2 \quad (1)$$

The general regression problem is to select a function $\tilde{h}$ such that,

$$\tilde{h} = \text{argmin}_{h \in \mathcal{H}} \mathcal{E}(h, \mathcal{S}^t) \quad (2)$$

The hypothesis space, $\mathcal{H}$, is the set of possible functions which fit a particular functional form. In order to perform learning, we need to define $\mathcal{H}$ and describe how we search it.

In our work, each $h \in \mathcal{H}$ is defined as a decision tree forest [16] composed of m trees such that,

$$h(s,u) = \lambda_0 f^0(\phi_u^s) + \cdots + \lambda_m f^m(\phi_u^s) \quad (3)$$

where f^i is a regression tree, ϕ_u^s represents the features generated for candidate u of site s, and λ_i is a parameter controlling the contribution of f_i to the prediction. Regression trees are appropriate for our task because they can address both numerical and categorical features and have been shown to be highly effective for ranking tasks [28].

Because finding the exact solution to Equation 2 for our hypothesis space is NP-Complete, we apply Friedman's Gradient Boosted Decision Tree (GBDT) algorithm to search the space [16]. The GBDT algorithm searches $\mathcal{H}$ using a boosting approach. GBDT begins with an initial function f^0 that is usually an average value of labels of all training samples. The subsequent trees, f^i, iteratively minimize the L_2 loss with respect to the residuals of the predicted values and the target values. Each weight, w_i is a monotonically decreasing function of i, parametrized by a value, η, referred to as the learning rate. In addition to η, the model has two other parameters: the number of trees and the number of nodes per tree.

We induce a ranking of quick links $\mathcal{U}(s)$ by computing $\tilde{h}(s,u)$ for each $u \in \mathcal{U}(s)$ and ranking by the prediction.

3.1.3 Class-Specific Modeling

In the previous section, we performed ranking of quick links for each site separately, and no information was shared between similar sites. We now turn to exploring the similarities between different sites.

Consider two sites s and s', both of which are restaurants. We know that, for sites of the class 'restaurant', quick link candidates with anchor or URL text containing the term 'menu' should receive the same relevance. That is, given two quick link candidates from sites in the same class, similar candidates will have similar relevance.

In order to exploit site classes, we need to classify sites in the first place. We accomplish this by clustering sites using a term-based representation. Let w_s be the $|\mathcal{V}| \times 1$ term vector for site s; terms are extracted from anchor text and URL paths of links in pages. Sites are then clustered using the diffusion wavelet approach introduced in [26]. This method works by constructing a term-term co-occurrence matrix from the bag-of-word representations of sites, i.e., by $T^\mathsf{T}T$ where T is the $|\mathcal{S}| \times |\mathcal{V}|$ 'collection' matrix. Then, using the diffusion wavelet algorithm [9], we obtain wavelet 'topic bases'. Each topic basis, ϕ^i, is a $|\mathcal{V}| \times 1$ vector capturing the behavior of terms in a particular class. We assign a site s to the class determined by $\text{argmax}_i \langle \phi_i, w_s \rangle$. The advantage of this approach is that it does not require that a fixed number of clusters be specified in advance. However, other clustering approaches can be applied as well. This provides a partitioning of sites, allowing us to learn class-specific models.

For each class $c \in \mathcal{C}$, we are interested in training a class-specific model, h_c, which leverages similarities between sites. In order to train a class-specific model, we adopt the Tree-based Domain Adaptation (TRADA) algorithm [7]. TRADA begins by training a generic model as in Section 3.1.2. The algorithm then modifies the generic model to minimize the loss function with respect to target values in a target domain; in our case, this target domain is a class of sites. That is, we are going to minimize our loss function, constraining ourselves to those instances in class c,

$$\tilde{h}_c = \text{argmin}_{h_c \in \mathcal{H}_c} \mathcal{E}(h_c, \mathcal{S}_c^t) \quad (4)$$

where $\mathcal{S}_c^t$ is a set of relevance-labeled sites of class c.

Equation 4 is equivalent to Equation 2 except for the set of training instances and the hypothesis space. Defining $\mathcal{H}_c$ becomes the critical part of this technique. As mentioned earlier, our algorithm needs to use features of quick link candidates that allow similar candidates to receive similar predictions. Unfortunately, neither the common features nor the head features capture the semantic similarity of pairs of candidates. We manage this by using term features. The idea here is that while common and head features provide evidence for quick link relevance in general (e.g., 'highly visited candidates are relevant'), term features provide class-specific evidence (e.g., 'for restaurants, candidates whose url contains 'menu' are relevant'). As a result, we define $\mathcal{H}_c$ such that

$$h_c(s,u) = \tilde{h}(s,u) + \lambda_0 f_c^0(w_u) + \cdots + \lambda_{m'} f_c^{m'}(w_u) \quad (5)$$

where $\tilde{h}$ is the generic model approximated in Section 3.1.2, which is fixed for all $h_c \in \mathcal{H}_c$, and w_u represents the bag of words associated with candidate u. Except for the addition of $\tilde{h}(s,u)$, Equation 5 is identical to Equation 3 and, as a result, TRADA searches $\mathcal{H}_c$ using the boosting approach described in Section 3.1.2.

Because Equation 5 uses fairly sparse term features, we need a substantial amount of training data for each class. If the number of classes is large, then collecting many manual labels for sites in every cluster can be expensive. In order to gather sufficient training data, we bootstrap by automatically labeling unlabeled sites and quick link candidates. That is, for each cluster, we first use the generic model to predict relevance scores of links in unlabeled sites. Next, we assign pseudo-labels to the links, e.g., links in the top 30% are relevant while links in the bottom 30% are non-relevant. Finally, we apply the TRADA algorithm with these pseudo-labels. The main advantage of this approach is that we can cheaply leverage an enormous number of homepages on the Web. This approach demonstrated strong performance in the context of vertical selection [1].

3.2 Dynamic Quick Links

The dynamic quick links task allows us to adjust the ranking of links depending on the context of the user. In this case, context is defined by the current page $u \in \mathcal{U}(s)$. Consider a user reading the 'menu' page of a restaurant. If we have observed many other visitors navigating to the 'directions' page immediately after reading this page, then there

is evidence supporting the relevance of the 'directions' page in this context.

In order to address the sparsity of browsing data for tail sites, we leverage information from semantically related quick link candidates. To accomplish this, we cluster quick link candidates within our site classes $\mathcal{C}$. Our quick link clustering algorithm will use term-based representations, resulting in clustering links with related text (e.g., 'directions' and 'location'). Specifically, we use anchor text and words in URL paths.

Although we could perform an unsupervised clustering method as in Section 3.1.3, here we have unique data that we can exploit to direct the clustering. We hypothesize that, given two sites in the same site class, two links are semantically similar if they share a similar number of visits. So, given two arbitrary restaurants, the two 'menu' quick links should receive comparable numbers of visits. In practice we normalize the number of visits by the number of site visits in general so that we can compare links from head and torso sites.

Our representation is term-based, our supervision is a real valued number (as explained above), and our clustering method is supervised Latent Dirichlet allocation (LDA) [2]. Supervised LDA projects each training instance into a k-dimensional 'topic space', represented as a multinomial distribution over topics; that is, for each u, we have a distribution $p(c|u)$ over all $c \in \mathcal{C}$.

Once we have abstract representations of links, we investigate browsing behaviors between links or the representations. We make a Markov assumption about link transition: the class of the next link to be browsed depends only on the class of the current link. Assume $\mathcal{B}$ is our browsing data encoded as url transitions. We compute the empirical distribution of transition probabilities from quick link class c_i to c_j as,

$$\mathbf{P}_{ij} = \frac{\sum_{u \to u' \in \mathcal{B}} p(c_i|u)p(c_j|u')}{\sum_{c_k} \sum_{u \to u' \in \mathcal{B}} p(c_i|u)p(c_k|u')}$$

While the estimated random walk matrix represents only one step of browsing, it may be beneficial to encode multiple steps of browsing. This is because some users may prefer shortcuts from one link to another link that is several hops away instead of going through several intermediate links that most users follow. To model this, we construct a new random walk matrix as follows:

$$\mathbf{R} = \frac{1}{\mathcal{Z}} \sum_{a=1}^{T} \gamma^{a-1} \mathbf{P}^a$$

where $\mathcal{Z}$ is a normalization factor, γ is a shrinkage parameter, and T is the maximum number of steps to consider.

Given this quick link transition matrix, we are now going to rank the quick link candidates. Assume $z_u = [p(c_0|u), p(c_1|u), \cdots, p(c_{k-1}|u)]$ is the $k \times 1$ topic vector of the current url the user is reading. We first compute scores for the possible classes of the next quick link as,

$$\tilde{z}_u = \mathbf{R}^\mathsf{T} z_u$$

To find links relevant to u, we compute cosine similarity between $\tilde{z}_u$ and topic vectors of each $v \in \mathcal{U}(s)$. Because this similarity captures only textual properties of quick link candidates, we combine the cosine similarity with the GBDT prediction, which is based on additional types of features (cf. Sections 3.1.1 and 3.1.2):

$$f(s, u, v) = \tau h(s, v) + (1 - \tau)\langle \tilde{z}_u, z_v \rangle$$

where τ is a parameter. Candidate links are then ranked by $f(s, u, v)$.

4. EXPERIMENTAL SETUP

4.1 Data

We constructed two sets of homepages: a collection of pages from popular Web sites with rich user traffic information from search logs or toolbar data (*head set*), as well as a collection of less visited pages (*tail set*). Note that in our experiments, candidates for link suggestion in a given site is restricted to links available in its homepage, thus we often use the words homepage and site interchangeably. For the head set, we randomly sampled 5,153 sites among popular web sites for which (a) Yahoo! Search was providing quick links on search result pages, and (b) there was sufficient user traffic information from toolbar logs and click logs. For the tail set, our goal was to construct a collection of homepages that were not as heavily visited. We were not aware of an existing index of all homepages online from which to sample. Thus, we constructed our dataset by extracting the most relevant search results for navigational queries. We first sampled 100,000 million queries from query logs of Yahoo! Web Search and applied an internal navigational query classifier trained on manually labeled examples. For more details on the classifier, see [10]. For each query classified as a navigational query, we identified its most clicked url. We refer to such urls as homepages. We then restricted to those homepages that were not heavily visited (less than 1,000 clicks recorded in a month-worth of click logs), but have more than 10 outgoing-links (otherwise link suggestions are not very usuful). We collected a total of 14,332 homepages from tail sites.

Simple statistics over the head and tail sets are shown in Table 1. Note that typically there are a large number of candidate links from the homepage of a given site: on average a head site has 173 links on its homepage.

Table 1: Statistics of head and tail sets

	#sites			Avg. #links
	Labeled	Unlabeled	Total	per site
Head set	786	4367	5153	173.72
Tail set	507	13825	14332	50.76

Dynamic quicklinks experiments used the same dataset and split as static quicklinks. We considered the bigger set as the training data — note that the original labels were not relevant here. We took one month of Yahoo! toolbar logs and extracted all browsing sessions which included the relevant homepages. We segmented the sessions according to rules similar to those used in [5]. For example, a session is terminated if any two consecutive clicks are longer than 10 minute apart or a Back button is clicked. URLs that were not in the candidate set were removed from the sessions.

4.2 Evaluation

In principle, performance of link suggestions on head sites can be evaluated using actual user traffic information. But since we do not have sufficient user traffic information for

tail sites, we randomly selected a sample of sites from both collections for manual annotations, where editors were asked to select up to 10 useful destinations for each site among all links in its homepage. The selected links were considered to be relevant, the rest were considered to be non-relevant.

In terms of metrics, we considered four standard evaluation metrics in information retrieval: precision, recall, F_1 score (the harmonic mean of precision and recall), and mean average precision (MAP). Following previous work [5], we assumed 8 links will be provided as quick links for a homepage on the search result page. Accordingly, we used 8 as the cut-off point for all metrics and report precision, recall, F1 and MAP at 8 (P@8, R@8, F1@8 and MAP@8, respectively). We performed the permutation test and considered an improvement statistically significant if p-value < 0.05.

In order to evaluate our class-based model, we focused our experiments on clusters that were large enough to have effective smoothing among pages in that cluster. Out of the 49 clusters in our dataset, the top 10 clusters cover more than 60%, each with over 314 sites. Note that in a real system, the number of considered sites could be much larger and the number of reasonably-sized clusters might increase as well.

Since it would be more difficult to obtain manual labels for the dynamic link suggestion task, evaluation used traffic information that was available for sites in the head set. We predict the next link to be visited given the current link in each user session in the test data. Given the motivation for this task, we evaluated the quality of the top 4 predicted link in each case. Let $\mathcal{Q}_u^4 \subset \mathcal{U}(s) - u$ be the set of predicted links. If $\mathcal{B}^*$ is our observed browsing data, then define our browsing metric as,

$$B_4 = \frac{\sum_{u \to v \in \mathcal{B}^*} \mathcal{I}(v \in \mathcal{Q}_u^4)}{|\mathcal{B}^*|}$$

where $\mathcal{I}$ is the indicator function.

4.3 Runs

We consider three baseline systems for the static quicklinks task. Two baselines represent simple ways of estimating the "popularity" of a candidate link. One estimate is derived from the Web graph and ranks candidate quicklinks according to the number of incoming links from the entire web. We refer to this baseline as # INLINKS. Another estimate of popularity can explicitly use visitation data. In this case, candidate quicklinks are ranked according to the number of visits registered in one month of Yahoo! Toolbar logs. We refer to this baseline as # VISITS. A much stronger comparison point is the state-of-the-art technique recently introduced by Chakrabarti et al. [5] (see Section 7 for more details). This greedy algorithm was shown to outperform other methods such as ranking by the number of visits recorded toolbar logs. We denote this system as GREEDY.

We tested three versions of our static quicklinks models. The basic GBDT model was trained with two sets of features. Runs labeled GBDT-C use only common features, while runs labeled GBDT-HC use both common and head features.

As a baseline for adaptive modeling, we consider an adaptive model which uses additional pseudolabeled data and features *without* clustering sites. That is, we performed the adaptation algorithm described in Section 3.1.3, but treated the entire dataset as one cluster. This baseline is labeled ADAPT-ALL. Runs labeled ADAPT-CLS build class-specific models with the additional pseudolabeled data and features.

4.4 Training

For GBDT models, the shrinkage factor was set to 0.05, and other parameters were chosen by 10-fold cross validation.

Our class-based model required additional parameters. Recall from Section 3.1.3, once we clustered homepages into different classes, the original GBDT-C model is applied to each cluster in the tail unlabeled set to obtain pseudo-labels. 10% of the homepages in the unlabeled tail set were set aside as the held-out set to estimate the two parameters for new regression trees, i.e., the number of trees and the number of nodes. They were chosen to minimize the loss function with respect to the pseudo-labels on the held-out set. Tree adaptation using the TRADA algorithm was performed on the rest 90% of homepages in the tail unlabeled set.

For dynamic quicklinks, our random walk parameters, i.e. γ and τ were tuned on the training data except for T which we set to 3. Topic models for clusters are learned using the collapsed Gibbs sampler [18]. We fix free parameters as follows: the number of topics $K = 20$, the Dirichlet hyperparameter $\alpha = 0.01$ and the variance of the response variables $\sigma^2 = 0.09$.

5. RESULTS

5.1 Static link suggestions

We evaluated the performance of our algorithms separately on head and tail sites. On head sites, we compare the proposed technique with the three baseline systems (# INLINKS, # VISITS, GREEDY). Since some of these systems relied heavily on head features, we expect them to be strong baselines. Because these baselines rely on features that are scarce in tail sites, we compare performance to GBDT-HC which, as we will see, can be considered a strong baseline.

Table 2(a) summarizes performance on head sites. The two simple baselines, # INLINKS and # VISITS, perform better than random but significantly underperform GREEDY. However, GBDT-HC, which uses both head and common features, yielded the best performance across all four metrics. Interestingly, the GBDT model using only common features (GBDT-C) performed comparably to GREEDY.

Table 2(b) summarizes performance on tail sites. The GBDT models used to obtain these results were trained on the labeled head set. Consequently, these results show that our model can generalize from head sites to tail sites. We also performed 10 fold cross-validation on the tail labeled set and observed almost identical trends.

5.2 Effect of class-based adaptation

We ran separate experiments to evaluate the performance of our class-based adaptation algorithm. These results are presented in Table 3. ADAPT-CLS outperforms GBDT-C for eight out of ten clusters. In order to confirm that our improvements were not the result of merely adding term-based features, we compared performance to ADAPT-ALL, a model which incorporates the additional term-based features without site clustering. As we can see from Table 3, ADAPT-ALL did not yield any improvement over GBDT-C. In fact, aggregated over these ten clusters, ADAPT-CLS outperforms both GBDT-C and ADAPT-ALL across all four metrics, sta-

Table 2: Comparison of quick link selection techniques on (a) 786 head sites and (b) 507 tail sites. A † indicates a statistically significant differences from both of two weak baselines, # INLINKS and # VISITS while a ‡ indicates a statistically significant differnece from a strong basline, GREEDY.

(a) head sites

Method	P@8	R@8	F1@8	MAP@8
# INLINKS	0.2221	0.2799	0.2420	0.1362
# VISITS	0.2875	0.3560	0.3113	0.1837
GREEDY	$0.3580^{\dagger}$	$0.4233^{\dagger}$	$0.3806^{\dagger}$	$0.2646^{\dagger}$
GBDT-C	$0.3477^{\dagger}$	$0.4275^{\dagger}$	$0.3763^{\dagger}$	$0.2890^{\dagger\ddagger}$
GBDT-HC	$0.4361^{\dagger\ddagger}$	$0.5330^{\dagger\ddagger}$	$0.4702^{\dagger\ddagger}$	$0.3915^{\dagger\ddagger}$

(b) tail sites

Method	P@8	R@8	F1@8	MAP@8
GBDT-HC	0.4308	0.4950	0.4277	0.2930
GBDT-C	0.4369	0.4928	0.4328	0.2904

Table 3: MAP@8 of quick link selection for each class (cluster) of sites. A † indicates a statistically significant improvement over both GBDT-C and ADAPT-ALL in aggregation. Performance on other metrics follows the same trends.

Cluster	GBDT-C	ADAPT-ALL	ADAPT-CLS
1	0.2949	0.2829	0.3066
2	0.2259	0.2233	0.2292
3	0.2549	0.2641	0.2640
4	0.3687	0.4000	0.3729
5	0.2271	0.2166	0.2383
6	0.2307	0.2374	0.2296
7	0.4877	0.4344	0.4877
8	0.2552	0.2490	0.2632
9	0.2575	0.2557	0.2610
10	0.3521	0.3860	0.3789
total	0.2775	0.2744	$0.2854^{\dagger}$

tistically significant in three cases. This provides empirical evidence for our hypothesis "similar types of sites have links corresponding to similar functions as good links".

5.3 Dynamic link suggestions

In our dynamic quicklinks task, we adopted the best static quicklink model, GBDT-HC. For each site class c, we trained GBDT-HC with all labeled examples except for those in c and tested on c. We removed the current URL from the candidate set for each test trail.

Experimental results are shown in Table 4. In seven clusters, the dynamic link suggestions outperformed the static link suggestions by large margins. In the other three clusters, the static link suggestions and the dynamic link suggestions showed little difference. In aggregation, the dynamic link suggestions yielded much better performance than the static link suggestions, and the improvements are statistically significant. This demonstrates that modeling user browsing patterns over types of links achieves more accurate link suggestions.

For reference, we also conducted experiments examining the upper-bound of performance over head sites. That is, for sites where we have sufficient user browsing data over individual links, can we reasonably predict where a user is going to go next based on what other users have done in this same site? Maybe there is a huge variance among users in this dynamic scenario, that there is not much room for improvement? To this end, we computed a transition matrix over individual links for each site, counting transitions among

Table 4: Results of link suggestions considering user browsing patterns according to types (clusters) of sites. "static" and "dynamic" represent the best performing model in the previous section, i.e., GBDT-HC and the random walk-based approach, respectively. The numbers are B_4 values. A † indicates statistically significant improvement on "static" in aggregation.

cluster	(#transitions)	static	dynamic
1	(1094)	0.3821	0.3803
2	(4694)	0.5187	0.6027
3	(6827)	0.2659	0.3098
4	(4840)	0.3407	0.3884
5	(3269)	0.3772	0.4035
6	(32522)	0.3421	0.3428
7	(840)	0.4214	0.4024
8	(3449)	0.2157	0.2873
9	(7267)	0.3041	0.3750
10	(2739)	0.2205	0.4045
total	(67541)	0.3344	$0.3682^{\dagger}$

physical links in user trails. The next links were predicted as the most likely outgoing links from the current position. Note that in this case a transition matrix built for a given site is completely not applicable to another site. We performed 10-fold cross-validation for each site in the test set, using browsing patterns learned over one subset of users to predict for unseen visitors to the same site. On average, we obtained an upper-bound of 0.654 in B_4 value. Indeed, on each particular site, there is a reasonable amount of regularity in terms of browsing behavior. This suggests that we still have plenty of room for improvement. Although we should also note that some of this regularity might be site-specific and does not generalize to other sites.

In contrast, our models were built over fairly general representations, specific only to a given site cluster. The models, once learned, can be applied to any unseen sites classified into an existing site cluster, just as we applied our model to the test sites that were unseen in the training phase.

6. DISCUSSION

6.1 Static link suggestion

Our results for static quick links are compelling because they imply that traffic information, while sufficient, is not necessary for strong performance. This result also means that existing traffic-based approaches can be improved with

Table 5: Top 10 features in GBDT models. p: a page; u: a candidate link; b: the DOM block to which u belongs. len(s) is the length of text in s, measured in bytes.

(a) GBDT-HC

Feature	Weight
# of visits to u	100.00
# of u in p	69.14
# of in-coming links to u	57.48
len(u)	47.13
the depth of the path of u	43.11
Are u and p in the same domain?	41.24
len(b) / len(p)	39.44
# of links in b / # of links in p	35.29
# of links in b / len(b)	33.56
# of images in b / # of links in b	31.17

(b) GBDT-C

Feature	Weight
# of u in p	100.00
len(u)	95.98
len(b) / len(p)	60.53
# of links in b / # of links in p	59.92
# of links in b / len(b)	52.42
# of images in b / # of links in b	50.75
Are u and p in the same domain?	49.83
the depth of the path of u	46.15
the position of b in p	41.25
len(anchor text of u)	31.25

content-based features. Furthermore, this result suggests that traffic-based approaches can be replaced entirely by an approach that uses only common features. Consequently, web sites or search engines can present quick links even without having to maintain a Web graph or toolbar data.

It is worth noting that performance numbers on the head set (Table 2(a)) are lower than on the tail (Table 2(b)). However, Table 1 demonstrates that there were fewer candidate links in tail sites compared to head sites, which effectively makes the problem slightly easier.

We were also interested in the importance of the different features in GBDT models. Tables 5(a) and 5(b) show the top 10 most important features of GBDT-HC and GBDT-C, respectively. The feature importance is computed based on how much each feature contributes to the loss reduction [16]. The top features of GBDT-HC and GBDT-C largely overlap, except for two head features (# of visits and # of incoming links to u) available only in GBDT-HC. Among the important common features, those based on page layout or link position dominate. This indicates that homepages indeed tend to be designed with useful links made more salient to users, and that our features can effectively capture such layout-based cues.

In order to explain the effectiveness of common features, we compared the distributions of relevant and non-relevant links for head and tail sites. We expect common features to behave similarly for both head and tail sites while head features should behave very differently. If the distributions are completely different, then it is unlikely we can apply a model learned on the head sites to the tail sites. Figure 2(a) shows the distributions of the top 6 common features. Note that the distributions in tail sites often follow similar contours as those in the head sites. In contrast, Figure 2(b) shows the distributions of the two most important head features in GBDT-HC (recall that all other important features for GBDT-HC were common features). Here the distributions in the tail sites are quite different from those in the head sites, lacking the differentiation between relevant and non-relevant sets. In fact, most head features in the tail set take a zero value due to the absence of user traffic information from which head features were extracted.

6.2 Dynamic link suggestion

Our results for dynamic link suggestion, although only evaluated on head sites, demonstrate the efficacy of clustering links within a site class. Because the lower dimensional representation of a link, z_u, is not dependent on traffic information, we should be able to extend models to tail sites within a cluster.

Nevertheless, performing the tail evaluation for this task requires more work. Editorial data is problematic because of the subjectivity in assessing a contextual suggestion (i.e., editors would have to assume the role of a user reading a certain page). However, there may be some combination of editorial and log data that provides good evaluation.

Despite having focused on quick link suggestion for our experiments, it is worth inspecting the link clusters. Table 6 shows an example of an estimated topic model for a site cluster related to "educational institutions". Each link cluster looks reasonable. For example, links about scholarship or financial aids make one topic (#14), while links about faculty or staff directories make another topic (#10).

In addition to inspecting link clusters, we can inspect dominant transitions between clusters within a class of sites. Figure 3 shows an example of dominant transitions in a random walk matrix estimated from sites related to "sports teams". We can observe some interesting patterns, for example, users visiting links about ballparks subsequently visit links about ticket sale with some probability. Also, links about multimedia clips follow links about fan forums. These transitions look reasonable enough to be easily understood.

7. RELATED WORK

Our work explores two ways of generalizing quick link selection, and proposes link suggestion methods that are applicable to any web site. Several prior studies demonstrated the usefulness of link suggestion. Juvina and Herder [20] used a user study to show that link suggestions help users navigate the web in a structured manner. White et al. [27] showed that different types of suggestions (e.g., query suggestion vs. link suggestion) are better suited for different types of tasks.

Link suggestion has been applied to web site design and organization. Perkowitz and Etzioni [23] addressed automatically generating index pages that contain useful links, reflecting evolving site usage patterns. Srikant and Yang [25] studied the scenario when the real location of a page can be different from where users expect it to be, and presented an algorithm selecting the expected locations. Doerr et al. [12]

Table 6: Topics estimated from a cluster of sites related to "educational institutions" by the supervised LDA model. Terms are stemmed by the Porter stemmer.

Topic	Topic terms
1	email mail login webmail e
2	employ job hr career human
3	school counti org museum scienc
4	download project org test softwar
5	research administr bookstor univers presid
6	student servic center career health
7	gov counti us educ court
8	class schedul cours blackboard regist
9	event new calendar emerg newsroom
10	faculti staff contact directori us
11	academ program degre school graduat
12	admiss appli student prospect undergradu
13	map campu direct visitor tour
14	student aid financi current scholarship
15	us contact about polici privaci
16	life z campu student hous
17	calendar academ event orient registrar
18	librari athlet univers scienc school
19	rss new how feed get
20	alumni give parent friend famili

and Kranakis et al. [21] proposed algorithms for suggesting shortcuts between pages in a web site by analyzing web logs.

Chakrabarti et al. [5] were the first to discuss quick link generation as link suggestion. The authors approached the quick link selection problem as a combinatorial optimization problem, since the space on search engine result pages is limited, and only high value link suggestions should be surfaced there. This study proposes an algorithm that is within a factor of $(1 - 1/e)$ from the optimum.

Link suggestion has been also applied to URL pre-fetching. Duchamp demonstrates that a popularity-based pre-fetching protocol can significantly reduce latency and bandwidth usage [14]. Subsequent work explored more efficient implementations [8], personalization [11], and longer-term modeling [13, 15]. This line of work requires availability of web site access logs, which can be scarce for tail sites or nonexistent for newly created sites.

The main limitation of the previous studies is that their techniques cannot be used for tail sites where there is not enough historical traffic information, such as site access logs or toolbar logs. In this work, we tackle this problem by introducing a feature-based model that can be effective even without such statistics.

Features we define are inspired by web page segmentation and template extraction studies [6, 22, 19, 24, 17, 4]. Structural features have also been proposed in a number of other studies. Lin and Ho [22] and Gupta et al. [19] proposed algorithms to extract content blocks from HTML pages using a DOM (Document Object Model)-based approach and an information theoretic approach, respectively. In the context of template extraction, Gibson et al. [17] studied the nature and prevalence of templates on the Web, introducing a randomized template extraction algorithm. Chakrabarti et al. [4] formulated smoothing of a classifier for scoring DOM blocks and showed that their approach is effective for tasks

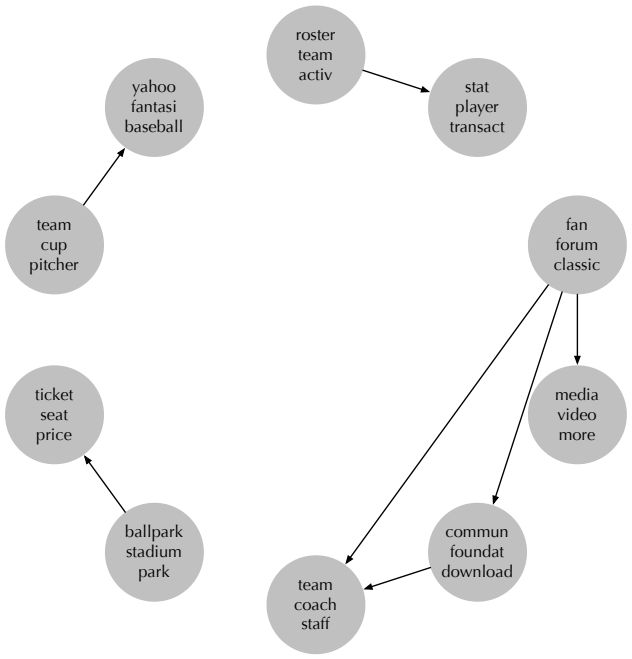


Figure 3: Example of transition between topics corresponding to clicked links. These topics are estimated from sites related to "sports teams". Terms are stemmed by the Porter stemmer.

such as web page classification or duplicate detection. Although we use similar features to those introduced in the above studies, our work is different in that their techniques focus on DOM blocks while we emphasize the links in those blocks.

We learn a feature-based model using the Gradient Boosted Decision Tree (GBDT) algorithm proposed by Friedman [16]. More recently, this method was adapted for ranking by Zheng et al. [28].

8. CONCLUSIONS

We have demonstrated that traffic-based link suggestion solutions, while effective, can be significantly improved using non-traffic-based data as well as clustering. These results imply not only that existing link suggestion systems can be improved, but also that their coverage can be extended to tail sites whose lower popularity often results in poorer performance on them.

There are several possible areas of future research. As mentioned earlier, if we are going to evaluate dynamic quick link algorithms for tail sites, we need to develop techniques for moving beyond traffic-based evaluation. We also think that there could be several improvements to our modeling, in terms of features, algorithms, and clustering.

We believe our approaches, though, suggest a compelling future research direction focusing on abstracting site and link semantics. Our clustering of sites and links was heavily motivated by a hypothesis that groups of sites form cohesive classes of concepts (e.g. 'restaurants', 'universities'), within which there exist prototypical link classes (e.g. for the 'restaurants' concept, 'menu', 'directions', and 'reservations' links). Our results support this hypothesis, and extensions to our models should certainly be explored. The

utility of these abstractions can also go beyond simple link suggestion; we can imagine a system more intelligently reasoning about a class of sites and prototypical links in response a specific user information need (e.g., 'find me menus for restaurants within 3 blocks').

9. REFERENCES

[1] J. Arguello, F. Diaz, and J.-F. Paiement. Vertical selection in the presence of unlabeled verticals. In *SIGIR '10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010.

[2] D. Blei and J. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems 20*. 2008.

[3] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36, 2002.

[4] D. Chakrabarti, R. Kumar, and K. Punera. Page-level template detection via isotonic smoothing. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 2007.

[5] D. Chakrabarti, R. Kumar, and K. Punera. Quicklink selection for navigational query results. In *WWW '09: Proceedings of the 18th international conference on World wide web*, 2009.

[6] J. Chen, B. Zhou, J. Shi, H. Zhang, and Q. Fengwu. Function-based object model towards website adaptation. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, 2001.

[7] K. Chen, R. Lu, C. K. Wong, G. Sun, L. Heck, and B. Tseng. TRADA: tree based ranking function adaptation. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, 2008.

[8] X. Chen and X. Zhang. A popularity-based prediction model for web prefetching. *Computer*, 36(3), 2003.

[9] R. R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1), July 2006.

[10] C. Danescu-Niculescu-Mizil, A. Z. Broder, E. Gabrilovich, V. Josifovski, and B. Pang. Competing for users' attention: on the interplay between organic and sponsored search results. In *WWW '10: Proceedings of the 19th international conference on World wide web*, 2010.

[11] B. D. Davison. Predicting web actions from html content. In *HYPERTEXT '02: Proceedings of the thirteenth ACM conference on Hypertext and hypermedia*, 2002.

[12] C. Doerr, D. von Dincklage, and A. Diwan. Simplifying web traversals by recognizing behavior patterns. In *HT '07: Proceedings of the eighteenth conference on Hypertext and hypermedia*, 2007.

[13] X. Dongshan and S. Junyi. A new markov model for web access prediction. *Computing in Science and Engineering*, 4, 2002.

[14] D. Duchamp. Prefetching hyperlinks. In *USITS'99: Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems*, 1999.

[15] E. Frias-Martinez and V. Karamcheti. A prediction model for user access sequences. In *WEBKDD Workshop: Web Mining for Usage Patterns and User Profiles, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2002.

[16] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29, 2000.

[17] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, 2005.

[18] T. L. Griffiths and M. Steyvers. Finding scientific topics. In *Proceedings of National Academy of Sciences. 101*, 2004.

[19] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. Dom-based content extraction of html documents. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, 2003.

[20] I. Juvina and E. Herder. The impact of link suggestions on user navigation and user perception. In *UM 2005: Proceedings of the 10th International Conference on User Modeling*, 2005.

[21] E. Kranakis, D. Krizanc, and S. M. Shende. Approximate hotlink assignment. In *ISAAC '01: Proceedings of the 12th International Symposium on Algorithms and Computation*, 2001.

[22] S.-H. Lin and J.-M. Ho. Discovering informative content blocks from web documents. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[23] M. Perkowitz and O. Etzioni. Adaptive web sites. *Commun. ACM*, 43(8), 2000.

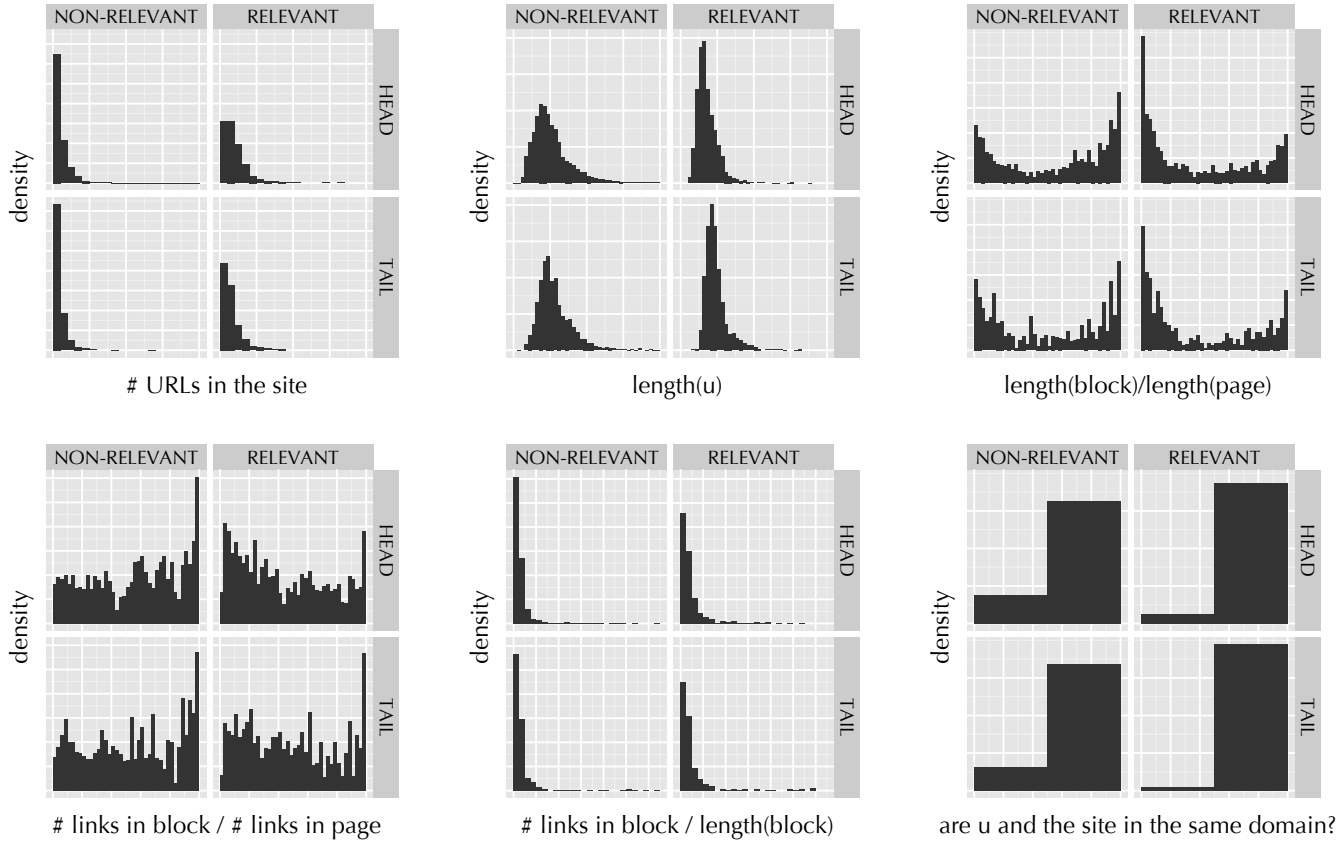
[24] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for web pages. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, 2004.

[25] R. Srikant and Y. Yang. Mining web logs to improve website organization. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, 2001.

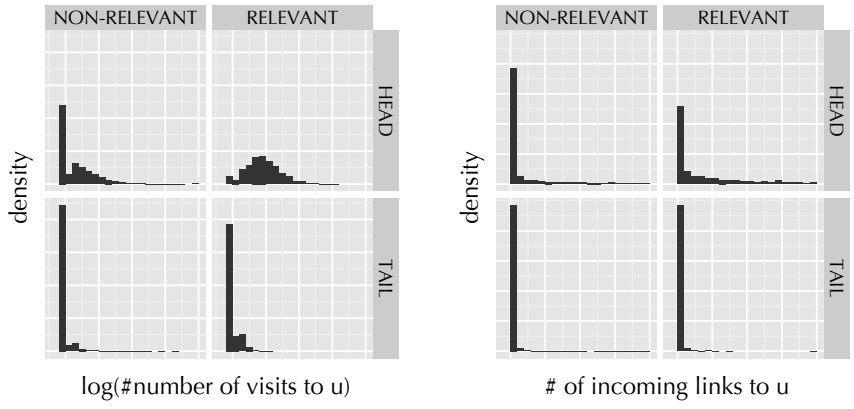
[26] C. Wang and S. Mahadevan. Multiscale analysis of document corpora based on diffusion models. In *IJCAI 2009: Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009.

[27] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[28] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems 20*. 2008.



(a) Top common features in GBDT-C



(b) Top head features in GBDT-HC

Figure 2: Distributions of top features of GBDT-C and GBDT-HC models for relevant and non-relevant sets in head and tail sites. Horizontal and vertical axis scales and ranges within each group of plots are identical. Axis tick labels removed for clarity.