Query-log mining for detecting polysemy and spam

Carlos Castillo[1], Claudio Corsi[2], Debora Donato[1], Paolo Ferragina[2], and Aristides Gionis[1]

[1] Yahoo! Research Labs, Barcelona, Spain
[2] Dipartimento di Informatica, University of Pisa, Italy

Abstract. Through their interaction with search engines, users provide implicit feedback that can be used to extract useful knowledge and improve the quality of the search process. This feedback is encoded in the form of a *query log* that consists of a sequence of *search actions*, which contain information about submitted queries, documents *viewed*, and documents *clicked* by the users. In this paper, we propose characterizing documents and queries via the information available within a query log, with the goal of detecting either query polysemy or spam-hosts and spam-queries, i.e., queries that shown the undesirable property of showing a higher rate of spam pages in their list of results than other queries. The main contribution of our paper consists of exploiting user feedback and query-log mining to combat spam and identify query polysemy. Our experiments attest the effectiveness of our approach for the applications we consider.

1 Introduction

Every day millions of users are searching for information on the web. Through their interaction with search engines, users locate not only the information they are looking for, but also provide implicit feedback to the results shown by the search engine for their queries. Nowadays search engines record query logs that consist of a sequence of *search actions*, which contain information about submitted queries, documents *viewed*, and documents *clicked*. Such information can be seen as "soft" relevance feedback for the documents that are clicked as results of specific queries, and queries can be interpreted as *tags* for the clicked documents.

Typically, a query log is abstracted by means of a graph structure, called *query-log graph* or *click graph* [1–3]. The click graph is an undirected labeled bipartite graph G in which nodes correspond to queries and documents, and edges indicate that at least one user who submitted query q has subsequently clicked on a result document d. Each edge is associated with a weight $w(q, d)$ that may indicate how many times query q led a user to click on document d, or how many distinct users clicked on d after submitting q. The click graph is *sparse* because many relevant documents have been possibly not clicked by users, and *noisy* because many non-relevant documents may have been clicked by several users (or robots). Nonetheless, the overall structure of the click graph captures

valuable information. For instance, an edge (q, d) can be interpreted as a vote of q about d; the larger is $w(q, d)$, the stronger is the relation between q and d. Also, document nodes with high degrees in G indicate documents that are reachable by many different queries, such as the home page of a large web portal or an auction site; similarly, query nodes with high degrees in G indicate queries that are not very precise, perhaps polysemous queries.

In this paper, we introduce two new variants of the click graph: the *view graph* and the *anti-click graph*. The view graph defines an edge (q, d) if the document d is in the list of results for query q. The anti-click graph defines an edge (q, d) if d has been ranked in the answers list for q before the first document that has been clicked by a user. Anti-clicks provide negative feedback, indicating documents that are less relevant for a query-user than the clicked documents.

Our main idea is to extract features from these three graphs in order to detect (1) spam hosts, (2) spam-prone queries, and (3) polysemous queries.

We propose two types of features: *syntactic* and *semantic*. The syntactic features, defined for document nodes, aim at capturing the *query attractiveness* of a document: to which extent a document attracts distinct queries. The semantic features are defined over all nodes, i.e., documents and queries, and aim at capturing the semantic diversity of queries and documents. We measure semantic diversity by exploiting a novel concept of entropy defined on the distribution of inferred topics over documents and queries. In order to get these categories, we follow [1, 4–6] and start from the labeling information available for a small set of topically-categorized documents, derived from human-edited web directories such as `DMOZ`, and propagating those topic labels to other unlabeled documents and queries in the query graphs.

The three applications we consider in this paper are the following:

Host-spam detection. This is a well-known problem in Adversarial IR [7]. However, unlike known approaches, which concentrate on specific techniques that spammers use to create their sites, we concentrate on the eventual outcome of these techniques by detecting specific structural and content patterns in our three query graphs. Our intuition is that spammers have an incentive to aim at the top results of frequent queries, and to aim at queries that are semantically far apart from each other in order to cover the largest web audience. So we use our syntactic and semantic features in order to detect those "query-attracting" hosts and improve known (host-)spam detection algorithms. Recently, usage data is being adopted in the literature as a source of information for spam detection. In [8] query logs were used to extract frequent query terms that are likely to appear in spam pages. In [9], query logs as well as browse logs from a toolbar were used to detect spam pages, using query independent features such as number of visits from search engines versus number of visits from browsing activity.

For the task of spam detection, our experiments show that we are able to achieve comparable accuracy with the results of [8] while ignoring the content of documents.

Query-spam detection. We are interested in detecting queries that generate a high number of spam pages placed in the top positions of their results. Detecting

such queries can be used for improving the quality of search results by, say, applying a more aggressive spam threshold, or by using those queries to design more sophisticated spam-detection algorithms. To our knowledge, this is the first paper in the literature that addresses the problem of detecting spam-attracting queries. Our experiments show that we are able to detect spam-attracting queries with a reasonable level of accuracy.

Query polysemy. Assessing the extent to which a query is polysemous (one classic example is the query "`jaguar`") can provide useful information for improving the quality of search results. For example, a search engine might take into account the context of a polysemous query, e.g., previous queries issued by the user, or user profile, in order to produce more accurate results. Our contribution in this paper consists of exploiting user feedback and query-log data in a novel way in order to measure the ambiguity of queries. For evaluating our approach, we built two benchmark datasets for detecting polysemous queries, and using query-graph features and a multi-level perceptron neural-network algorithm we obtain classification accuracy of 31% (difficult dataset) and 82% (easy dataset).

The rest of the paper is organized as follows. Section 2 discusses previous work, while Section 3 introduces our notation. In Section 4 we propose novel features for spam detection and query polysemy detection. In Section 5 we describe in more details our three applications and discuss our experimental results.

2 Related work

Query log mining has received substantial attention in the last years (see e.g., [2] and references therein) in the context of query suggestions, query expansion or clustering, enriching Web taxonomies [10], query classification [11, 6, 12, 1], polysemous word detection [13], boosting search-engine efficacy [14] or efficiency [15], building sets of popular queries for spam detection [16, 8], etc. In our work we use query-log graphs in a novel way to make one step further in three applications: detection of polysemous queries, spam sites, and spam queries. Below, we comment the literature which relates more with our results, and highlight the main differences between our and known techniques.

Query classification. This problem consists of classifying queries according to a set of predefined topics. Given that queries contain very few terms, successful approaches (e.g., [11]) employ snippets, and other meta-information. Some of the features we compute are based on propagation of `DMOZ` categories like [6], but our paper differentiate from this like of work because our ultimate goal is not query/document classification.

Finding ambiguous queries. Measuring ambiguity of queries and their relation in information retrieval has been studied extensively in the literature, e.g., see [17–20]. These approach use natural-language processing techniques, such as language models, WordNet, and thesauri, and thus they are significantly different than our method. More in-spirit to our approach are the papers of Baeza-Yates [13], Qiu et al. [4], and Song et al. [5], in the sense that user feedback in query-log mining is used for detecting polysemous queries. However, these

approaches differ completely from our proposal since we utilize novel concepts of query-log graphs, we suggest new algorithms for propagating topic labels on the graphs, and we propose new entropy-based measures.

Propagation-based methods. The idea of propagating meta-information (query content, document content, topics, etc.) by using the link structure of the Web graph or the click graph has been used in the past to derive metrics for query classification, see e.g. [14, 6, 3, 21]. Our paper takes inspiration from those ideas, and propagates category-based taxonomies on our three graphs (see Section 3). However, unlike those papers, we are not interested in the semantic proximity of two objects (queries or documents), but we are rather concerned with the distribution of their topics over those taxonomies in order to infer some of their properties (polysemy or spam).

Host-spam detection. A number of different techniques have been proposed for dealing with web spam [8, 22–24, 16, 7]. Most of these methods are based on either content analysis, link analysis, or query mining. Unlike all known approaches we concentrate on the final outcome of the spam-sites rather than on a specific detection technique. Our approach is complementary to existing techniques and can be use to strengthen current spam-detection mechanisms.

Some of the results we present about using the query logs for detecting spam, were presented in preliminary form in [25].

3 Preliminaries and Notation

The *click graph* $C = (V_Q, V_D, E)$ is an undirected, weighted and labeled bipartite graph, consisting of a node set $V_Q \cup V_D$ and an edge set E. Nodes in V_Q denote the set of distinct queries, nodes in V_D denote a set of distinct documents, and an edge $(q, d) \in E$ denotes that a user clicked on the document $d \in V_D$ after submitting the query $q \in V_Q$. Every edge $(q, d) \in E$ has an associated weight $w(q, d)$. We will consider two weights: (i) the number of times a user clicked on document d after submitting query q; (ii) the number of distinct search-sessions in which q clicked on d. We use $\mathcal{N}_k(x)$ to denote the set of nodes in G that are at distance *exactly* k from node x, and use $\mathcal{N}_{\leq k}(x) = \cup_{i=1,\ldots,k} \mathcal{N}_i(x)$ to indicate the set of nodes in G that are at distance *at most* k from node x. In addition to the click-graph, we also define two alternative graphs:

The view graph. We define the click graph by considering as edges (q, d) all the documents d in the results set of a query q. The view graph is a generalization of the click graph since each click is also a view. Moreover a query could produce no clicks and so be not present in the click graph, but be present in the view graph. We notice that the view graph is more noisy than the click graph because it does not contain any user feedback. Nonetheless it can be still useful to detect spam sites, since spam sites try to be in the answer lists of different queries, while users do not necessarily click on them.

The anticlick graph. We define the *anti-click graph* A_r by considering an edge (q, d) whenever all the following conditions are met: (i) the document d appears in the top-r positions of the ranked list of results for the query q, (ii) the user who

submitted q did not click on d, but (*iii*) that user clicked on another document ranked below d. The anti-click graph intends to capture the negative judgment that users give implicitly with their clicks. Negative feedback was also shown to be useful by [26]. We used small values for r, considering that most users look only at the first few results [27]. In our experiments we set $r = 3$ but other values could be used in the future.

To reduce the sparsity of the data, the above graphs (click, view and anti-click) can be defined on *hosts* rather than URLs, by replacing the set of document nodes with their hosts. Therefore, in total, we define six types of graphs ({click, view, anti-click} $\times$ {documents, hosts}). In the rest of the paper we will refer to these graphs as C_d and C_h, V_d and V_h, A_d and A_h. When there is no need to distinguish among those graphs we will use the generic name G. Additionally, for every node $x \in V_Q \cup V_D \cup V_H$, we associate a string $\ell(x)$ describing the node: if $x \in V_Q$, $\ell(x)$ is the query string, otherwise $x \in V_D$ or $x \in V_H$ then $\ell(x)$ is the document URL/host string.

4 Query-graph features

4.1 Syntactic features

The most obvious feature is simply the degree of a node. For a document d, $|\mathcal{N}_1(d)|$ is the number of queries adjacent to d (the set $\mathcal{N}_1(d)$ provides a good description of the content of document d [28, 15]). Similarly, for each query q we consider $|\mathcal{N}_1(q)|$, the number of distinct documents clicked for q. To refine this feature we concentrate on *popular* queries:

- For each document d, we define $\texttt{topQ}_x(d)$ as the set of queries adjacent to d in G and being among the fraction x of the most frequent queries in the query log. We consider $x = 0.01$ and $x = 1.0$, where $\texttt{topQ}_{1.0}(d) = \mathcal{N}_1(d)$, and select as syntactic feature the cardinality $|\texttt{topQ}_x(u)|$. We notice that $x = 0.01$ works well as a feature for the applications we describe on this paper, and similar values of x give similar results.
- For each document d, we define $\texttt{topT}_y(d)$ as the set of query terms (except stop words) which compose the queries adjacent to d in G and being among the fraction y of the most frequent terms in the query log. We consider $y = 0.01$ and $y = 1.0$, where $\texttt{topT}_{1.0}(u)$ is the dictionary of all query terms (except stop words). Again, we select as another syntactic feature the cardinality $|\texttt{topT}_y(d)|$.

Note that $\texttt{topT}_y(u)$ is less precise than $\texttt{topQ}_x(u)$ but it relaxes the dependence on small variations in the query composition. The intuition underlying the selection of the above two features is that the larger their values are, the stronger and wider the *query attractiveness* of d should be and thus the more evident should be that d is a spam page. Clearly, this may induce *false positive* detection for good pages dealing with several topics (e.g., a multi-author blog). However, those pages can be easily detected as non-spam by using classic link- and/or content-based approaches (see section 5). Figure 1 shows the distribution

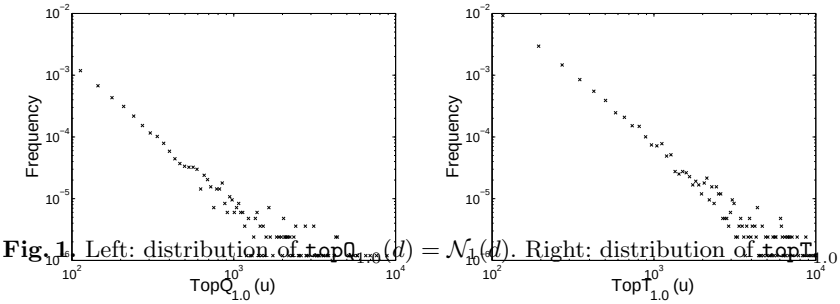

Fig. 1. Left: distribution of $\texttt{topQ}_{1.0}(d) = \mathcal{N}_1(d)$. Right: distribution of $\texttt{topT}_{1.0}(d)$.

of these two quantities on the click-graph C_d defined over the one month query log described in section 5.1. As in [1], a power law is observed.

4.2 Semantic features

The values of the syntactic features for a document d are not *robust* estimators of the *semantic coverage* of that document. For instance, a site about music can be reached by many distinct queries (e.g.: the names of many musicians) but still be topically coherent. Also, the semantic coverage could be a useful measure in order to make the detection of spam hosts more effective; for instance cloaked hosts are returned as results to queries that are semantically diverse.

We then propose new measures of *semantic coverage* of a node (document or query), based on a novel use of our graphs and of Web directories, such as the Open Directory Project `DMOZ`.[3] Our approach consists of two phases: (i) we first categorize the subset of documents in V_D that can be found in `DMOZ`, (ii) then, in order to amplify the `DMOZ` coverage we *propagate* the categories assigned to the (few) documents/hosts nodes of G to other document/host nodes. At the end, many categories may be associated with a node of G, indicating that this node is polysemous, in case of a query node, or multi-topical, in case of a document/host node. Furthermore, each category will have associated an *assignment strength* denoting the relation between the node content and the category label. We point out that this multi- and weighted-category assignment will be deployed at the end of this Section to derive three dispersion measures that capture the semantic spread of every node (document or query) in G.

The category tree. Let $\mathcal{T}_L$ be the category tree underlying the `DMOZ` directory, pruned to the top L levels. In our experiments we consider $L = 2$, thus managing 577 categories. We assume that every category (node) c of $\mathcal{T}_L$ is associated with a string $\ell(c)$, which denotes the name of the category. Our goal is to associate one category tree $\mathcal{T}_L(v)$ to each node v of the graph G (either query or document node), in such a way that the score $\texttt{score}_v(c)$ denotes the *strength* of the relation

[3] `http://www.dmoz.org/`

between v and the topic c. We observe that $\mathcal{T}_L(v)$ offers a good description of the semantic coverage of the node v. We then assert that the "wider" is the distribution of positive scores among the nodes of this tree, the wider should be the semantic spread of node v.

In order to compute these trees, we associate an initial tree $\mathcal{T}_L(v)$ to each node v in G, with all the scores set to zero. (Recall that G may be any one of the six graphs mentioned in Section 3.) Then, we scan through the document-nodes d of G and check whether d is assigned to some category c of DMOZ. If so, we increment by 1 the score of c and of all its ancestors in $\mathcal{T}_L(d)$. Note that, if c occurs deeper than level L in DMOZ, then we take its ancestor at level L, and perform the above updates on this node (and its ancestors). We normalize all the scores in such a way that $\sum_{c' \in \text{child}(c)} \texttt{score}_v(c') = 1$. After the normalization we can consider $\texttt{score}_v(c)$ as the probability that node v is about sub-topic c', conditioned to the fact that it is about c.

Given the score values $\texttt{score}_v(c)$, we also define $\texttt{score}'_v(c) = \texttt{score}'_v(\pi(c)) \times \texttt{score}_v(c)$, where $\pi(c)$ is the parent of c in $\mathcal{T}_L$. In particular, for the category node r at the root of $\mathcal{T}_L(v)$, we define $\texttt{score}'_v(r) = 1$ because we assume that category r includes all possible topics. In some sense $\texttt{score}'_v(c)$ captures the probability of reaching a category-node c of $\mathcal{T}_L$, when one starts on the root of $\mathcal{T}_L$ and moves downwards according to the scores $\texttt{score}_v(c)$.

After the initialization step, few category trees are non-null because DMOZ covers a small portion of the Web (see Table 1). Then we apply a *category propagation* process whose goal is to spread the category scores to other query and document nodes, driven by the structure of G and boosted by its edge weights. We propose two propagation strategies.

Tree-based propagation by weighted average. The first propagation algorithm views the graph G as a network of *voters*. Their contributions may sum up over multiple paths, but also they decay with the lengths of those paths. Thus we suggest to propagate the category scores through the graph G by boosting their contributions according to the edge weights, and by damping their impact according to the propagation distance. This way the value of $\texttt{score}_v(c)$ will be large if there is a large *volume* of *short* paths that start from vertex u with $\texttt{score}_u(c) > 0$ and end at vertex v. We implement these ideas as follows. At the generic step $i = 0, \ldots, t$, we scan through the nodes v in G and update the scores of all categories c in $\mathcal{T}_L(v)$ as:

$$\texttt{score}_{\texttt{v}}^{\texttt{i+1}}(\texttt{c}) \;\; += \;\; \alpha^{\texttt{i-1}} \sum_{(\texttt{v}',\texttt{v}) \in \texttt{E}} \texttt{score}_{\texttt{v}'}^{\texttt{i}}(\texttt{c}) \times \texttt{f}(\texttt{v}', \texttt{v})$$

where $\texttt{score}_v^0(c) = \texttt{score}_v(c)$, f is a increasing function set to $\log_2(1 + w(v', v))$, and α is a *damping* factor that takes into account the fact that the *relatedness* between two nodes at distance t in G decays with t. Notice that at step i, $\texttt{score}_v^i(c)$ receives contributions from all vertices in $\mathcal{N}_{\leq i}(v)$ properly weighted and scaled. This means that, in computing $\texttt{score}_v^\ell(c)$ we are eventually accounting for the *volume* of votes that suggest to attach category c to v's content. In our experiments α has been set to 0.85, as is usual in the PageRank algorithm, and $t = 2$,

which means that we propagate forth and back the category trees starting from the document nodes of G. We normalize the scores of each category tree at each propagation step in order to guarantee $\sum_{c' \in \text{child}(c)} \texttt{score}_v(c') = 1$.

Propagation by random walk. Our second propagation algorithm flattens the category structure by considering only the 17 top-level categories in DMOZ. For a fixed category c, the random-walk approach models the behavior of a random surfer that walks through the click-graph G and swaps her interests from queries to documents, and reverse. This is what is done in [29] for pages and [3] for query logs. The way the surfer chooses the next node among the ones adjacent to the current one (being either a document or a query) depends on their popularity among the search-engine users: The transition probability over edge (v, v') is proportional to $w(v, v') / \sum_z w(v, z)$ [3], where the weight $w(\cdot)$ can be defined either in terms of number of clicks or on the number of distinct search-sessions (we use the first one). The surfer has no memory of her previous location and sometimes she may restart (or "teleport" herself) to a document belonging to category c [29], chosen among all those documents with probability proportional to $\texttt{score}'_v(c)$. This way we take into account the relatedness of v's content with c's topic. Notice that the restart of the random walk reaches only document nodes and that, by using the scores $\texttt{score}'_v(c)$ instead of $\texttt{score}_v(c)$, we uncondition on the structure of the categories in the hierarchy tree.

By repeating this calculation for all categories of DMOZ in the 17 top-level categories, we get the category trees for all nodes in G as probability distribution $\mathbf{p}[\mathbf{v}]$ over all the considered categories. In particular we consider the normalized version of such vectors: $\mathbf{p}[\mathbf{v}] / ||\mathbf{p}[\mathbf{v}]||_1$.

Entropy-based semantic features. We propose three dispersion measures that capture the semantic spread of a node v by exploiting its category tree. Since $\texttt{score}_v(c)$ captures the probability that the query or document v is related to the topic c, the classic notion of entropy provides a good starting point for the design of such measures. However, given that we want to preserve the tree structure of the categories, we cannot use the classic definition directly. Therefore we fix a level i in $\mathcal{T}_L(v)$ and consider the conditional entropy

$$\mathtt{H_i(v)} = - \sum_{\text{level}(\mathtt{c})=\mathtt{i}-1} \mathtt{p(c)} \sum_{\mathtt{c'} \in \text{child}(\mathtt{c})} \mathtt{p(c'|c)} \log_2 \mathtt{p(c'|c)},$$

where c ranges among the level-$(i-1)$ nodes of DMOZ, and

$$\mathtt{p(c'|c)} = \frac{\mathtt{score_u(c')}}{\sum_{\mathtt{x} \in \text{child}(\mathtt{c})} \mathtt{score_u(x)}}$$

is the probability to reach node c' given that we are at its parent node c. Therefore, $H_i(v)$ measures the uncertainty of selecting a category at level i given that we are at some category at level $(i-1)$. The larger is this uncertainty, the wider should be the semantic coverage of v, and thus the stronger should be the evidence that v is a polysemous query, or a document covering multiple topics.

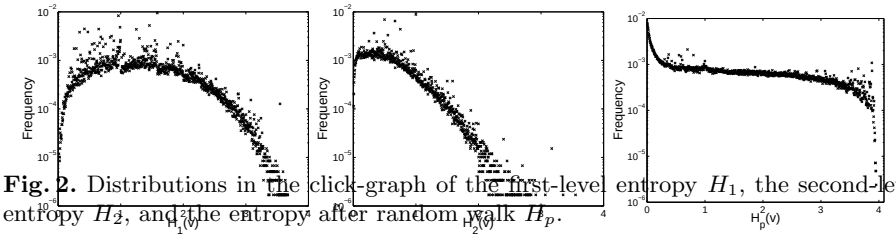

Fig. 2. Distributions in the click-graph of the first-level entropy H_1, the second-level entropy H_2, and the entropy after random walk H_p.

Having fixed the maximum depth of the trees to $L = 2$, we define a first measure of dispersion as follows: $H(v) = \beta H_1(v) + (1 - \beta)H_2(v)$. In this case, if $\beta = 0$ then the distribution among the level-2 categories dominates; if $\beta = 1$ then the distribution among the level-1 categories dominates; finally, if $\beta = 0.5$ then $H(v)$ is the half of the joint entropy of the first two levels of $\mathcal{T}_L(v)$. Therefore by setting $\beta = 0.75$ we give a preference to the distribution in the level-1 categories. In defining $H(v)$, we use the intuition that categories at the i-th level of DMOZ-taxonomy should be weighted more than categories in its $(i+1)$-level. For example, consider a category tree T_1 that has exactly k nodes with positive scores and all nodes lie at the fist level; and consider a category tree T_2 that has also k nodes with positive scores, but which lie at the second level. It is intuitive that we should consider T_1 more disperse than T_2.

In a similar way we define the second measure for the semantic coverage of a node v in the graph, called the *joint entropy* (HJ). Considering the nodes c on level 2 of $\mathcal{T}_L(v)$, we compute their joint probability as $p(c) = p(c|parent(c)) \times p(parent(c))$. We then apply the standard entropy function over the resulting probability distribution to obtain HJ.

As last semantic feature we compute the classic notion of entropy over the vectors $\mathbf{p}[\mathbf{v}]$ computed in the propagation based on random walk, where we considered just the 17 top-level categories of DMOZ. We denote such measure as H_p. This value is minimal if the vertex v has a non-zero score for exactly one category, and maximal if v has the same score in all categories. This captures the notion of polysemy that we are interested upon.

The distributions H_1, H_2 and H_p are shown in Figure 2. There are 17 top-level categories in the dataset which means a maximum value of $log_2(17) \approx 4.08$ in H_1 and H_p, and the number of second-level categories varies in different parts of the taxonomy but documents seem to be highly concentrated on a few second-level categories.

5 Applications

We describe three applications of the (syntactic and semantic) features introduced in the previous section: detecting polysemous queries, detecting spam

Table 1. Statistics on the query graphs expressed in percentage.

	Document-level			Host-level		
	C_d	A_d	V_d	C_h	A_h	V_h
Queries	1.59M	0.75M	2.78M	1.59M	0.75M	2.78M
Docs/hosts	2.75M	1.31M	23.47M	0.83M	0.40M	3.08M
Edges	3.69M	1.67M	40.71M	3.50M	1.53M	3.45M
$\mathcal{C}_D(0)$	0.05	0.08	0.03	0.28	0.35	0.15
$\mathcal{C}_Q(1)$	0.18	0.24	0.39	0.58	0.75	0.92
$\mathcal{C}_D(2)$	0.22	0.22	0.45	0.70	0.75	0.94
$\mathrm{CC}_{\max}$	0.32	0.19	0.92	0.80	0.83	0.98
$\lvert\mathrm{CC}\rvert$	0.21	0.23	0.007	0.08	0.06	0.006

Table 2. Top ambiguous and unambiguous queries found on the test set

Ambiguous
israel, addresses, indonesia, turkey, brazil, taiwan, ikea, films, germany, spain, tax, history, bristol, romania

Unambiguous
acyclovir, lithospermum, mencius, nelfinavir, amiodarone, aqueducts, balcombe, bergen-belsen

hosts, and detecting spam-attracting queries (a novel problem we introduce in this paper).

5.1 Datasets

As dataset we used an in-house query log. We picked a sample consisting of about 1.6 million queries leading to clicks on about 2.8 million distinct documents. The average number of terms per query is 3 in our sample. We also parsed the `DMOZ` hierarchy, as of October 2007, which contained about 4.2 million distinct documents in about 600 thousand distinct categories.

In Table 1, the number of query nodes, document (or host) nodes, and edges is given for all six graphs extracted from the query log (both URL-based and Host-based versions of our three graphs). In the table, $\mathcal{C}_D(k)$ is the fraction of documents (or hosts) covered by `DMOZ` after the k-th propagation step; $\mathcal{C}_Q(k)$ is the fraction of queries covered by `DMOZ` after the k-th propagation step; $\mathrm{CC}_{\max}$ is the size of the largest connected component, and $|\mathrm{CC}|$ is the number of connected components, both given as as a fraction of all the nodes in the graph. The size of the connected components follows a power-law distribution as in [1]. We also note that two propagation steps are enough cover 20% of the click graph C_d, and about 40% of view-graph on docs V_d and 90% of view-graph on hosts V_h (which are indeed highly connected). Thus, the former graph induces a stronger relation than the latter does.

Starting from this dataset we generate the graphs C_d, V_d and A_d together with their host-based version (C_h, V_h and A_h). Over them we generate all the syntactic and semantic features as described in sections 4.1 and 4.2. We also include also some statistical information about our graphs, as the number of clicks, views, and anti-clicks per query and document. We computed these statistics in two versions: discarding multiple clicks/views/anti-clicks in the same session for the same query/document pair, or counting them. In total, we have 61 features for each node.

5.2 Application to polysemy

First we split the queries into two sets: one containing unambiguous queries, and the other containing ambiguous queries. The heuristic we used to build the set of ambiguous queries was to consider a query ambiguous if it occurs as name of two or more distinct categories of DMOZ. As an example "jaguar" is ambiguous because it occurs in "Shopping: Vehicles: ... British: Jaguar" and "Kids and Teens: ... : Animals: Mammals: Jaguar". Instead, a string occurring as a name of only one category, is considered unambiguous. In total we had 8,461 queries, out of which 23% were ambiguous and the remaining 77% were unambiguous.

We built a fast decision tree (i.e. `reptree` as implemented in `weka`) as automatic classifier, in order to capture the extent to which a query is polysemous. After ten-fold cross-validation, the best performance we obtained was a 31.8% of detection rate (true positives) with 8.8% rate of false negatives. The AUC of this classifier was 0.728. Table 2 lists a sample of the top 15 queries (ordered by confidence) that were considered to be ambiguous or non-ambiguous on a test set (after training on 66% of the data, and testing on the rest).

Most of the queries considered to be ambiguous are names of countries, which seems intuitive given that there can be many possible intents by users typing a query like "taiwan". Most of the queries considered to be unambiguous, on the other hand, are technical terms, for instance, names of drugs.

The low detection rate achieved can be explained, in our opinion, by the fact that it is in inherently hard to say if a query is ambiguous or not in *absolute terms*. However, we can try to check if one query is *more ambiguous* than another query. For this second task, we use pairs of queries and ask an automatic classifier to learn to order them correctly using the features we provide.

We used WordNet,[4] a free linguistic resource consisting of a large set of annotated entities. We considered the "hypernym" relation that indicates that one sense is more general than the other. For instance, going from specific to general, a "horse" is an "equine", which is a "mammal", which is an "animal", which is a "living thing" and so on. Our learning task is, given a pair of queries for which there exists a "hypernym" relation, identify which of the two is more specific than the other. Using our features and a multi-level perceptron neural-network algorithm we obtain an accuracy of 0.82, meaning that we classify 82% of the pairs correctly (50% would be a random classification).

5.3 Application to Web spam

Finding Web spam pages. For building and testing our automatic classifier, we use the hosts occurring in the click-graph C_h and in the WEBSPAM-UK2006 dataset [30]. A set of pre-computed content-based and link-based features is available for this collection.[5]

Note that the search engine already filters out a large fraction of Web spam, so the spam hosts that are clicked in the query log are either: high quality pages

[4] http://wordnet.princeton.edu/

[5] http://webspam.lip6.fr/

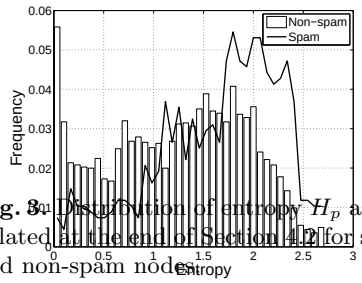

Fig. 3. Distribution of entropy H_p as calculated at the end of Section 4.2 for spam and non-spam nodes.

Feature set	Features	TP	FP	F_1	AUC
Content (C)	98	75.8%	9.8%	0.692	0.912
Links (L)	139	84.2%	9.5%	0.739	0.939
Usage (U)	61	54.2%	7.4%	0.557	0.872
C ∪ L	237	83.9%	8.6%	0.756	0.952
C ∪ U	159	68.4%	6.6%	0.693	0.917
L ∪ U	200	78.5%	6.5%	0.757	0.951
C ∪ L ∪ U	298	78.9%	6.2%	0.765	0.951

Fig. 4. Results for Web spam classifiers.

that use spam tricks according to the assessors of the collection, but that are nevertheless valuable for users of the search engine; or pages that simply can defeat the automatic spam filter of the search engine. In any case, the performance results that we report below are shown for the fraction of spam that is seen by the users of the search engine.

Previous systems for Web spam detection and demotion [8, 23] are typically based on two classes of features: *content-based* features (statistics on the text and formatting of the pages) and *link-based* features (statistics on the web graph). In this paper we introduce *usage-based* features, some of which are very good at separating spam and normal hosts; in Figure 3 where the distributions of H_p in C_h for spam hosts (lines) and normal hosts (bars) are clearly different.

We then used a cost-sensitive decision tree with bagging (for an introduction on these techniques see [31]) and adopted the following performance measures: the true positive rate TP (or recall), the false positive rate FP, the F_1 metric, which combines precision P and recall R by $F_1 = 2\frac{PR}{P+R}$, and the area under the ROC curve (AUC). We compare the performance of our 61 usage-based features against the 98 link-based and the 139 content-based features provided with the dataset. Figure 4 reports the experimental results, in terms of F_1 and AUC. We notice that, in term of global accuracy, the classifiers (C ∪ L ∪ U), (C ∪ L) and (L ∪ U) are comparable. Nevertheless the (L ∪ U)-classifier could be considered an improvement on the previous methods for the following reasons:

- it uses less features than the (C ∪ L)-classifier introduced in [8].
- it does not need to look at the content of the pages.

Detecting Spam Queries. We refer to a *spam-attracting query* as a query that has a high number of spam hosts in the set of its results. We start with V_h, the view-graph at the host level, and keep only the hosts in V_h for which we have a non-spam or spam label. We use the WEBSPAM-UK2006 dataset to obtain spam and non-spam labels for a subset of the hosts in V_h. Next we define the spamicity of a query as the number of results labeled as spam and shown to the user for that query, over the total number of results labeled (spam or normal) and shown

to the user. For instance, a spamicity of 0.5 for a query q indicates that from the labeled hosts present in the result set of q, half of them were spam.

To reduce the noise given the low coverage of our labels, we consider only the set of queries for which we have at least 10 labeled hosts in the results shown to the user. We pick a sample of 1.2 million queries. We use this number to ensure that we have enough labeled data for a query before reading the fraction of spam found in the results for that query.

Next we divide the queries into two groups: queries having spamicity ≥ 0.5 and queries having spamicity < 0.5. We then build a decision tree, and obtain an AUC of 0.798, true positive rate of 73.7% with 29.0% of false positives. If instead we consider the task of finding queries with spamicity=0 versus queries with spamicity≥ 0.5, we obtain an AUC of 0.838 with a true positive rate of 74.0% and false positive rate of 22.1%. These results suggest that usage data can be used to automatically extract queries that are likely to be showing a substantial amount of spam results; however, probably other features are required to improve the accuracy of such a detection system.

6 Conclusions

In this paper we have considered the click graph, and two novel variants—the view graph and the anti-click graph. We proposed syntactic and semantic features extracted from these graphs and used them to address three applications: query polysemy, document-spam detection and query-spam detection. Our experiments have shown good performance, which sometimes improved known results, as occurred in Web spam detection were we can achieve the same performance as known classifiers with less features and considering usage information. Usage-based spam-detection methods are just starting to be explored [25, 9]. For future work, we would like to refine the metrics we have proposed here and study other potential applications in the future. Studying the individual impact of each graph and feature we have introduced in this paper is another interesting topic.

References

1. Baeza-Yates, R., Tiberi, A.: Extracting semantic relations from query logs. In: Proceedings of KDD. (2007)
2. Baeza-Yates, R.A.: Mining queries. In: ECML/PKDD. (2007)
3. Craswell, N., Szummer, M.: Random walks on the click graph. In: SIGIR. (2007)
4. Qiu, G., Liu, K., Bu, J., Chen, C., Kang, Z. In: SIGIR. (2007)
5. Song, R., Luo, Z., Wen, J.R., Yu, Y., Hon, H.W. In: WWW. (2007)
6. Xue, G.R., Yu, Y., Shen, D., Yang, Q., Zeng, H.J., Chen, Z.: Reinforcing web-object categorization through interrelationships. Data Min. Knowl. Discov. **12**(2-3) (2006)
7. Fetterly, D.: Adversarial information retrieval: The manipulation of web content. ACM Computing Reviews (2007)
8. Castillo, C., Donato, D., Gionis, A., Murdock, V., Silvestri, F.: Know your neighbors: Web spam detection using the web topology. In: SIGIR. (2007)

9. Liu, Y., Cen, R., Zhang, M., Ru, L., Ma, S.: Identifying web spam with user behavior analysis. In: AIRWeb, Beijing, China (2008)
10. Chuang, S.L., Chien, L.F.: Enriching web taxonomies through subject categorization of query terms from search engine logs. Decision Support Systems **35**(1) (2003)
11. Shen, D., Sun, J.T., Yang, Q., Chen, Z.: Building bridges for web query classification. In: SIGIR. (2006)
12. Dai, W., Yu, Y., Zhang, C., Han, J., Xue, G.: A novel webpage categorization algorithm based on block propagation using querylog information. In: WAIM. (2006)
13. Baeza-Yates, R.: Graphs from search engine queries. In: SOFSEM 2007: Theory and Practice of Computer Science. (2007)
14. Shen, D., Sun, J.T., Yang, Q., Chen, Z.: A comparison of implicit and explicit links for web page classification. In: WWW. (2006)
15. Puppin, D., Silvestri, F.: The query-vector document model. In: CIKM. (2006)
16. Wu, B., Davison, B.D.: Detecting semantic cloaking on the web. In: WWW. (2006)
17. Cronen-Townsend, S., Croft, W.B.: Quantifying query ambiguity. In: International Conference on Human Language Technology Research. (2002)
18. Gonzalo, J., nas, A.P., Verdejo, F.: Lexical ambiguity and information retrieval revisited. In: EMNLP/VLC. (1999)
19. Richardson, R., Smeaton, A.F.: Using WordNet in a knowledge-based approach to information retrieval. In: BCS-IRSG. (1995)
20. Schutze, H., Pedersen, J.: Information retrieval based on word senses. In: Symposium on Document Analysis and Information Retrieval. (1995)
21. Szummer, M., Jaakkola, T.: Partially labeled classification with markov random walks. In: NIPS. (2001)
22. Gyöngyi, Z., Garcia-Molina, H.: Web spam taxonomy. In: AIRWeb. (2005)
23. Ntoulas, A., Najork, M., Manasse, M., Fetterly, D.: Detecting spam web pages through content analysis. In: WWW. (2006)
24. Wu, B., Davison, B.D.: Cloaking and redirection: A preliminary study. In: AIRWeb. (2005)
25. Castillo, C., Corsi, C., Donato, D., Ferragina, P., Gionis, A.: Query-log mining for detecting spam. In: AIRWeb, Beijing, China (2008)
26. Radlinski, F., Joachims, T.: Query chains: learning to rank from implicit feedback. In: KDD. (2005)
27. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback. In: SIGIR. (2005)
28. Xue, G.R., Zeng, H.J., Chen, Z., Yu, Y., Ma, W.Y., Xi, W., Fan, W.: Optimizing web search using web click-through data. In: CIKM. (2004)
29. Haveliwala, T.H.: Topic-sensitive pagerank. In: WWW. (2002)
30. Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., Vigna, S.: A reference collection for web spam. SIGIR Forum (2006)
31. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)