Using the Wisdom of the Crowds for Keyword Generation

Ariel Fuxman
Microsoft Research
Mountain View, California
arielf@microsoft.com

Panayiotis Tsaparas
Microsoft Research
Mountain View, California
panayiotis.tsaparas@microsoft.com

Kannan Achan
Microsoft Research
Mountain View, California
kachan@microsoft.com

Rakesh Agrawal
Microsoft Research
Mountain View, California
rakesha@microsoft.com

ABSTRACT

In the sponsored search model, search engines are paid by businesses that are interested in displaying ads for their site alongside the search results. Businesses bid for keywords, and their ad is displayed when the keyword is queried to the search engine. An important problem in this process is *keyword generation*: given a business that is interested in launching a campaign, suggest keywords that are related to that campaign. We address this problem by making use of the query logs of the search engine. We identify queries related to a campaign by exploiting the associations between queries and URLs as they are captured by the user's clicks. These queries form good keyword suggestions since they capture the "wisdom of the crowd" as to what is related to a site. We formulate the problem as a semi-supervised learning problem, and propose algorithms within the Markov Random Field model. We perform experiments with real query logs, and we demonstrate that our algorithms scale to large query logs and produce meaningful results.

Categories and Subject Descriptors: H.3.5 [Information Systems]: Information Storage and Retrieval — On-line Information Services; J.0 [Computer Applications]: General

General Terms: Algorithms

Keywords: Absorbing Random Walks, Keyword Generation, Markov Random Fields,Query Click Logs, Sponsored Search

1. INTRODUCTION

The main source of income for search engines is *web search advertising*, which places relevant advertisements together with the search engine results. Given a specific keyword (a single word or a short phrase), advertisers bid for the keyword, and the winner of the auction has her ads displayed as *sponsored links* next to the "algorithmic" results of the search engine.

The problem of identifying an appropriate set of keywords for a specific advertiser is called *keyword generation* (or *keyword research*). This problem is important for both the search engines and the advertisers. Providing a broad set of relevant keywords to an advertiser enables the search engine to tap the long tail of small businesses and organizations. At the same time, such small businesses now have inexpensive access to a powerful advertising medium, and they can attract traffic to their site by bidding on the appropriate keywords. All major search engines provide services for keyword research (e.g., Google's AdWords Keyword Tool[1], Overture/Yahoo! Keyword Selector Tool[2], and Microsoft adCenter Labs' Keyword Group Detection[3]).

Previous approaches to the keyword generation problem have exploited the content of either Web pages [10, 20, 26] or search engine results [14]. In this paper, we tackle the problem by making use of search engine *query-click logs*, an approach that has received limited attention in the literature [4]. Search engine query-click logs maintain the queries that users pose to the search engine and the documents that are clicked in return. Clicks define a strong association between queries and URLs. The semantics of a query is captured in the URLs that are clicked as a result of the query, while the queries that result in a click to a URL provide a short, concise description of that URL. We exploit this reinforcing relationship between queries and URLs to find queries that are related to the interests of the advertiser. Our approach has the advantage that it exploits the proverbial "wisdom of the crowds" for keyword generation. Furthermore, the suggested keywords take into account the click-through traffic that they generate in the search engine, and as a result they are more directly monetizable.

As an illustrative example, suppose that the owner of the `shoes.com` online store decides to launch an ad campaign. We can safely assume that most of the queries that end up in the Web site of `shoes.com` come from users interested in buying shoes. Furthermore, the click-logs enable us to expand this immediate set of queries further by exploiting semantic associations between queries and documents. Consider the sample of a search engine click log shown in Table 1. Since some user issued the query "running shoes" and clicked on `www.shoes.com`, we can assume that this query is about shoes. Notice that the query "running shoes" also resulted

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

[1] http://adwords.google.com/select/KeywordToolExternal
[2] http://inventory.overture.com/d/searchinventory/suggestion
[3] http://adlab.msn.com/contextSim/Default.aspx

Table 1: A sample of a click log

Query	Clicked Url
running shoes	www.shoes.com
running shoes	www.runningshoes.com
reebok shoes	www.runningshoes.com
rebok shoes	www.runningshoes.com

in a click to www.runningshoes.com, which in turn attracted clicks from the queries "reebok shoes" and "rebok shoes". We may then conclude (perhaps with less certainty) that these two queries and the URL www.runningshoes.com are also about shoes. Notice that although the latter query has a misspelling, it still represents an interest in buying shoes of a specific brand, and is therefore valuable to the advertiser.

We can now define the keyword generation problem as follows. Given a *concept* (such as shoes), a set of elements that represent the concept (such as a set of URLs), and the relationships between the documents and the queries, obtain a set of keywords that best capture the concept. We formulate the problem as a semi-supervised learning problem. The input consists of (i) a set of labeled objects about a concept (e.g., the URLs in the shoes.com domain); (ii) a set of unlabeled objects (the remaining URLs and the queries in the log); and (iii) a set of constraints between labeled and unlabeled objects (the click log). The goal is to label some of the unlabeled elements in a meaningful way.

There is an extensive literature in the area of semi-supervised learning [7]. In this work, we employ Markov Random Fields [17] to model the query click graph. Specifically, the clicks quantify pairwise relationships between documents and queries, and define a bipartite graph which induces a Markov Random Field. Informally, in a Markov Random Field the probability that an object is associated to a label depends only on the labels of its neighbors. In our context, this means that the probability of a query being associated to a concept is determined by the documents that were clicked for this query (and, analogously, for the probabilities of the documents).

The Markov Random Field model lends itself to different inference algorithms for computing the probability distribution over the labels for unlabeled objects. The main algorithm we consider in this paper makes use of absorbing random walks. This is an inference algorithm for Gaussian Markov Random Fields, that computes the probability of a query to belong to a concept as the average of the probability of the clicked URLs for this query (similarly for URLs). We also consider a variational inference algorithm where we attempt to maximize the entropy of an approximating label distribution while respecting the underlying constraints.

The contributions of this paper are the following:

- We provide an approach for keyword generation based on exploiting the query-click graph.
- We propose an algorithm for keyword generation based on a random walk with absorbing states. The algorithm is intuitively appealing and performs well in practice.
- We define a formal framework for keyword generation in terms of Markov Random Fields. We show how the random walk algorithm can be modeled within this framework, and we consider an alternative approach.
- We perform an experimental study in the context of a large-scale click log. The experiments show that this approach can be used to produce large, high-quality lists of keywords with minimal effort. For example, in one of the experiments we show that by leveraging a list of 12 popular domains, it is possible to construct a list of 500,000 keywords, 95.9% of which are relevant to the desired concept.

The rest of the paper is structured as follows. In Section 2, we define the keyword generation problem. In Section 3, we present an algorithm for the keyword generation problem that is based on random walks with absorbing states. In Section 4, we demonstrate how the keyword generation problem and the random walk algorithm can be formulated within the Markov Random Field model, and we consider an alternative algorithm within the Markov Random Field model. In Section 5, we perform an experimental validation of our techniques using real click logs. In Section 6 we present related work, and in Section 7 we give some concluding remarks.

2. PROBLEM DEFINITION

We now define the *click-based keyword generation problem*. We assume that the following is given as input.

1. A search engine click log $\mathcal{L}$. The search engine click log consists of triples $\langle q, u, f_{qu} \rangle$, where q is a query, u is the URL of a document, and f_{qu} is the number of times that the users issued query q to the search engine and clicked on URL u. We use $\mathcal{Q}$ and $\mathcal{U}$ to denote the set of all queries and all URLs, respectively, in the click log $\mathcal{L}$. We have that $\mathcal{L} \subseteq \mathcal{Q} \times \mathcal{U} \times N^+$.

   We will consider the click log $\mathcal{L}$ as a *weighted* bipartite graph $\mathcal{G} = (\mathcal{Q}, \mathcal{U}, E)$, henceforth called the *click graph*. The URLs $\mathcal{U}$ and queries $\mathcal{Q}$ constitute the partitions of the graph, and for every record $\langle q, u, f_{qu} \rangle$ in the log, there is an edge $(q, u) \in E$ with weight f_{qu}.

2. A set of *concepts* $\mathcal{C} = \{c_1, ..., c_k\}$. The concepts represent abstract themes that the advertiser is interested in. Concepts may be general (e.g., shoes), or specific (e.g., running shoes for teenagers). In the simplest case, $\mathcal{C}$ consists of a single concept provided by the advertiser. In the more complex one, it is a full taxonomy of different classes.

3. A *seed set* $\mathcal{S} \subseteq \mathcal{U} \times \mathcal{C}$ of URLs in the click log that are manually assigned to the concepts in $\mathcal{C}$. The seed set $\mathcal{S}$ consists of pairs $\langle u, c \rangle$ where $u \in \mathcal{U}$ and $c \in \mathcal{C}$ is the label of concept c. The set of URLs assigned to the concept c can be thought of as a representation of the concept c in terms of URLs.

Given $\mathcal{G}$, $\mathcal{C}$, and $\mathcal{S}$ the goal of the keyword generation problem is to populate the concepts in $\mathcal{C}$ with queries from $\mathcal{Q}$. These queries will be used as keyword suggestions to the advertisers that are interested in the specific concept.

Note that in the problem definition we defined the seed set to be a set of labeled URLs. We could also define the seed set as a set of labeled queries, or a mix of queries and URLs. For simplicity, we will restrict ourselves to the case that the seed set consists only of URLs.

3. A RANDOM WALK ALGORITHM

We begin the algorithmic exploration of the keyword generation problem by describing an algorithm that is based on a random walk with absorbing states. The algorithm is intuitively appealing and can be implemented efficiently. In the next section, we show the connection between this algorithm and the theory of Markov Random Fields.

Recall the motivating example presented in the previous section. The advertiser is interested in the general concept of "shoes" and the seed set consists of the URL `www.shoes.com`. Starting from this URL, we were able to associate the query "running shoes" to the concept "shoes" as some user had posed this query and clicked on the above URL. That is, we assumed that the query is associated to the concept "shoes" because one of its immediate neighbors (i.e., the URL) was also related to this concept. We also made a similar assumption for URLs associated to queries about shoes. Intuitively the queries (and URLs) are related to the concept of shoes because they are connected closely in the graph to the seed set that represents this concept.

We capture this intuition using a random walk. For some query $q \in \mathcal{Q}$, we compute the affinity of q to some seed node $s \in \mathcal{S}$ as the probability that a random walk that starts from q ends up at node s. The affinity of q to the class $c \in \mathcal{C}$ is the probability that the random walk that starts from q ends up in any seed node in the class c. Note that in this walk the nodes in the seed set act as *absorbing nodes*, that is, sink nodes in the state transition graph from which the random walk cannot escape. Absorbing nodes naturally model the fact that for the URLs in the seed set we have complete certainty about their class since they were manually assigned to that class, and thus their assignment should not be changed by the algorithm.

Note that for the case where there exists a single seed node s in the graph, every query that is reachable from the seed node will be absorbed with probability 1 at node s. However, the further away a query is from the seed URL, the less related it should be to the URL's class. We model this by introducing an absorbing "null class" node ω to the graph, and connecting this node to every other node in the graph. The effect of this null class node is that long paths are penalized by the algorithm, and the probability of a node being absorbed at the seed node decreases exponentially with the length of the path.

Performing a random walk for every query in the graph is computationally prohibitive for a large graph. However, there is a simple iterative algorithm that can compute the class probabilities efficiently. We will now describe the algorithm for the case that we have a single concept c (that is, $\mathcal{C} = \{c\}$), and then show how to generalize to the case of multiple classes.

Let ℓ_q (or ℓ_u) denote the random variable pertaining to the concept label for query q (or URL u). We want to compute $P(\ell_q = c)$, that is, the probability that a random walk that starts from q will be absorbed at some node of the class c. Let α be the probability of making a transition to the null class absorbing node, from any node in the graph. Then we have that

$$P(\ell_q = c) = (1-\alpha) \sum_{u:(q,u)\in E} w_{qu} P(\ell_u = c) \qquad (1)$$

where

$$w_{qu} = \frac{f_{qu}}{\sum_{u:(q,u)\in E} f_{qu}}$$

is the transition probability from query q to URL u, and $P(\ell_u = c)$ is the probability that a random walk that starts from URL u ends up being absorbed in class c. For all URLs u in the seed set, we have that $P(\ell_u = c) = 1$ if the pair $\langle u, c\rangle$ belongs in the seed set, and zero otherwise. For all other URLs, the probability $P(\ell_u = c)$ is again recursively computed as

$$P(\ell_u = c) = (1-\alpha) \sum_{q:(u,q)\in E} w_{uq} P(\ell_q = c) \qquad (2)$$

where

$$w_{uq} = \frac{f_{qu}}{\sum_{q:(q,u)\in E} f_{qu}}$$

is the transition probability from u to q.

The iterative process defined by Equations 1 and 2 is known to converge [9]. The computation has interesting connections with electrical networks. Consider the click graph as an electrical network, where each edge (q, u) is a wire, and the weight f_{qu} of the edge is the conductance of the wire. If we apply a unit of voltage to the nodes in the seed set, and we ground the absorbing node ω, then the probabilities that we compute are the voltages on the non-seed nodes.

The outline of the Absorbing Random Walk (ARW) algorithm is shown in Algorithm 1. Note that in lines 5 and 9 of the algorithm we discard probabilities that are below a threshold γ. This step is mainly for efficiency purposes. If we do not do any pruning, then all nodes that are reachable from the seed set will get some probability of belonging to the class regardless of how small that probability is. However, we are not interested in such queries, since they are too far from the seed set. Making the probabilities of these nodes to be zero conceptually means that we temporarily make them absorbing nodes and place them in the null class. At convergence, we have a set of null class nodes that define the boundary of the graph that we have explored.

Algorithm 1 The ARW algorithm for a single class

Input: the seed set $\mathcal{S}$ for class c, the click-graph $\mathcal{G}$, the threshold parameter γ, the transition probability α to ω
Output: $P(\ell_q = c)$, for every query q.

1: **for** $u \in \mathcal{S}$ **do** $P(\ell_u = c) = 1$
2: **repeat**
3: **for all** $q \in \mathcal{Q}$ **do**
4: $P(\ell_q = c) = (1-\alpha) \sum_{u:(q,u)\in E} w_{qu} P(\ell_u = c)$
5: **if** $P(\ell_q = c) < \gamma$ **then** $P(\ell_q = c) = 0$
6: **end for**
7: **for all** $u \in \mathcal{U} \setminus \mathcal{S}$ **do**
8: $P(\ell_u = c) = (1-\alpha) \sum_{q:(u,q)\in E} w_{uq} P(\ell_q = c)$
9: **if** $P(\ell_u = c) < \gamma$ **then** $P(\ell_u = c) = 0$
10: **end for**
11: **until** convergence
12: Output $P(\ell_q = c)$, for every query q in $\mathcal{Q}$

The algorithm generalizes naturally to the case where there are multiple concepts. More specifically, Equations 1 and 2 can still be used for computing the probabilities $P(\ell_q = c)$ and $P(\ell_u = c)$ for each concept $c \in \mathcal{C}$. We can thus

perform the same iterative algorithm for all classes simultaneously. However, this process is memory intensive since it requires to maintain probability vectors for all k classes. Furthermore, the fraction of the graph that will be explored will increase substantially even if we prune nodes with low probabilities, since the size of the seed set may be significantly larger.

This problem can be addressed by considering the concepts one at the time. However, this should be done carefully so as to take into account all the information contained in the seed set. Let $\mathcal{S} = \{\mathcal{S}_1, \cdots, \mathcal{S}_k\}$ be the partition of the seed set into the k concepts. Applying Algorithm 1 directly for each set $\mathcal{S}_i$ is incorrect, since we discard the information we have about the seed nodes in the other concepts. For these nodes, we know with certainty that they cannot belong to class c_i; thus, they are negative examples for the class c_i. We can easily model this effect in the absorbing random walk: when considering class c_i we make all the remaining seed nodes in $\mathcal{S} \setminus \mathcal{S}_i$ to be in the null class.

The outline of the ARW algorithm for multiple classes is shown in Algorithm 2. The outer loop of the algorithm iterates over all classes, and for each class we perform an absorbing random walk. The absorbing random walk for each individual class c_i is very similar to Algorithm 1, albeit with two distinct differences. In line 2 of the algorithm, we fix the probability of the nodes in $\mathcal{S} \setminus \mathcal{S}_i$ to belong to the class c_i to zero, thus making them into null absorbing nodes. Also, when we update the URL probabilities (lines 9 – 12), we do not update the probabilities for any of the URLs in $\mathcal{S}$. In effect, the seed set for the absorbing random walk for class c_i consists of the entire set $\mathcal{S}$; the nodes in $\mathcal{S}_i$ have probability 1, while the nodes in $\mathcal{S} \setminus \mathcal{S}_i$ have probability zero.

The effect of this optimization is that the memory footprint is decreased dramatically, since we only need to consider a single class at the time. Single class runs are also fast, since the size of the explored graph is smaller. Furthermore, the null absorbing nodes in $\mathcal{S} \setminus \mathcal{S}_i$ "block" the random walk, and thus speed up the convergence. Note also that the algorithm is amenable to parallel execution.

Algorithm 2 The ARW algorithm for multiple classes

Input: the seed set $\mathcal{S} = \{\mathcal{S}_1, \cdots, \mathcal{S}_k\}$ for concepts $\mathcal{C} = \{c_1, \cdots, c_k\}$, the click-graph $\mathcal{G}$, the threshold parameter γ, the transition probability α to ω.
Output: $P(\ell_q = c)$, for every query q and every class c.

1: **for all** $c_i \in \mathcal{C}$ **do**
2: **for** $u \in \mathcal{S} \setminus \mathcal{S}_i$ **do** $P(\ell_u = c_i) = 0$
3: **for** $u \in \mathcal{S}_i$ **do** $P(\ell_u = c) = 1$
4: **repeat**
5: **for all** $q \in \mathcal{Q}$ **do**
6: $P(\ell_q = c) = (1 - \alpha) \sum_{u:(q,u)\in E} w_{qu} P(\ell_u = c)$
7: **if** $P(\ell_q = c) < \gamma$ **then** $P(\ell_q = c) = 0$
8: **end for**
9: **for all** $u \in \mathcal{U} \setminus \mathcal{S}$ **do**
10: $P(\ell_u = c) = (1 - \alpha) \sum_{q:(u,q)\in E} w_{uq} P(\ell_q = c)$
11: **if** $P(\ell_u = c) < \gamma$ **then** $P(\ell_u = c) = 0$
12: **end for**
13: **until** convergence
14: Output $P(\ell_q = c)$, for every query q and class c
15: **end for**

Further optimizations are possible. Once the run for concept c_i is completed, we know that with probability $P_{qi} = P(\ell_q = c_i)$, the query q belongs to class c_i. Therefore, the probability mass "available" for the remaining classes is $1 - P_{qi}$. When considering another class c_j, we can create a random jump with probability P_{qi} to the null absorbing node. This will result in faster convergence for each individual run.

4. MARKOV RANDOM FIELDS

We now demonstrate how the keyword generation problem and the ARW algorithm that we described in the previous section can be formulated within the Markov Random Field model. In order to compare different Markov Random Field formulations, we also present another algorithm for computing the class probabilities. In Section 5, we report experiments comparing this algorithm against the ARW algorithm.

4.1 Model and definitions

A *Markov Random Field (MRF)* is an undirected graph, where each node in the graph is associated with a random variable and the edges model the pairwise relationships between the random variables. MRFs define a probability model where the value of a random variable in the field depends on the prior knowledge we may have for that variable, and the values of all the adjacent variables in the graph. The characteristic of the MRFs is the *Markovian assumption* that the value of a random variable is independent of the rest of the graph, given the values of all its neighbors. Given a set of *observations* about certain variables in the MRF, an important problem is to compute the most likely assignment of values for the other unobserved variables.

We model the keyword generation problem using an MRF as follows. Consider the random variables ℓ_q and ℓ_u that represent the concept label for each query q and URL u, respectively. The domain of these variables is the set of concept labels $\mathcal{C}$. The pairwise relationships between the nodes are defined by the edges derived from the click graph, and their strength depends on their weight. The seed set $\mathcal{S}$ defines the observations for the variables. The observations in our case are of the form $P(\ell_u = c) = 1$ for all pairs $\langle u, c\rangle$ in the seed set. We are interested in assigning concepts to the queries and the unlabeled URLs in the graph in such a way that we respect the constraints defined by the click graph.

In the MRF model this is translated to finding the assignment of values to the random variables such that the posterior probability $P(\{\ell_q\}, \{\ell_u\}|\mathcal{S})$ is maximized. The main assumption is that given the Markovian assumption we can decompose the joint probability distribution to factors defined over the edges in the graph. For some edge (q, u) in the MRF, we define a compatibility function $\psi_{qu}(\ell_q, \ell_u)$ (also called *potential function*) that scores the relationship between q and u. The choice of potential function typically reflects the prior knowledge one has about the problem under consideration. As noted earlier, in our case a click to a URL from a query can be viewed as an endorsement of similar concept. Hence, a larger click count implies a stronger agreement of concepts between the connected nodes. The definition of ψ will be made to reflect this prior belief about concept agreement and click counts.

Given the potential function we can now succinctly write

the joint probability distribution of the MRF as follows:

$$P(\{\ell_q\},\{\ell_u\},\mathcal{S}) \propto \prod_{(q,u)\in E} \psi_{qu}(\ell_q,\ell_u) \prod_{s\in\mathcal{S}} \phi_s(\ell_s;k) \qquad (3)$$

where $\phi_s(\ell_s;k) = P(\ell_s = c_k)$, and $P(\ell_s = c_k) = 1$ if $\langle s, c_k\rangle \in \mathcal{S}$.

The exact definition of the MRF depends on the choice of the actual potential function, which dictates the way that the probabilities are computed. However, given a specific potential function $\psi_{qu}(\ell_q,\ell_u)$, finding the optimal label assignment is NP-hard for graphs that have cycles, which is the case with the click-log graph. Thus, approximate inference algorithms are required. In the next sections we describe two different ways for defining the potential function, and performing approximate inference. The first approach leads to the absorbing random walk algorithm we described in the previous section.

4.2 Gaussian Markov Random Fields

One way to relax the MRF inference problem is to assume that instead of having discrete labels $\mathcal{C}$ we have continuous ones. For simplicity assume that $\mathcal{C} = \{0,1\}$, that is we have only two classes and we want to assign each query q and URL u to one of those classes. The continuous relaxation assumes that $\mathcal{C} = [0,1]$, that is the class labels ℓ_q and ℓ_u are now real numbers in the $[0,1]$ interval. These continuous labels can then be converted to discrete ones by rounding them to the closest discrete value.

Given this relaxation assumption, we can now work within the *Gaussian* Markov Random Field model. The potential function $\psi_{qu}(\ell_q,\ell_u)$ is now defined as follows

$$\psi_{qu}(\ell_q,\ell_u) = \exp(-f_{qu}(\ell_q-\ell_u)^2)$$

The joint distribution of the MRF can be written as

$$\begin{aligned} P(\{\ell_q\},\{\ell_u\},\mathcal{S}) &\propto \prod_{(q,u)\in E} \exp(-f_{qu}(\ell_q-\ell_u)^2) \\ &= \exp(\sum_{(q,u)\in E} -f_{qu}(\ell_q-\ell_u)^2) \end{aligned}$$

where $\ell_u = 1$ for all seed URLs $u \in \mathcal{S}$. We have that

$$P(\{\ell_q\},\{\ell_u\}|\mathcal{S}) = \frac{P(\{\ell_q\},\{\ell_u\},\mathcal{S})}{P(\mathcal{S})}$$

where $P(\mathcal{S})$ is the probability of the observations in the seed set. Since $P(\mathcal{S})$ is independent of the labels $\{\ell_q\}$ and $\{\ell_u\}$, maximizing $P(\{\ell_q\},\{\ell_u\}|\mathcal{S})$ is the same as maximizing $P(\{\ell_q\},\{\ell_u\},\mathcal{S})$. Finding the label assignment that maximizes the probability $P(\{\ell_q\},\{\ell_u\},\mathcal{S})$ is the same as minimizing

$$E = -\log P(\{\ell_q\},\{\ell_u\},\mathcal{S}) \propto \sum_{(q,u)\in E} f_{qu}(\ell_q-\ell_u)^2$$

The quantity E is often referred to as the *energy* of the Markov Random Field. This minimization criterion requires that for edges (q,u) with large weight f_{qu} the labels ℓ_q and ℓ_u should be close, so that the value $f_{qu}(\ell_q-\ell_u)^2$ is minimized. This is intuitive; a large number of clicks f_{qu} implies a strong association between the query q and URL u, and thus their labels should be similar. This effect is a direct consequence of the choice of the potential function which was chosen so as to penalize discrepancy between the labels of edges with large weight.

Finding the optimal labeling is now a tractable problem. Zhu et al. [27] demonstrate that the optimal labeling is *harmonic* and the optimal value can be found by iteratively setting each label value to be the weighted average of the labels of its neighbors. Note that this is exactly the property satisfied by the absorbing random walk algorithm we described in the previous section. The probability of a query to belong to a class is computed as the weighted average of the probabilities of its neighbors. The harmonic property guarantees that the iterative computation leads to a unique solution. Thus the ARW algorithm can be naturally viewed as an inference algorithm in the Gaussian Markov Random Field.

4.3 Variational Inference and the Mean Field Algorithm

For comparison purposes we also consider a different Markov Random Field formulation, and a different algorithm for computing the posterior distribution. Unlike the Gaussian Markov Random Field, in this case we consider the labels to be discrete. To model the fact that for edges with large weight we want the assigned labels to agree, we set the potential function to be

$$\psi(\ell_q,\ell_u) = \begin{cases} \exp(\lambda f_{qu}), & \text{if } \ell_u = \ell_q \\ \alpha, & \text{if } \ell_u \neq \ell_q \end{cases}$$

where α is a constant. Therefore, the potential function rewards agreement in class for query-URL pairs when there is a large number of clicks.

As we have already argued, finding the optimal label configuration that maximizes the posterior probability is intractable. In the previous section, we addressed this problem by relaxing the labels. Here, we relax our requirements on the posterior distribution, and we approximate it by one that is easier to compute. This approach falls under the general framework of *variational inference*.

Variational inference is an important class of approximate inference algorithms [13]. The goal of variational inference is to approximate the true intractable posterior distribution P with a tractable distribution $\hat{P}$ that has a simpler form. The parameters of the approximating $\hat{P}$ distribution are computed by directly optimizing a similarity measure between the approximating and the true distribution. A common choice for the (dis)similarity measure is the Kullback-Liebler divergence (KL-divergence). The KL-divergence between two distributions $\hat{P}$ and P is defined as

$$\mathrm{K}\left[\hat{P}\|P\right] = \sum_{\boldsymbol{x}} \hat{P}(\boldsymbol{x}) \log \frac{\hat{P}(\boldsymbol{x})}{P(\boldsymbol{x})} \qquad (4)$$

For the variational inference algorithm described here, we define the approximate distribution to take a fully factored form

$$\hat{P}(\{\ell_q\},\{\ell_u\}|\mathcal{S}) = \prod_q \hat{P}_q(\ell_q) \prod_u \hat{P}_u(\ell_u) \qquad (5)$$

This is also known as the *Mean Field approximation*. The distributions $\hat{P}_q$ and $\hat{P}_u$ in the above equation model the posterior over queries and URLs respectively; more specifically, from the point of view of parametrization, $\hat{P}_q$ and $\hat{P}_u$ are multinomial distributions that define the posterior over categories. For example, if there are k categories $\{c_1, c_2, \ldots c_k\}$,

then the parametrization of $\hat{P}_q$ will have k numbers, $\hat{P}_{qi} = \hat{P}_q(\ell_q = c_i)$ such that $\sum_i \hat{P}_{qi} = 1$.

We now seek to estimate the parameters of the approximating distributions by minimizing the KL divergence between it and the true posterior. The minimization will be performed with respect to every parameter of the $\hat{P}$ distribution. The KL-divergence can be computed as follows.

$$\begin{aligned}
\mathrm{K}\left[\hat{P}\|P\right] &= \sum_{\ell_q,\ell_u} \hat{P}(\ell_q,\ell_u)\log\frac{\hat{P}(\ell_q,\ell_u)}{P(\ell_q,\ell_u|\mathcal{S})} \\
&= \sum_{\ell_q,\ell_u} \hat{P}(\ell_q,\ell_u)\log\frac{\hat{P}(\ell_q,\ell_u)P(\mathcal{S})}{P(\ell_q,\ell_u|\mathcal{S})P(\mathcal{S})} \\
&= \sum_{\ell_q,\ell_u} \hat{P}(\ell_q,\ell_u)\log\frac{\hat{P}(\ell_q,\ell_u)}{P(\ell_q,\ell_u,\mathcal{S})} + \log(P(\mathcal{S})) \\
&= -H(\hat{P}) - \sum_{\ell_q,\ell_u} \hat{P}(\ell_q,\ell_u)\log P(\ell_q,\ell_u,\mathcal{S}) + \log P(\mathcal{S}) \\
&= E + \log P(\mathcal{S}) \qquad (6)
\end{aligned}$$

The first term E in Equation 6 is generally referred to as the mean field free energy[4], the intuition being low energy configurations are more stable (and likely). The second term $\log P(\mathcal{S})$ is the log probability of observations, and does not depend on the parameters of the $\hat{P}$ distribution; hence we can safely ignore the term while performing the minimization and work only with E. To ensure that $\sum_c \hat{P}_q(\ell_q = c) = 1$ for all queries q (and similarly for $\hat{P}_u$) we need to introduce appropriate Lagrangian multipliers during the minimization. The minimization is performed by doing gradient descent on the parameter space $\{\hat{P}_{qi}\}, \{\hat{P}_{qi}\}$

The expansion of the mean-field energy term

$$E = -H(\hat{P}) - \sum_{\ell_q,\ell_u} \hat{P}(\ell_q,\ell_u)\log P(\ell_q,\ell_u,\mathcal{S})$$

results in two terms, one that is independent of observations $\mathcal{S}$, and the other encoding it. The former corresponds to the (negative) entropy of the approximating distribution and the latter can be associated with the expectation of the joint distribution with respect to the approximating distribution. During the minimization, the first terms favors a $\hat{P}$ distribution with high entropy, while the evidence term, which encodes $\mathcal{S}$, tends to explain the observations by pulling $\hat{P}$ away from the uniform distribution. The updates for the $\hat{P}_q$ and the $\hat{P}_u$ distributions will be coupled whenever there is an edge between the particular query and URL. Using the fact that the distribution $P(\{\ell_q\},\{\ell_u\},\mathcal{S})$ for a given configuration readily factors into a product of pairwise potentials, we can now write the update equations as follows.

$$\log \hat{P}_q(\ell_q) = \sum_{u:(q,u)\in E} \sum_c \hat{P}_u(\ell_u = c)\log\psi_{qu}(\ell_q,\ell_u) - 1 + \lambda_q$$

$$\log \hat{P}_u(\ell_u) = \sum_{q:(q,u)\in E} \sum_c \hat{P}_q(\ell_q = c)\log\psi_{qu}(\ell_u,\ell_q) - 1 + \lambda_u$$

where λ_q and λ_u are the Lagrangian multipliers that normalize the resulting distributions. Iterating yields a local minimum, and we obtain a distribution over the class labels.

[4]Different from the Markov Random Field energy we minimized before.

5. EXPERIMENTS

We now present an experimental evaluation of our approach, centered around the ARW algorithm. In Section 5.1, we present the experimental setup. In Section 5.2, we study the properties of the ARW algorithm and the quality of the results that it returns. Finally, in Section 5.3, we compare the ARW algorithm with other algorithms, including the Mean Field algorithm presented in Section 4.3.

5.1 Experimental setup

The query click graph. We perform experiments on a click graph constructed from a snapshot of the query log obtained from a major search engine. The graph has 41 million queries, 55 million URLs, and 93 million edges. The URLs correspond to both clicked documents and ads. The query-URL pairs account for 490 million clicks in the log. Each edge has a frequency of at least two: we pruned all query-URL pairs with just one click in the snapshot, since we considered them to be too rare to define a meaningful association.

Seed sets. We use three different seed sets that are motivated by specific scenarios. In the first scenario, the seed set is created by just specifying the domain of the advertiser's Web site. We consider the scenario in our motivating example in Section 1 where an advertiser is interested in promoting the online shoe store `www.shoes.com`. The goal of the advertiser is to find queries that are related to her domain, and could potentially result in clicks to her advertisement. Therefore, the URLs in the domain `shoes.com` can be thought of as a representation of the abstract concept "shoes" that the advertiser is interested in. We construct the seed set by obtaining all the URLs in the domain (i.e., all URLs that start with `www.shoes.com`) that appear in the click log, and we label them with the "shoes" concept. We call this seed set `Shoes`.

In the second scenario, the advertiser leverages publicly-accessible data to produce the seed set. Consider an advertiser who is interested in promoting a health-specific site, that is, she is interested in the abstract concept of "health". The advertiser is interested in capturing queries related to her site, but also queries that go to sites which are similar to hers and are authoritative in the health domain. In this case, the seed set can be obtained by resorting to a collection of popular health sites. For our experiment, we use the 12 most popular sites in the health domain obtained from Nielsen NetRating[5]. The sites are the following:

`health.yahoo.com, webmd.com, health.msn.com, kidshealth.org, www.prevention.com, www.cdc.gov, www.mayoclinic.com, familydoctor.org, emedicine.com, health.ivillage.com, parenting.ivillage.com, www.medicinenet.com`

Similarly to the `Shoes` seed set, once we have the domains we construct the seed set by obtaining all the URLs in the click log from those domains. We call this seed set `Health`.

In the last scenario, we assume that the search engine offers the advertiser a set of pre-existing concepts in the form

[5]`http://www.nielsen-netratings.com/`

query set	γ	$\lvert Q_k^i \cap Q_k^0 \rvert$	$\lvert Q_k^i \cap Q_k^0 \rvert / \lvert Q_k^0 \rvert$
Q_k^0	10^{-5}	50,000	100%
Q_k^1	10^{-4}	45,645	91.3%
Q_k^2	10^{-3}	37,338	74.7%
Q_k^3	10^{-2}	15,276	31.2%

Table 2: Effect of threshold γ for $\alpha = 0.001$ for `Shoes`

query set	α	$\lvert Q_k^i \cap Q_k^0 \rvert$	$\lvert Q_k^i \cap Q_k^0 \rvert / \lvert Q_k^0 \rvert$
Q_k^0	0.0001	50000	100 %
Q_k^1	0.001	49812	99.6 %
Q_k^2	0.01	48259	96.5%
Q_k^3	0.1	43608	87.2%
Q_k^4	0.2	41359	82.7%
Q_k^5	0.3	39564	79.1%

Table 3: Effect of α for $\gamma = 0.0001$ for `Shoes`

query set	γ	$\lvert Q_k^i \cap Q_k^0 \rvert$	$\lvert Q_k^i \cap Q_k^0 \rvert / \lvert Q_k^0 \rvert$
Q_k^0	0.0001	500000	100%
Q_k^1	0.001	456892	91.3 %
Q_k^2	0.01	393582	78.7%
Q_k^3	0.05	211391	42.3%

Table 4: Effect of threshold γ for $\alpha = 0.001$ for `Health`

query set	α	$\lvert Q_k^i \cap Q_k^0 \rvert$	$\lvert Q_k^i \cap Q_k^0 \rvert / \lvert Q_k^0 \rvert$
Q_k^0	0.0001	500,000	100%
Q_k^1	0.001	493,258	98.6%
Q_k^2	0.01	456,344	91.2%
Q_k^3	0.1	399,607	80%
Q_k^4	0.2	381,747	76.3%
Q_k^5	0.3	370,751	74.1%

Table 5: Effect of α for $\gamma = 0.0001$ for `Health`

of a taxonomy. The advertiser can select the concepts that she interested in and choose keywords from these concepts. Our objective is to populate the taxonomy with keywords by assigning queries to the different concepts. For the seed set, we assume the existence of a directory that maps URLs to some fixed taxonomy. In our experiment, we employ a directory designed for marketing purposes that is similar in spirit to the ODP directory. We focus on the third level of the taxonomy, which contains 1,158 categories. We consider each third-level category as a distinct concept, and we set the seed set for that concept to be the URLs in the corresponding category in the directory. We call this seed set `Directory`.

5.2 Experiments with the ARW algorithm

The goal of the following experiments is to evaluate the properties of the ARW algorithm and the quality of the results that it returns. Given a concept c and a seed set $\mathcal{S}$, the ARW algorithm returns a set of queries related to the concept. The queries are then sorted according to the probability of belonging to the concept, and the top-k queries Q_k are returned as candidates to the advertiser. We are interested in evaluating the answer set Q_k. We first investigate how the parameters of the ARW algorithm affect the set Q_k, and then the quality and properties of the queries in Q_k for different values of k. We use the `Shoes` and `Health` seed sets for these experiments.

Setting the ARW parameters. We begin by studying the effect of the ARW parameters α and γ on the result set Q_k. Increasing α causes query probabilities to decrease, so very large values of α, in combination with the threshold γ, will cause the size of the set Q_k to diminish. However, it is not clear what the effect of a small increase of α is on the *ranking* produced by the query probabilities, and thus the contents of Q_k. Similarly, increasing γ will cause some more nodes to be pruned, but it is not clear what effect it will have on the high-probability queries.

We study the effect of the threshold value γ for the `Shoes` seed set. We set α to 0.001, we run the ARW algorithm for different values of γ, and we observe the change in the result set Q_k for $k = 50$K. The results are presented in Table 2. In this table, we consider query sets $Q_k^0, \ldots, Q_k^3$ corresponding to values of γ ranging from 10^{-5} to 10^{-2}. For each query set Q_k^i, we show the size of the intersection with Q_k^0, and the fraction of Q_k^0 that it represents. We can observe in Table 2 that if we decrease γ by one order of magnitude (from 10^{-5} to 10^{-4}), we still retain 91.3% of the queries in the intersection. If we decrease it by two orders of magnitude (from 10^{-5} to 10^{-3}) the intersection still contains 74.7% of the queries in Q_k^0. For $\gamma = 10^{-2}$, the intersection drops to 31.2%. However, for this high value of the threshold, the algorithm retrieves only 15,360 queries, 99.4% of which intersect with Q_k^0.

We perform a similar study for the effect of the parameter α. We set the threshold γ to 10^{-4} and run the ARW algorithm on the `Shoes` seed set for different values of α. For each run we consider the top $k = 50$K queries. The results are shown in Table 3. We denote the different query result sets as $Q_k^0, \ldots, Q_k^5$, corresponding to values of α ranging from 0.0001 to 0.3. As before, for each query set Q_k^i, we show the size of the intersection with Q_k^0, and the fraction of Q_k^0 that it represents. We observe in the table that if we decrease the threshold by one order of magnitude (from 0.0001 to 0.001), we still retain 99.6% of the queries. Even if we decrease the threshold by three orders of magnitude (from 0.0001 to 0.1), the intersection is still 87.2% of the top 50K queries.

We observe very similar trends for the `Health` seed set. The results are shown in Tables 4 and 5. We can thus conclude that there is a wide range of values for γ and α for which the query set Q_k returned by the algorithm remains relatively unchanged. The algorithm is robust to the changes of the parameters. This is not surprising since we expect nodes that are close to the seed set to be in the top-k results. The ranking of these nodes should not be significantly affected by moderate changes to α or γ. This has implications for the efficiency of the algorithm as well, since we can pick relatively high values for the parameters and speed-up the convergence of the algorithm, while retaining consistent results.

Query set evaluation. We now evaluate the quality of the query set Q_k produced by the ARW algorithm. We consider the following two metrics for evaluating the results.

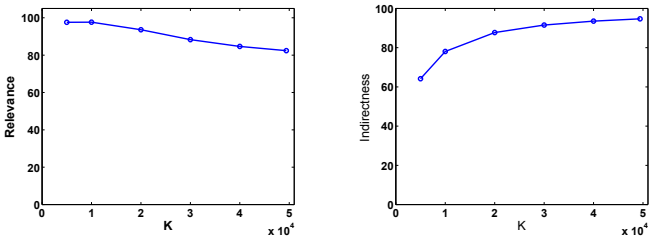
- **Relevance:** We compute the relevance ratio $R(Q_k)$ of the query set Q_k as the fraction of relevant queries in the set S. The evaluation of queries was done by human evaluators. In each experiment, the judges were

k	Relevance $R(Q_k)$	Indirectness $N(Q_k)$
5K	97.6%	64.2%
10K	97.7%	78.1%
20K	93.6 %	87.8%
30K	88.3%	91.6%
40K	84.7%	93.6%
50K	82.4%	94.7%

Table 6: Relevance and indirectness for `Shoes`

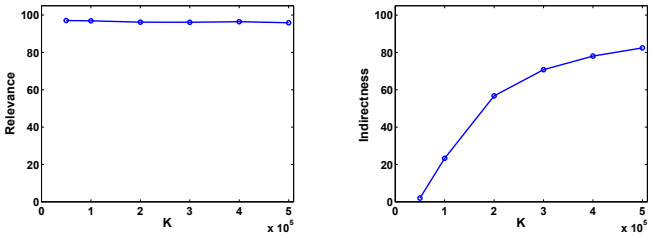
k	Relevance $R(Q_k)$	Indirectness $N(Q_k)$
50K	97.1%	2.0%
100K	96.9%	23.3%
200K	96.2%	56.7%
300K	96.1%	70.8%
400K	96.4%	78.1%
500K	95.9%	82.4%

Table 7: Relevance and indirectness for `Health`



(a) relevance $R(Q_k)$ (b) indirectness $N(Q_k)$

Figure 1: Plots for evaluation metrics for `Shoes`



(a) relevance $R(Q_k)$ (b) indirectness $N(Q_k)$

Figure 2: Plots for evaluation metrics for `Health`

presented with a description of the concept and a set of queries, and they were instructed to consider the queries as "good" if they could find a relationship between the query and the concept, and "bad" otherwise.

- **Indirectness:** Given the seed set $\mathcal{S}$, let $Q(\mathcal{S})$ denote the set of queries that are directly connected to $\mathcal{S}$. We say that a query q is *indirect*, if $q \notin Q(S)$. We denote with $N(Q_k)$ the fraction of indirect queries in a result set Q_k. Indirect queries are of particular interest when the seed set contains URLs associated to the advertiser's site, such as in the `Shoes` scenario, since the advertiser may be interested in discovering relevant queries that have not been generating traffic to her site.

We first study the trade-off between the size of the query result set and its relevance. For each seed set, we run a random walk with $\alpha = 0.001$ and $\gamma = 10^{-4}$. For each result set, we evaluate the relevance of the top-k queries, for different values of k. The relevance results for `Shoes` and `Health` are shown in Tables 6 and 7, respectively. We also plot them in Figures 1(a) and 2(a), respectively. For the `Shoes` experiment, the relevance decreases with the value of k, but even for the top 50K queries it is reasonably high (82.4%). For the `Health` experiment, the relevance remains relatively constant as we increase k, and it is 95.9% for the largest result set that we consider ($k = 500K$).

We then consider how indirectness changes for different values of k. The results for `Shoes` are given in Table 6, and plotted in Figure 1(b). Indirectness increases considerably when k goes from 5K to 20K (from 64.2% to 87.8%), and then continues to increase, but more slowly. Indirectness of 87.8% can be achieved with a relevance of 93.6%. Table 7 and Figure 2(b) show the indirectness values for the `Health` seed set. Indirectness increases sharply from $k = 50$K to $k = 500$K. Initially, only 2% of the queries are indirect, while for $k = 500$K, the percentage of indirect queries is 82.4%. This increase in indirectness does not come at the expense of relevance, which is 97.1% in the former case and 95.9% in the latter. These numbers indicate that we can obtain high quality results even when we consider queries that are not directly connected to the seed set in the click graph.

5.3 Comparison with other algorithms

In this section we compare the ARW algorithm with two other algorithms. The first is the Mean Field algorithm as it is described in Section 4.3. The second is an algorithm that categorizes a query by grouping the snippets of the returned URLs into a *snippet document*, and classifying the snippet document. This method was first employed by Broder et al. [6]. For the classification of the snippet document we use a Rocchio classifier [18], trained using the content of all the documents in the `Directory` seed set. The snippet document is constructed by merging the snippets of the top-40 documents, a number that was shown to perform well by Broder et al. [6]

Both algorithms we compare against are better defined for the case that there are multiple concepts, so for this experiment we use the `Directory` seed set. We measure relevance by sampling from the results and evaluating the query-category pairs. For the evaluation of individual pairs, we consider the third level of the taxonomy. Given the large size of the taxonomy, for presentation purposes we aggregate the results into the 20 top-level categories of the taxonomy. We present micro-averaged relevance [25] for each category. The micro-averaged relevance for a high level category c is computed using the query-category pairs $\langle q, c' \rangle$, where c' ranges over all subcategories of c. The micro-average relevance of category c is the fraction of such pairs that are judged as "good". Notice that even though we aggregate the values at the top level of the taxonomy, the queries are actually evaluated at the third-level. Table 8 summarizes

Category	Micro-averages ARW	Mean Field	Snippets
Food and Drinks	**99%**	84.6%	92%
Social Sciences and Humanities	**91.3%**	35%	78%
Computers and Computing	**89.6%**	71.8%	70.1%
Health and Wellness	**97.8%**	83.8%	84.9%
Vehicles and Transportation	83.3%	**90.9%**	**90.9%**
Lawn and Garden	**96.6%**	66.7%	60%
Travel	94.7%	80%	**96.3%**
Sports and Recreation	**91.1%**	51.9%	76.7 %
Financial Services	**96.7%**	85.4%	82.3%
Arts and Entertainment	**88.1%**	78.7%	78.6%
Games Puzzles	94.2%	**94.7%**	77.8%
Kids and Teens Lifestyle	**89.3%**	77.3%	72.2%
Society and Culture	67.1%	54.5%	**78.7%**
Business	**87.2%**	72.1%	70.9%
Clothing and Shoes	**98.4%**	95.5%	95.7%
Science	86.2%	57.3%	**86.3%**
Educational Institutions	**87.2%**	34.4%	61%
Families and Relationships	**94.1%**	90.9%	80%
Animals	94.2%	90.9%	**100%**
Home Improvement	**96.8%**	84.2%	93.8%
Micro-average	**88.2%**	69%	80.2%

Table 8: Micro-average relevance. Aggregated for top-level categories of the taxonomy, evaluated for third-level categories.

the results. The first column lists the 20 top-level categories in the taxonomy. The second column corresponds to the ARW algorithm with parameters $\alpha = 0.01$ and $\gamma = 0.001$; the third column to the Mean Field algorithm; and the last column to the snippets-based algorithm.

The ARW algorithm has the highest micro-average relevance (88.2%, as opposed to 80.2% for the Snippets algorithm). Notice that although the Snippets algorithm has a lower micro-average relevance than the ARW on the entire sample, it has higher micro-average relevance for 5 out of the 20 classes that we consider. This suggests that, as future work, it may be fruitful to consider an approach that exploits both the content of documents and their clicks. On the other hand, the Mean Field algorithm has a lower micro-average relevance of 69%. It produces the best results for two of the categories, but usually it exhibits the lowest relevance ratio. It appears that the ARW algorithm benefits significantly by the introduction of the null category, and the exponential decay in the probability of reaching a node in the seed set. This effect is not modeled well in the case of the Mean Field algorithm, which results in poor performance in the case that there are weak connections between queries and URLs of non-relevant classes. Understanding fully how the Mean Field algorithm can be optimized is also an interesting problem for future work.

This experiment offers also an interesting insight into how the ARW algorithm performs when the seed set contains multiple concepts, as well as concepts of fine granularity. The third level of the taxonomy contains 1158 categories, some of which are fairly narrow. For example, under the top-level category "Arts and Entertainment", there are third-level categories such as "Movies/Film Festivals", "Movies/Awards", "Movies/Filmaking" and "Movies/Theaters". The micro-average relevance that we obtained indicates that the algorithm produces relevant results even for fine-grained categories of the taxonomy.

6. RELATED WORK

Most of the work on keyword generation has focused on extracting keywords from documents. Turney [20] proposed GenEx, a rule-based keyword extraction system tuned using a genetic algorithm. Another well-known keyword extraction system is KEA [10], which employs a naive Bayes learning algorithm. Later work added web document related features to KEA, such as the number of documents returned by a search engine [21] and link information [15]. The use of natural language techniques for keyword extraction was initially studied by Hulth [12]. Yih et al. [26] proposed a system for extracting keywords from Web pages for contextual advertising. Among other features, they use the frequencies of candidate keywords in the query log. However, they do not employ click information (i.e., relationships between queries and documents). These techniques for keyword extraction from documents can be viewed as complementary to ours. In particular, we can consider the keywords extracted from the given document as a seed set (of queries) for our click-based algorithm.

Some approaches in the literature exploit the results returned by a search engine instead of the user clicks [1, 14]. The idea is to start with a seed set of keywords, submit them to the search engine and then use the retrieved text snippets or documents to extract relevant keywords. Joshi and Motwani [14] introduced a notion of non-obviousness which, though similar in spirit to the notion of indirectness presented in our work, is defined quite differently. In particular, they assume that a set of keywords is given as input, and a term is considered non-obvious if it does not contain any of the input keywords. To the best of our knowledge, the only previous approach to keyword generation that employs the click graph is by Bartz et al. [4]; however, their techniques are quite different from ours (they employ logistic regression and collaborative filtering techniques).

The techniques presented in this paper can be used to populate queries within a taxonomy if a directory of URLs is used to represent the concepts. There are a number of proposals in the literature that obtain a query category by classifying the search results of the query (either documents or snippets) [6, 19]. In contrast, we use clicks, and we do not need to crawl content in order to train a classifier. More related to our approach is the work of Xue et al. [23], which considers the combination of the signal coming from document content and the click logs, and shows an improvement over pure content-based classification methods.

The click graph has been used extensively for other applications. Some related approaches include the following. Beeferman and Burger [5] and Wen et al. [22] employ clustering techniques to determine query-to-query similarity. Xue et al. [24] use the click graph to find document-to-document similarities. Craswell and Szummer [8] consider random walk techniques on the click graph to produce a ranking of documents for image search. Baeza-Yates and Tiberi [3] use the click graph to extract semantic relationships between queries.

In our approach, we start with a seed set (of queries or URLs) and expand it in order to generate keywords. The seed set expansion problem has been considered in other contexts such as detecting hubs and authorities [16], community discovery [2], and detecting spam pages [11].

7. CONCLUSIONS

We have introduced an approach to keyword generation that leverages the information available in the search engine click logs. Our approach requires minimal effort from the part of the advertisers. In some cases, it just suffices to provide one domain in order to produce large lists of relevant keywords. Promising experimental results demonstrate that our algorithms can scale to large query logs and produce high-quality results.

There are several directions for future work. One important direction is to consider richer click graphs, with additional attributes such as dwell times, and position of clicked documents in the click graph. We can also add other types of edges to the graph. For example, the query reformulations made by users induce edges between queries; and hyperlinks induce edges between URLs. In addition to using click information, we could also use the impressions (i.e., the results that are shown to the users). This may yield valuable negative information: a "non-click" to a URL may be an indication of the fact that it is not relevant to the query. Another interesting direction is to consider the content of Web pages. We could use a standard document classifier to classify some of the documents, and use the output of the classifier as priors in the click-based algorithm. The query string could be used as well; if a query is not in the output of our algorithm, we could consider approximate matching techniques to find a similar query which is in the output.

Finally, it would be interesting to study the effect of spam on the keyword generation techniques. There are two types of spam that should be taken into account: spurious clicks (usually from bot agents) and spam Web pages, each one affecting the algorithm in a different way.

Acknowledgements: We would like to thank Marc Najork for providing the tools that we used for storing and processing the click logs, and Alan Halverson for his help in using these tools. We are also grateful to Vu Ha and Lingfeng Wu for providing the Rocchio classifier used in the Snippets algorithm, and Alex Ntoulas for his help in setting up a Web interface to evaluate the queries.

8. REFERENCES

[1] V. Abhishek and K Hosanagar. Keyword generation for search engine advertising using semantic similarity between terms. *International Conference on Electronic Commerce*, pages 89–94, 2007.

[2] R. Andersen and K. Lang. Communities from seed sets. *WWW*, pages 223–232, 2006.

[3] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. *KDD*, pages 76–85, 2007.

[4] K. Bartz, V. Murthi, and S. Sebastian. Logistic regression and collaborative filtering for sponsored search term recommendation. *Second Workshop on Sponsored Search Auctions*, 2006.

[5] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. *KDD*, pages 407–416, 2000.

[6] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. *SIGIR*, pages 231–238, 2007.

[7] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 2006.

[8] N. Craswell and M. Szummer. Random walks on the click graph. *SIGIR*, pages 239–246, 2007.

[9] P. Doyle and L. Snell. *Random Walks and Electrical Networks*. Mathematical Association of America, 1984.

[10] E. Frank, G. Paynter, I. Witten, C. Gutwin, and C. Nevill-Manning. Domain-specific keyphrase detection. *Proc. of IJCAI*, pages 668–673, 1999.

[11] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. *VLDB*, pages 576–587, 2004.

[12] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. *EMNLP*, pages 216–223, 2003.

[13] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, Norwell MA., 1998.

[14] A. Joshi and R. Motwani. Keyword generation for search engine advertising. *ICDM Workshops*, pages 490–496, 2006.

[15] D. Kelleher and S. Luz. Automatic hypertext keyphrase extraction. *IJCAI*, pages 1608–1609, 2005.

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

[17] S. Li. *Markov random field modeling in computer vision*. Springer-Verlag, 1995.

[18] J. Rocchio. *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter Relevance feedback in Information Retrieval, pages 313–323. Prentice Hall, 1971.

[19] D. Shen, R. Pan, J. Sun, J. Pan, K. Wu, J. Yin, and Q. Yang. Q2C@UST: Our winning solution to query classification in KDDCUP 2005. *SIGKDD Explorations*, 7:100–110, 2005.

[20] P. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

[21] P. Turney. Coherent keyphrase extraction via web mining. *IJCAI*, pages 434–439, 2003.

[22] J. Wen, J. Nie, and H. Zhang. Clustering user queries of a search engine. *WWW*, pages 162–168, 2001.

[23] G. Xue, Y. Yu, D. Shen, Q. Yang, H. Zeng, and Z. Chen. Reinforcing web-object categorization through interrelationships. *Data Mining and Knowledge Discovery*, 12:229–248, 2006.

[24] G. Xue, H. Zeng, Z. Chen, Y. Yu, W. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. *CIKM*, pages 118–126, 2004.

[25] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999.

[26] W. Yih, J. Goodman, and V. Carvalho. Finding advertising keywords on web pages. *WWW*, pages 213 – 222, 2006.

[27] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *ICML*, pages 912–919, 2003.