Exploiting User Clicks for Automatic Seed Set Generation for Entity Matching

Xiao Bai *
Yahoo! Research
Barcelona, Spain
xbai@yahoo-inc.com

Flavio P. Junqueira *
Microsoft Research
Cambridge, United Kingdom
fpj@apache.org

Srinivasan H. Sengamedu *
Komli Labs
Bangalore, India
srisrishs@yahoo.com

ABSTRACT

Matching entities from different information sources is a very important problem in data analysis and data integration. It is, however, challenging due to the number and diversity of information sources involved, and the significant editorial efforts required to collect sufficient training data. In this paper, we present an approach that leverages user clicks during Web search to automatically generate training data for entity matching. The key insight of our approach is that Web pages clicked for a given query are likely to be about the same entity. We use random walk with restart to reduce data sparseness, rely on co-clustering to group queries and Web pages, and exploit page similarity to improve matching precision. Experimental results show that: *(i)* With 360K pages from 6 major travel websites, we obtain 84K matchings (of 179K pages) that refer to the same entities, with an average precision of 0.826; *(ii)* The quality of matching obtained from a classifier trained on the resulted seed data is promising: the performance matches that of editorial data at small size and improves with size.

Categories and Subject Descriptors

I.5.3 [**Clustering**]: Algorithms; H.2.8 [**Database Applications**]: Data mining

Keywords

Entity matching, user clicks, random walk, co-clustering

1. INTRODUCTION

Entity matching is an important problem in data analysis and data integration. We focus in this paper on the entity matching problem that aims to determine whether two (or more) references to an entity *from different information sources* describe the same real-world object. This problem is important since multiple organizations may describe the same entity using various descriptions. Matching such references gives information providers the opportunities to improve their services by *(i)* enriching the description of an entity with more comprehensive information; and *(ii)* providing comparison of various descriptions of an entity, especially when inconsistent descriptions exist. For instance, travel websites like Yahoo! Travel[1] and TripAdvisor[2] only provide very brief descriptions of tourist attractions, hotels, *etc.* Matching pages from such sites with Wikipedia pages that refer to the same entities may help these sites to automatically generate more comprehensive descriptions of entities. Moreover, when users look for tourist attractions in Paris, "Musee de l'Orangerie" receives very high rating compared to most of the well-known attractions like "Eiffel Tower" and "Musee du Louvre" in TripAdvisor. Matching pages from Tripadvisor with pages from other travel websites that refer to the same entities may help users to realize that the high rating of "Musee de l'Orangerie" in TripAdvisor is mainly provided by art lovers rather than ordinary tourists. In fact, matching entities from different sources is also important for on-line comparison shopping sites like PriceGrabber[3], NextTag[4], *etc.* to gather and compare the prices, descriptions and reviews of the same products from different on-line shopping sites like Amazon, Walmart, *etc.*, which helps users to determine where to purchase a product.

However, the lack of unique entity identifiers across different information sources, the large amount of information sources (*e.g.*, travel, shopping websites, *etc.*), and the larger amount of entities (*e.g.*, tourist attractions, products, *etc.*) make it very challenging to efficiently match the multiple references to the same entity. Existing entity matching approaches mainly rely on machine learned classifiers (*e.g.*, SVM [3], decision tree [21], *etc.*) to determine whether a pair of entities match not. The performance of these supervised learning approaches highly depends on the quality and size of the available training data. Since labeling training data usually requires editorial efforts, it is very expensive and inefficient to produce large-scale training data to ensure accurate entity matching.

In this paper, we propose an unsupervised approach for matching entities. More precisely, we propose an unsupervised approach for generating *seed* matches which can serve as training data for supervised approaches. There are already unsupervised approaches for matching entities [20, 13], which usually determine whether two references are likely

*Authors are ordered alphabetically.

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

[1] http://travel.yahoo.com/
[2] http://www.tripadvisor.com/
[3] http://www.pricegrabber.com/
[4] http://www.nextag.com/

to refer to the same entity according to their similarity on a set of attributes. However, as information sources are usually maintained by different organizations, it is not always obvious to access the entities and identify the corresponding attributes beforehand. Differently, we leverage the click behavior of Web search users to generate matching entities across websites. The basic intuition of the approach is the observation that users looking for an entity like "Eiffel Tower" usually click on pages of the same entity from different websites in response to search queries. Therefore, it is possible to link pages of the same entity across sites based on user click behaviors. This approach allows matching entities from different sources without knowing any specific attribute associated to them, and is thus more flexible.

However, there are a few challenges with this approach. First, users may use different keywords for searching the same entity. Consequently, for robustness, we need to both correlate search queries and match pages using search queries. Second, clicks of search users are generally sparse as users often click on a very limited number of pages in response to their queries. This sparseness is both due to users only clicking on pages appearing relevant to their queries and to users not clicking on any page if the titles and snippets of the search results already provide sufficient information.

Contributions: Our main contribution in this work is an unsupervised approach for identifying matching of entities based on user click behaviors in search. More concretely:

- We leverage the connectivity of pages and queries to generate matchings of pages that refer to same entities.
- We take advantage of the duality of queries and pages, *i.e.*, queries can be clustered on the basis of pages that they co-click while pages can be clustered on the basis of queries that lead to co-click on them, to increase the number of matchings obtained through co-clustering.
- We rely on random walk with restart to discover plausible missing clicks. This reduces the sparseness of user behavior and is key for the matching effectiveness.
- We exploit the similarity among pages to refine ambiguous matchings, increasing the number of matchings obtained with high precision.
- We evaluate our approach with a large-scale real-world dataset. Half of the targeted pages involved in user clicks are matched among each other with an overall precision of 0.826 in a few minutes, confirming the effectiveness, accuracy and efficiency of our approach.
- We use pages that are automatically matched by our approach as seed data for training a simple classifier for the entity matching task. We show that *large* automatically generated seed data results in better classifier than *small* editorially generated seed data, revealing the potential of our approach on generating large-scale seed data for supervised entity matching approaches.

The paper is organized as follows. Section 2 defines the problem. Section 3 presents our approach and analyzes its complexity. Section 4 reports its experimental performance. Section 5 surveys the related work and Section 6 concludes the work.

2. PROBLEM DEFINITION

In this work, we focus on matching Web pages from different websites that refer to the same entity by mining the Web query click logs. We use click data alone, without considering page content or query content, to ensure scalability.

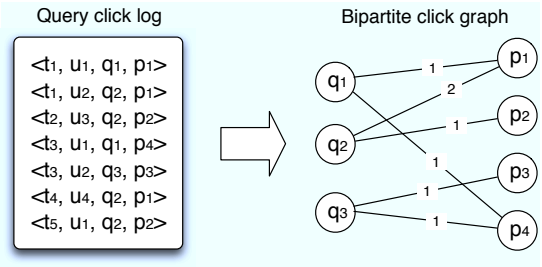


Figure 1: Generation of bipartite click graph.

Data model. We model Web query click logs as a query-page bipartite click graph $\mathcal{G} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{E} \rangle$. The queries in the logs constitute the partition $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$ and the clicked pages constitute the partition $\mathcal{P} = \{p_1, p_2, ..., p_m\}$. There is an edge $e_{i,j} \in \mathcal{E}$ between $q_i \in \mathcal{Q}$ and $p_j \in \mathcal{P}$ if query q_i leads to at least one click on page p_j. Each edge is associated with a weight $w_{i,j}$, indicating the frequency that q_i leads to a click on p_j. Figure 1 illustrates how the click graph is built based on a query click log through an example. Since page p_1 is clicked in response to query q_2 by user u_2 at time t_1 and by user u_4 at time t_4 respectively, the weight of edge $e_{2,1}$ is set to 2. We use the term "co-click" in the paper to refer to the fact that query q_i and q_k both lead to clicks on page p_j, *i.e.*, there exist both edges $e_{i,j}$ and $e_{k,j}$ in $\mathcal{E}$. We say that query q_i and q_k co-click page p_j.

Problem statement Given a bipartite click graph $\mathcal{G} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{E} \rangle$, our objective is to determine, among all the pages in $\mathcal{P}$, the subsets of pages $\mathcal{P}_{matching} = \{P_1, P_2, ..., P_z, ...\}$, where $P_z \subset \mathcal{P}$ and each page $p_j \in P_z$ refers to the same entity o_z. We call P_z a matching of pages. Note that $\cup_{\forall P_z \in \mathcal{P}_{matching}} P_z \subseteq \mathcal{P}$ as there may be pages in $\mathcal{P}$ that cannot form matching with any page. To identify the matching of pages, the algorithm that mines the click graph should be

- *effective*, so that many matchings can be extracted;
- *accurate*, so that many matchings indeed refer to the same entity.

We achieve these goals through a practical approach that exploits the click behavior of Web search users.

3. MATCHING ON CLICK GRAPH

3.1 Overview

The underlying intuition of our approach is the observation that users looking for entities such as "Eiffel Tower" usually click on pages from different websites in response to search queries of this entity. Although whether a clicked page refers to the searched entity depends on the query and its context, high level agreement among users (*i.e.*, multiple users click on the same page to respond to the same query) can be a good indicator of the true page-entity association. Moreover, if two pages are frequently clicked to respond to the same queries, they are likely to refer to the same entity. Similarly, if two queries frequently lead to clicks on the same page, they are also likely to refer to the same entity.

Based on these observations, we rely on a spectral co-clustering approach [10] to cluster the queries and pages in the bipartite click graph. Basically, this approach recursively determines the queries that lead to clicks on the same pages, which in turn determine the pages that are clicked to respond to the same queries. The key challenges to apply this approach in our context are *(i)* the sparseness of

the click graph[5] as pages and queries referring to the same entity may not be sufficiently associated to each other due the the plausibly missing clicks; *(ii)* the necessity of determining the expected number of clusters given a click graph as required by the co-clustering approach; and *(iii)* the ambiguity of the resulted clusters as some pages with similar titles (or snippets) but different content may be co-clicked when users were looking for their desired responses.

To reduce sparseness, before co-clustering the click graph, we smooth it using random walk with restart [19, 22, 23] to associate pages with more queries. A crucial aspect of smoothing is to adjust the weights of original edges with respect to additional edges. We explain this in Section 3.2.

To determine the expected number of clusters and refine the resulted clusters to form matchings of pages, we rely on an observation which we call "*one page per entity per site*" . That is *"for an entity-centric website, there is at most one page referring to a given entity".*

We use "entity-centric" to refer to websites like travel and shopping sites that provide information in forms of description and reviews on individual entities to facilitate content sharing and serving. Typically, there is only one page in such sites to describe an entity[6]. For example, for the attraction "Eiffel Tower", there is only one (English) page in Wikipedia[7] that gathers information about its different aspects, and one (English) page in TripAdvisor[8] that presents both travel information and user-generated reviews about it. In fact, as our objective is to match pages from different websites, even if some website occasionally has multiple pages for the same entity, it does impact the correctness of the matching we obtain.

We explain in Section 3.3 how we rely on the "one page per entity per site" observation to determine the number of clusters and identify matching of pages from the resulted clusters. We refer to such matchings as ClusteredMatching.

It is still possible to have clusters that contain multiple pages from the same website, since ambiguous titles (or snippets) of some pages in the search results may lead users to co-click pages that do not refer to the same entity. We call such a cluster *ambiguous* and use AmbiguousCluster to denote it. We will see in Section 4.2, one third of clusters obtained through co-clustering are ambiguous. To improve the effectiveness of our approach, *i.e.*, identifying more matchings in a click graph, we further refine AmbiguousCluster by exploiting similarity among pages. We refer to the matchings of pages obtained in this way as AdvancedMatching and detail in Section 3.4 how they are identified.

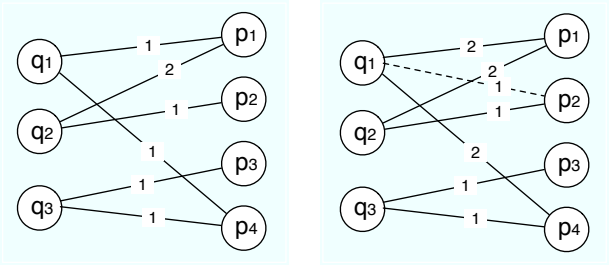
Since co-clustering only applies for connected graphs, we first use depth-first search (DFS) [8] to identify all the connected components in the click graph. If any pages in a connected component are not from the same website, they naturally form a matching that refers to the same entity. We refer to this kind of matching as SimpleMatching. Queries in each connected component naturally provide an explanation of the entity referred to by the SimpleMatching. We then smooth, co-cluster, and refine each of the click graphs corresponding to the remaining connected components to identify ClusteredMatching and AdvancedMatching, which we explain in the following sections. Figure 2 summarizes our entity matching approach.

[5]We will see in Section 4.1, in a graph of 190K queries and 360K pages, on average, each query is linked to 3.13 pages. The density of this graph is only 3.26×10^{-5} [7].

[6]We ignore different versions of a page appearing in different languages and focus on English pages.

[7]`http://en.wikipedia.org/wiki/Eiffel_Tower`.

[8]`http://www.tripadvisor.com/Attraction_Review-g187147-d188151-Reviews-Eiffel_Tower-Paris_Ile_de_France.html`.



(a) Before smoothing. (b) After smoothing.

Figure 3: Smoothing the click graph.

3.2 Smoothing the graph: random walk with restart

We smooth each connected click graph by performing random walk with restart on it to discover plausible missing clicks. Intuitively, there exists close semantic relation among neighbor vertices (*i.e.*, queries and pages) in the click graph. For example, in Figure 3(a), q_1 and q_2 both lead to clicks on page p_1, indicating that they are likely to look for the same entity. In the meantime, q_2 leads to clicks on page p_1 and p_2, indicating that p_1 and p_2 are likely to refer to the same entity. Thus, it is likely that q_1 would also lead to a click on page p_2. In other words, following the edges in the click graph, a random walk starting from q_1 has a probability of arriving at p_2 through path $q_1 \rightarrow p_1 \rightarrow q_2 \rightarrow p_2$. If this probability is high enough, we smooth the click graph by adding the edge $e_{1,2}$ (dash line in Figure 3(b)).

Formally, given a connected click graph $\mathcal{G}_c = \langle \mathcal{Q}_c, \mathcal{P}_c, \mathcal{E}_c \rangle$ ($\mathcal{Q}_c \subseteq \mathcal{Q}$, $\mathcal{P}_c \subseteq \mathcal{P}$, $\mathcal{E}_c \subseteq \mathcal{E}$), we have a $n \times m$ query-by-page matrix A where A_{ij} is the weight $w_{i,j}$ of edge $e_{i,j}$ in $\mathcal{E}$. The $(n+m) \times (n+m)$ adjacency matrix of $\mathcal{G}_c$ can be written as

$$B = \begin{bmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{bmatrix},$$

where the elements are ordered: the first n elements in each row (column) index the queries in $\mathcal{Q}_c$ and the last m elements in each row (column) index the pages in $\mathcal{P}_c$. v_i denotes a vertex in $\mathcal{G}_c$ that is either query q_i or page p_i.

We define the transition probability $p(v_j|v_i)$ from vertex v_i to vertex v_j by normalizing the clicks from v_i to v_j over all the clicks originated from v_i. We obtain the transition matrix M of one step random walk on $\mathcal{G}_c$ such that

$$M_{ij} = p(v_j|v_i) = \frac{B_{ij}}{\sum_{1 \le j \le n+m} B_{ij}}.$$

To perform a random walk with restart on $\mathcal{G}_c$, at each step, instead of always following an edge in $\mathcal{G}_c$ to walk to v_j with probability defined by M_{ij}, the random walk starting at v_i has probability c ($0 \le c \le 1$) to go back to (restart from) v_i. Intuitively, higher probability for a random walk starting at v_i to arrive at v_j implies higher probability for them to refer to the same entity. We define the similarities between vertex v_i and any other vertices in $\mathcal{G}_c$ as the $(n+m) \times 1$ steady-state probability vector $\vec{s}_i$ that satisfies

$$\vec{s}_i = (1-c)M\vec{s}_i + c\vec{r}_i,$$

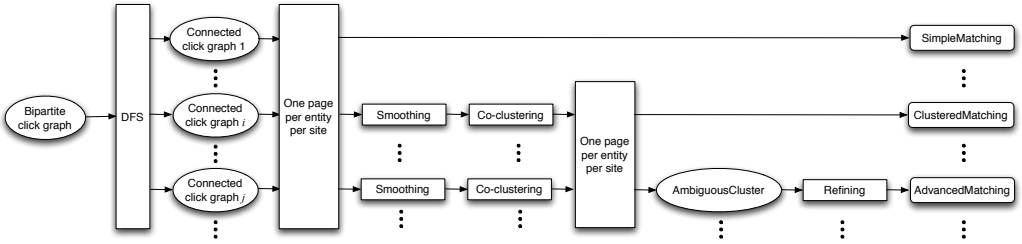


Figure 2: Overview of the entity matching algorithm in click graph.

where $\vec{r}_i$ is the $(n+m) \times 1$ restart vector that corresponds to the position for the random walk starting at v_i to restart. Thus, the value at the i^{th} position of $\vec{r}_i$ is 1 and all the other values are 0. Therefore, we have

$$\vec{s}_i = c(I_{n+m} - (1-c)M)^{-1}\vec{r}_i,$$

where I_{n+m} is a $(n+m) \times (n+m)$ identity matrix. The value of the j^{th} element in $\vec{s}_i$ corresponds to the steady-state probability of the random walk starting at v_i to arrive at v_j and thus defines the similarity between these two vertices.

More generally, the similarities between any pair of vertices in $\mathcal{G}_c$, denoted as matrix S, can be obtained with

$$S = c(I_{n+m} - (1-c)M)^{-1}.$$

The value of S_{ij} corresponds the steady-state probability of the random walk starting at v_i to arrive at v_j and defines the similarity between v_i and v_j. In fact $S = \{\vec{s}_1, \vec{s}_2, ..., \vec{s}_{n+m}\}$.

Given graph $\mathcal{G}_c$ and its steady-state probability matrix S, the sub-matrix with values S_{ij}, $1 \le i \le n$ and $n+1 \le j \le n+m$, corresponds to the probability of the random walk starting at query q_i to arrive at page p_j in $\mathcal{G}_c$. To discover plausible missing edges originating from query q_i, we iterate over all the pages p_j in $\mathcal{G}_c$ ($p_j \in \mathcal{P}_c$), *i.e.*, the values from the $(n+1)^{th}$ position to the $(n+m)^{th}$ position of $\vec{s}_i$. For a pre-defined threshold α ($0 \le \alpha \le 1$), if $S_{ij} > \alpha$ and there is no such an edge in $\mathcal{G}_c$, we add an edge between p_i and q_j in the corresponding smoothed graph $\mathcal{G}_s$. Note that $\mathcal{G}_s = (\mathcal{Q}_c, \mathcal{P}_c, \mathcal{E}_s)$ where $\mathcal{Q}_c$ and $\mathcal{P}_c$ are the same as in $\mathcal{G}_c$.

Once an edge is added in $\mathcal{G}_s$, we need to assign a weight to it. To motivate our design, we plot the distribution of weights in the original graph $\mathcal{G}$ in Figure 4 (Graph characteristics in Section 4.1). We observe that the weights of edges follow a power-law distribution. 59% edges have weight 1 and 99% edges have weight smaller than 10. Since high weight indicates high probability of referring to same entity, the additional edges should not have high weights comparing to the actual edges. Therefore, for each additional edge, we set its weight to 1. We also increment the weights of all the edges in $\mathcal{G}_c$ that start from q_i by 1 if at least one edge that starts from q_i is added in $\mathcal{G}_s$. This ensures that an actual edge always has higher weight (and higher transition probability M_{ij}) than any additional edge that starts from the same query. Figure 3(b) depicts the smoothed graph after adding edge $e_{1,2}$, where $e_{1,2}$ has weight 1 and the actual edges starting from q_1, *i.e.*, $e_{1,1}$ and $e_{1,4}$, have weight 2.

3.3 Matching the entities: co-clustering pages and queries

Once the click graph is properly smoothed, we perform co-clustering on the smoothed graph to generate clusters of pages and queries that refer to the same entities. The matchings of pages are extracted from these clusters.

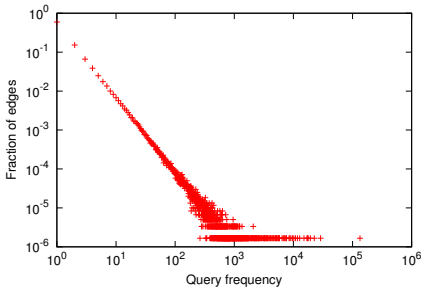


Figure 4: Distribution of weights (log-log scale).

Co-clustering pages and queries aims to partition the smoothed click graph into k partitions such that the crossing edges among partitions have minimum weights. We perform singular value decomposition (SVD) on normalized matrix A_n of A to obtain its n left singular vectors $U = \{\vec{u}_1, \vec{u}_2, ..., \vec{u}_n\}$ and m right singular vectors $V = \{\vec{v}_1, \vec{v}_2, ..., \vec{v}_m\}$. If k clusters are expected, we use $l = \lceil \log_2 k \rceil$ singular vectors $\vec{u}_2$, $\vec{u}_3$, ..., $\vec{u}_{l+1}$ to form U_k and $\vec{v}_2$, $\vec{v}_3$, ..., $\vec{v}_{l+1}$ to form V_k. We finally compute the l-dimensional data set Z_k as

$$Z_k = \begin{bmatrix} D_1^{-\frac{1}{2}} U_k \\ D_2^{-\frac{1}{2}} V_k \end{bmatrix},$$

and apply k-means algorithm to obtain the desired k clusters. We discuss the choice of k in the following as it is key to the quality of obtained clusters and matchings.

Ideally, the pages in a cluster should directly form a matching of pages that refer to the same entity. As the number of matchings that can be obtained in a click graph highly depends on the pages it contains, the value of k should be determined according to those pages. Clearly, if a graph only contains pages from one website, these pages do not form any matching of pages. Similarly, if a graph contains one page from Wikipedia and two pages from TripAdvisor, it makes no sense to obtain more than two clusters ($k \ge 3$) as a matching requires at least two pages. According to the "one page per entity per site" observation, it is also undesirable to have one cluster ($k = 1$) as once the three pages are clustered together, it requires additional refinement to keep only one page from TripAdvisor to form the matching with the Wikipedia page. Based on these observations, we use the following heuristic to determine the value of k.

Suppose the smoothed click graph $\mathcal{G}_s$ contain pages from X different websites, where each website $site_x$ contributes c_x pages to this graph (*i.e.*, $|\mathcal{P}_s| = \sum_{1 \le x \le X} c_x$). Since each website has at most one page referring to a given entity, the c_x pages from $site_x$ belong to c_x different clusters. In order to have a matching contain pages from as many different websites as possible, we set the value of k to the maximum

Algorithm 1 Co-clustering on smoothed click graph

Input: smoothed click graph $\mathcal{G}_s$
Output: a set C of clusters of pages and queries

build adjacency matrix A from $\mathcal{G}_s$
compute D_1, D_2 and A_n
$SVD(A_n)$ to obtain U and V
for each $p_i \in \mathcal{P}$ **do**
  if $p_i \in site_x$ **then**
    increment c_x by 1
$k \leftarrow \max\{c_x, 1 \leq x \leq X\}$
build U_k and V_k and compute Z_k
run k-means on Z_k to obtain k clusters $C = \{C_1, ..., C_k\}$
return C

number of pages from the same website, *i.e.*,

$$k = \max\{c_x, 1 \leq x \leq X\}.$$

Algorithm 1 shows the pseudo-code of the co-clustering process on a smoothed graph. For each resulted cluster C_z in the set C of k clusters, if all the pages in C_z are from different websites, they form a ClusteredMatching of pages P_z that refer to the same entity. The queries in C_z naturally form an explanation of the entity that are referred to by this ClusteredMatching. There are also AmbiguousCluster that contain multiple pages from the same website. Additional refinement is necessary for them to build matchings that only consist of pages from different websites.

3.4 Refining the matchings: exploring page similarities

Given an AmbiguousCluster, we selectively choose one page for each website that contributes multiple pages to the cluster to form a meaningful matching, *i.e.*, AdvancedMatching. We rely on the similarity among pages to select the pages to form AdvancedMatching. Without knowing the content of pages, an effective way to quantify their similarity is to exploit their semantic relations exposed by the edges in the click graph. Therefore, we use the steady-state probability matrix S that is computed to smooth the click graph (Section 3.2). This does not require additional computation. The sub-matrix with values S_{ij}, $n+1 \leq i \leq n+m$ and $n+1 \leq j \leq n+m$, corresponds to the probability of random walk starting at page p_i to arrive at page p_j and thus the similarity between p_i and p_j.

Algorithm 2 depicts the pseudo-code for refining an AmbiguousCluster. If AmbiguousCluster C_a contains pages from Y websites and each $site_y$ contributes c_y pages to C_a, the pages from different $site_y$ with $c_y = 1$ directly form part of a refined cluster $C_{r,l}$. For $site_y$ with $c_y > 1$, the page p_i having the maximum similarity with the pages that are already in $C_{r,l}$ is added to $C_{r,l}$, if this similarity is larger than a pre-defined threshold β ($0 \leq \beta \leq 1$). The similarity $Similarity(p_i, C_{r,l})$ between page p_i and the refined cluster $C_{r,l}$ is computed as

$$Similarity(p_i, C_{r,l}) = \frac{1}{|C_{r,l}|} \sum_{p_j \in C_{r,l}} \frac{S_{ij} + S_{ji}}{2},$$

where $|C_{r,l}|$ is the size of the refined cluster $C_{r,l}$.

If there are multiple $site_y$ in C_a having $c_y > 1$, besides this refined cluster, other pages from these websites can also form refined clusters. For instance, an AmbiguousCluster with pages of "Disneyland" may contain several matchings

Algorithm 2 Refining an ambiguous cluster

Input: ambiguous cluster C_a
Output: a set C_r of refined clusters

group pages in C_a by website and rank sites by ascending c_y
for each $site_y$ **do**
  if $c_y \geq 1$ **then**
    get p_h from $site_y$ and $C_{r,l} \leftarrow \{p_h\}$
    $site_y \leftarrow site_y \setminus \{p_h\}$
  for each $site_z$ ranked behind $site_y$ **do**
    select $p_i \in site_z$ with maximum $Similarity(p_i, C_{r,1})$
    if $Similarity(p_i, C_{r,1}) > \beta$ **then**
      $C_{r,l} \leftarrow C_{r,l} \cup \{p_i\}$
      $site_z \leftarrow site_z \setminus \{p_i\}$
      for each q_j in C_a **do**
        if $e_{jh} \in \mathcal{G}_s$ and $e_{ji} \in \mathcal{G}_s$ **then**
          $C_{r,l} \leftarrow C_{r,l} \cup \{q_j\}$
          $C_a \leftarrow C_a \setminus \{q_j\}$
  $C_r \leftarrow C_r \cup \{C_{r,l}\}$
return C_r

of pages that refer to Disneyland in different locations of the world. Hence, we iterate over all the websites having multiple pages in C_a, in ascending order of the number of pages they have (*i.e.*, c_y), to obtain more refined clusters. If page p_i from $site_y$ has the maximum $Similarity(p_i, C_{r,l})$ with all the pages in the refined cluster $C_{r,l}$ and $Similarity(p_i, C_{r,l})$ is larger than β, p_i is added to $C_{r,l}$.

The pages in each refined cluster containing multiple pages form an AdvancedMatching. We finally map queries to these clusters by selecting the queries in C_a that associate at least one pair of pages in the refined clusters. Again, the queries in a refined cluster provide an explanation of the entity referred to by the corresponding AdvancedMatching.

3.5 Complexity analysis

As we have explained, the entire entity matching process consists of identifying the connected components in the original click graph $\mathcal{G} = \langle \mathcal{Q}, \mathcal{P}, \mathcal{E} \rangle$ and smoothing, co-clustering, refining each connected click graph $\mathcal{G}_c$ with n queries and m pages. The DFS algorithm identifies connected components in linear time, *i.e.*, $O(|\mathcal{Q}| + |\mathcal{P}| + |\mathcal{E}|)$. This can be reduced to logarithmic time with parallel algorithms [14].

The time for smoothing a graph is dominated by the inversion of transition matrix, which is in $O((n+m)^3)$. If $\mathcal{G}$ consists of N connected components, the time for smoothing them does not surpass $O(N(n+m)^3)$. Yet, as we will see in Section 4.6, if we double the size of the original click graph $\mathcal{G}$, the number of connected components it contains (*i.e.*, N) also doubles, while the size of the largest connected component (*i.e.*, $n+m$) is almost the same and is very small. This suggests that the computational cost of smoothing increases linearly with the size of $\mathcal{G}$ as N increases linearly with it.

Regarding to co-clustering, the choice of the desired number of clusters k can be achieved in $O(m)$, SVD can be achieved in $O(min\{n^2m, nm^2\})$ and k-means can be achieved in $O((n+m)kI)$ with the bounded number of iterations I. Again, for each smoothed graph, the values of n and m are very small compared to the size of $\mathcal{G}$. Thus, the overcall cost of co-clustering increases linearly with the size of $\mathcal{G}$.

The time for refining an ambiguous cluster depends on the number of websites Y in the cluster and the number of pages c_y from each website. In the worst case where all the Y sites have the same number of pages, *i.e.*, $c_y = m/Y$, this process requires $\frac{c_y \times (c_y+1)}{2} \times (Y-1)$ computations of *Simi-*

Website	Number of pages	Number of edges
tripadvisor.com	64,459	123,320
travel.yahoo.com	50,897	91,134
travelpod.com	3,598	4391
virtualtourist.com	153	234
wikipedia.org	166,350	258,801
yelp.com	74,625	121,863

Table 1: Statistics of the click graph.

$larity(p_i, C_{r,l})$, *i.e.*, $O(m^2/Y)$. Hence, the overall cost of refining the ambiguous clusters is bounded by $O(Nm^2/Y)$.

As we will see in Section 4.2, only a small subset of connected components requires smoothing, co-clustering and refining to obtain ClusteredMatching and AdvancedMatching. More importantly, despite the cubic complexity to process each connected component, the overall cost only increases linearly with the size of the entire click graph, since the maximum size of the connect components remains *small* and *stable* in click graphs of different sizes. This is key to the scalability of our entity matching approach.

4. EXPERIMENTAL RESULTS

4.1 Experimental settings

Dataset: In the experiments, we use a sample of recent 30-day query click logs of Yahoo! Web search engine. We build the click graph with pages from 6 websites, including "tripadvisor.com", "travel.yahoo.com", "travelpod.com", "virtualtourist.com", "wikipedia.org" and "yelp.com". The first four sites are famous travel websites that assist users in gathering travel information and posting reviews, while "yelp.com" provides a complete list of businesses throughout US and Canada with user-generated reviews. This graph contains $23,112,812$ queries, $3,527,069$ pages and $24,935,578$ edges. Our approach takes advantage of shared clicks to perform clustering. If a page is not co-clicked with pages from other websites for any query, it cannot form any matching. Therefore, we filter out the queries that only lead to clicks on pages from the same websites, as well as the associated edges. We obtain a click graph with $191,645$ queries, $360,082$ pages and $599,743$ edges. The numbers of pages and edges that related to each website are shown in Table 1.

Methodology: We first rely on DFS to identify all the connected components in the click graph. We identify SimpleMatching from the connected components that only consist of pages from different websites. We then apply each step presented in Section 3 to identify matchings in the connected components in which there are at least two pages from the same website. We compare the performance of our approach against an alternative that does not smooth the graph to highlight the benefits of smoothing. We also train a simple SVM model using the matchings generated with our approach to show their potential as seed data for supervised entity matching approaches.

4.2 Structural results

We first present the statistics of the original, smoothed, clustered graphs, and the different types of matchings obtained. By performing DFS on the click graph, we obtain $85,282$ connected components. Since $68,600$ of them only consist of pages from different websites, the pages in each connected component form a SimpleMatching. We obtain $68,600$ SimpleMatching.

α	Number of edges	Fraction of added edges
0.05	497,888	1.4166
0.1	320,767	0.5569
0.2	220,873	0.0720
0.3	207,118	0.0053

(a) Edges in smoothed graph ($c = 0.15$).

α	Number of edges	Fraction of added edges
0.001	252,868	0.2273
0.0015	215,188	0.0444
0.002	207,361	0.0065
0.0025	206,162	0.0006

(b) Edges in smoothed graph ($c = 0.85$).

α	# ClusteredMatching	# AmbiguousCluster
0.05	11,199	9507
0.1	12,619	8679
0.2	12,618	8057
0.3	12,458	7939

(c) Output of co-clustering ($c = 0.15$).

α	# ClusteredMatching	# AmbiguousCluster
0.001	12,572	8347
0.0015	12,517	7992
0.002	12,485	7964
0.0025	12,462	7956

(d) Output of co-clustering ($c = 0.85$).

Table 2: Structural results

For the remaining $16,682$ connected components that consist of at least two pages from the same website, we first smooth their corresponding click graphs and then identify the matchings on the smoothed graphs.

We set c to 0.15 and 0.85. 0.15 is typically used in Web graphs to compute PageRank [17] and we use 0.85 as an opposite case. For each c value, we set the smoothing threshold α to four different values. Table 2(a) and Table 2(b) depict the number of edges in the $16,682$ smoothed graphs for $c = 0.15$ and $c = 0.85$. There are $206,031$ edges in the original graph. Clearly, higher value of the smoothing threshold α leads to fewer added edges in the smoothed graphs.

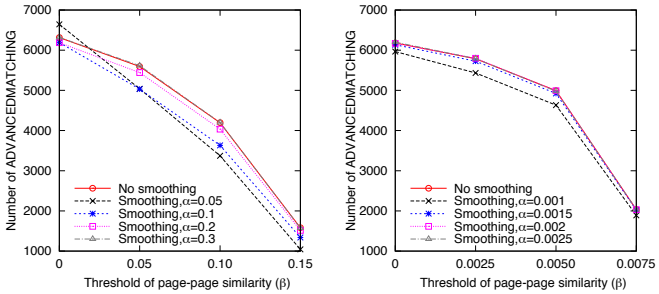
We co-cluster the graphs to obtain ClusteredMatching. Table 2(c) and Table 2(d) summarize the number of ClusteredMatching and AmbiguousCluster obtained on the smoothed graphs through co-clustering. If the graphs are not smoothed, we obtain $12,444$ ClusteredMatching and 7947 AmbiguousCluster. Smoothing the graphs slightly increases the number of ClusteredMatching and adding more edges results in more matchings. Smoothing also increases the number of AmbiguousCluster.

We refine obtained AmbiguousCluster as described in Section 3.4 to exact AdvancedMatching. Figure 5 compares the number of AdvancedMatching identified in the original graph and different smoothed graphs with respect to different values of the page-page similarity threshold β. We observe that larger β leads to fewer AdvancedMatching.

We have measured the number of matchings (*i.e.*, *Coverage*) that can be obtained using our approach, *i.e.*, $68,600$ SimpleMatching, more than $12,000$ ClusteredMatching and up to 6600 AdvancedMatching. We further evaluate the quality of these matchings with respect to different smoothing and refining parameters.

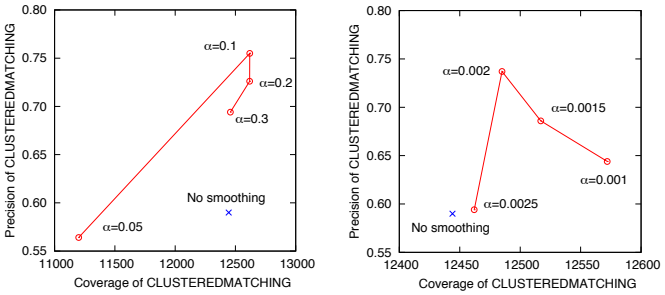
4.3 Matching quality: precision and coverage

We evaluate the accuracy of our approach using *Precision*, which is the number of correct matchings in which the pages refer to the same entities divided by the



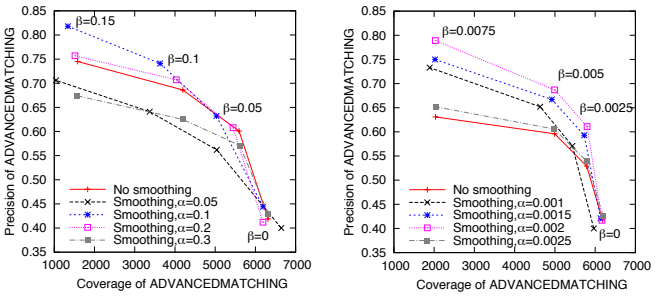
(a) $c = 0.15$. (b) $c = 0.85$.

Figure 5: Number of AdvancedMatching.



(a) $c = 0.15$. (b) $c = 0.85$.

Figure 6: Performance of ClusteredMatching.



(a) $c = 0.15$. (b) $c = 0.85$.

Figure 7: Performance of AdvancedMatching.

total number of obtained matchings. Higher value of precision indicates higher accuracy of the matching approach.

To assess the quality of each obtained matching, we conduct a user study to examine if all the pages it contains refer to the same entity. Different from the top-k results of a keyword search, whether two pages refer to the same entity is independent of the assessor's own preference, the user study rarely introduces disagreement. Thus a matching is either correct or wrong as an entity can be uniquely identified by a combination of features. For example, for a matching of a hotel, we can compare the name, the address, the homepage of the hotel and the external booking websites (*e.g.*, "booking.com") from each website to verify if they refer to the same hotel. If the available features are not enough to make a judgment, the matching is considered as wrong.

Table 3 gives some examples of correct and wrong matchings. We see that the queries of correct matchings usually indicate the name of the corresponding entity, such as "hyatt regency phoenix" for the hotel entity, and "latin quarter montreal" for the tourist attraction entity. Wrong matchings are often due to the ambiguity in the entity names. In the example of wrong matching, the Wikipedia page presents the chain restaurant "the hat" in general while the other two pages refer to its branches in different locations.

We manually inspect the quality of SimpleMatching, ClusteredMatching and AdvancedMatching, obtained either on original or smoothed graphs. In each experiment, we randomly sample 1% to 10% from each kind to ensure that at least 100 matchings are examined for that kind.

The precision of SimpleMatching is 0.843, indicating pages that are co-clicked by the same set of queries have high probability to refer to the same entities. This also confirms our motivation of mining page matchings in click graphs.

The precision of ClusteredMatching is 0.590 if the graphs are not smoothed. Figure 6 shows the precision and coverage of ClusteredMatching on smoothed graphs with different restart probability c and smoothing threshold α. We observe that with appropriate values of α, *e.g.*, $\alpha \geq 0.1$ for $c = 0.15$ and $\alpha \geq 0.001$ for $c = 0.85$, the precision of ClusteredMatching on smoothed graph is consistently better than on the original graph. Smoothing too much, *i.e.*, adding too many edges with small α, the precision may become worse than if the graph is not smoothed, such as $\alpha = 0.05$ for $c = 0.15$. In contrast, if the graph is not smoothed enough with sufficient number of additional edges, the benefits of smoothing may not be significant as in the graph obtained with $c = 0.85$ and $\alpha = 0.0025$.

Figure 7 compares the precision and coverage of AdvancedMatching for different parameters. We observe that refining AmbiguousCluster results in high quality AdvancedMatching and the precision can reach at 0.818. Yet, there is a trade-off between the number of AdvancedMatching and the corresponding precision. A larger value of β ensures high precision but fewer matchings are obtained. With $\alpha = 0.1$ for $c = 0.15$ and $\alpha = 0.002$ for $c = 0.85$, which provide the best precision of ClusteredMatching, the corresponding precisions of AdvancedMatching are consistently better than the precisions obtained on original graph.

4.4 Matching as seed data

To demonstrate the effectiveness of the matchings as a seed set, we train a linear SVM model with bag-of-words features. For the baseline, we train the classifier with an editorially labeled set of 100 pages. From the matchings generated in by our approach (with the setting $c = 0.15$, $\alpha = 0.1$ and $\beta = 0.1$), we choose seed sets of varying sizes: 100, 1000, 10000, and 50000 pages. The classifiers are tested on another set of 96 editorially labeled non-overlapping pages. Both editorially labeled sets are sampled from the correct matchings generated by our approach.

Table 4 summarizes the *Precision* (fraction of correct matchings in the classifier output). We observe that when the size of seed set is small, the performance of the classifier trained with automatically generated seed set performs the same as that trained with editorially labeled seed set. Moreover, by increasing the size of the seed set, even if the automatically generated seed matchings are not completely accurate, the performance of the classifier improves. Although we do not perform this experiment for large editorially labeled seed set given the associated difficulties (the main motivation of our approach), and we rely on simple machine learning model, this result reveals that our approach is promising to generate high quality seed data in a scalable way for supervised entity matching approaches.

Status	Pages	Queries
Correct	travel.yahoo.com/p-hotel-357759-hyatt_regency_phoenix-i;_ylc=X3oDMTBtMGZ1MWNiBF9TAzI3NjY2NzkEX3MDOTY www.tripadvisor.com/Hotel_Review-g31310-d73855-Reviews-Hyatt_Regency_Phoenix-Phoenix_Arizona.html	hyatt regency phoenix
Correct	en.wikipedia.org/wiki/Quartier_Latin,_Montreal travel.yahoo.com/p-travelguide-2803822-latin_quarter_montreal-i www.travelpod.com/ad/Latin_Quarter_Quartier_Latin-Montreal www.tripadvisor.com/Attraction_Review-g155032-d240017-Reviews-Latin_Quarter_Quartier_Latin-Montreal_	latin quarter montreal canada latin quarter montreal montreal latin quarter
Wrong	en.wikipedia.org/wiki/The_Hat www.yelp.com/biz/the-hat-pasadena www.tripadvisor.com/Restaurant_Review-g33206-d468695-Reviews-The_Hat-Upland_California.html	the hat the hat restaurant the hat restaurant pasadena

Table 3: Example of matchings.

	Seed (Editorial)	Seed (Our Approch)			
# pages	100	100	1000	10,000	50,000
Precision	0.09	0.09	0.4	0.51	0.62

Table 4: Matching precision wrt. different seed data

Matching type	Processing	Avg.	StdDev.
SimpleMatching	DFS	389	44
	Matching	180	26
ClusteredMatching	Smoothing	144	2
	Co-clustering	164	10
	Matching	16	1
AdvancedMatching	Matching	8	2

Table 5: Running time of our approach (in seconds).

4.5 Matching efficiency: running time

We measure the efficiency of our approach using its running time for matching pages. All the experiments are conducted on a machine with Intel 2GHz CPU and 4G RAM, except the raw data is cleaned on a Hadoop cluster. We run each experiment for 3 times and report the average running time in Table 5. Basically, our approach takes 15 minutes to identify the potential matchings in the click graph. It is worth noticing that the value of smoothing parameter α has almost no impact on running time as the similarity between every pair of query and page needs to be examined in the smoothing process regardless of the value of α, and although the time for identifying AdvancedMatching increases when the page-page similarity threshold β decreases, the actual running time does not surpass 15 seconds in our settings. Compared to matching the pages in the original graph without any smoothing, our approach only takes 154 seconds (16%) longer as no smoothing is performed and less information is examined in the following processes.

4.6 Scalability

To convey the scalability of our approach, we conduct the same experiment with a smaller graph of 67,898 queries, 167,116 pages and 226,675 edges, derived from 10-day query logs of the same period. Interestingly, among the 41,443 connected components requiring co-clustering to identify matching of pages, the largest connected component only contains 203 queries and pages, *i.e.*, $\max\{n+m\} = 203$. This value is almost the same as the maximum value in the larger graph derived from 30-day query logs, for which $\max\{n+m\} = 212$. This reveals that performing our approach in larger click graph does not increase the computational cost cubicly even if the most expensive smoothing is in $O((n+m)^3)$. Instead, the cost only increases *linearly* as there are twice more connected components to process in the larger graph with twice more vertices and edges.

Matching type	Our approach		Original graph	
	Cov.	Pre.	Cov.	Pre.
SimpleMatching	68,600	0.843	68,600	0.843
ClusteredMatching	12,619	0.755	12444	0.590
AdvancedMatching	3632	0.741	4196	0.686

Table 6: Best-case performance (coverage and precision) of the proposed matching technique (for $c = 0.15$, $\alpha = 0.1$, $\beta = 0.1$) and comparison with baseline.

4.7 Summary

Our evaluation demonstrates that user clicks on Web search is a valuable source of information that can be leveraged for effective entity matching. With appropriate parameters, *e.g.*, $c = 0.15$, $\alpha = 0.1$ and $\beta = 0.1$, we obtain up to $84,851$ matchings on a click graph of $360,082$ pages. Table 6 summarizes the three kinds of matchings with respect to their coverage (Cov.) and precision (Pre.). These matchings consist of 179,563 pages, accounting for 49.9% pages in the click graph. This conveys the effectiveness of our entity matching approach. Importantly, using the proposed methods (*i.e.*, co-clustering and refining) increases the coverage of SimpleMatching by 23.7% at the cost of modest decrease (11%) in precision. Even if about 0.4% fewer matchings are obtained on the smoothed graph, the precisions of ClusteredMatching and AdvancedMatching are 28% and 8% better than those on the original graph. The overall precision of matching on smoothed graph can reach 0.826, outperforming the precision on the original graph by 4%. This confirms the accuracy of our approach. In fact, our approach is also efficient: 15 minutes are enough to obtain the potential matchings implied in the click graph and the time only increases linearly as the original graph grows. And, finally, we demonstrate the effectiveness of the seed set generated as training data for entity matching.

5. RELATED WORK

The work presented in this paper straddles two areas of research: entity matching and user click analysis. Entity matching, also known as record linkage [2], object identification [21], de-duplication [3], *etc.*, has been extensively studied in the literature [12, 16]. Numerous approaches, both supervised [3, 21] and unsupervised [13, 20], have been proposed for entity matching. The effectiveness of supervised approaches depends on size and quality of training data [16]. Providing training data typically involves editorial efforts to create and choose entity pairs to match, label matched entities, *etc.* Given the large amount of entities and their varieties in nature, it is usually difficult and time-consuming to determine a good set of training data. Unsupervised approaches alleviate the necessity of training data, while the

similarity functions they use rely on attributes to determine whether a pair of entities is a matching. Yet, information sources are usually maintained by different organizations. It is not always obvious to access the underlying databases of entities (*e.g.*, XML or relational records), while extracting attributes from textual description [11] is not practical given the large amount of entities to match.

Therefore, we propose in this work to a novel approach that leverages user clicks on Web search to automatically generate large-scale training data for entity matching. User interactions with search engines has been used for several tasks such as query expansion and clustering [1, 5, 24], Web page clustering [6, 9], entity ranking [4, 15], *etc.* Specifically, it is revealed in [1] that search click graph captures semantic relations between queries while bicliques and session information in the click graph is used in [5] and [24] to cluster and suggest similar queries. In [6], search logs are used to find paths in the DOM trees that mark out important content of pages to perform more accurate clustering of pages. In [9], pages relevant to a query are ranked based on user clicks through forward and backward random walks on the click graph. In [4], random walk is performed on both click graph and session graph to rank pages relevant to an entity. Differently, we use random walk with restarts to both smooth the graph and rank the pages.

In the area of entity matching, search logs are used to train a machine learned ranking model to predict the relevance of query-page pairs in [15]. An entity-aware click graph derived from search logs is used to match websites that fulfill similar user needs in [18]. It is worth noticing that these approaches using user clicks to cluster Web pages or rank entities are not very strict in the sense that related, but not necessarily the same, entities belong to a cluster. For example, while a cluster consisting of queries (or pages corresponding to) "Eiffel Tower" and "Notre Dame" will be considered acceptable in all the above approaches, it is incorrect for entity matching. Hence entity matching is a very demanding clustering task. *To the best of our knowledge, we are the first to leverage user click behaviors in search for entity matching.*

6. CONCLUSION

We propose in this paper a practical approach that exploits, for the first time, user clicks on Web search to generate seed data that can be used to train models for large-scale entity matching. Experiments on real datasets show that shared clicks among queries and pages are good indicators for disambiguating and matching the pages referring to the same entities. Reducing the sparseness of Web search data through smoothing further improves the effectiveness and accuracy of matching. Use of the matched data as training data in supervised approach to entity matching appears promising: the performance matches that of editorial data at small size and improves with size.

There is still room to improve the performance. For instance, parallel algorithms [14] might be used to identify connected components in click graphs. Once having the connected components, the process of identifying matchings in each component can be easily parallelized. Content-based features, such as keywords in queries and URLs, might be considered in addition to content-independent clicks as some incorrect matchings are due to ambiguity implied in the entities and could be clarified with more context information.

Acknowledgement

This work was supported by the LEADS project (ICT-318809), funded by the European Community, and the Torres Quevedo Program from the Spanish Ministry of Science and Innovation, co-funded by the European Social Fund.

7. REFERENCES

[1] R. Baeza-Yates and A. Tiberi. Extracting semantic relations fromquery logs. In *KDD*, 2002.

[2] M. Bilenko, B. Kamath, and R. J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *ICDM*, 2006.

[3] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*, 2003.

[4] B. Billerbeck, G. Demartini, C. S. Firan, T. Iofciu, and R. Krestel. Ranking entities using web search query logs. In *ECDL*, 2010.

[5] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD*, 2008.

[6] D. Chakrabarti and R. R. Mehta. The paths more taken: matching DOM trees to search logs for accurate webpage clustering. In *WWW*, 2010.

[7] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 1983.

[8] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill, 2001.

[9] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, 2007.

[10] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *SIGKDD*, 2001.

[11] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the world-wide web. In *AGENTS*, 1997.

[12] C. F. Dorneles, R. Gonçalves, and R. dos Santos Mello. Approximate data instance matching: a survey. *Knowledge and Information Systems*, 2011.

[13] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice and open challenges. *PVLDB*, 2012.

[14] J. Greiner. A comparison of parallel algorithms for connected components. In *SPAA*, 1994.

[15] C. Kang, S. Vadrevu, R. Zhang, R. v. Zwol, L. G. Pueyo, N. Torzec, J. He, and Y. Chang. Ranking related entities for web search queries. In *WWW*, 2011.

[16] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69(2):197–210, 2010.

[17] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *SIGMOD*, 2010.

[18] P. N. Mendes, P. Mika, H. Zaragoza, and R. Blanco. Measuring website similarity using an entity-aware click graph. In *CIKM*, 2012.

[19] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *SIGKDD*, 2004.

[20] V. Rastogi, N. Dalvi, and M. Garofalakis. Large-scale collective entity matching. *PVLDB*, 2011.

[21] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Inf. Syst.*, 26(8):607–633, Dec. 2001.

[22] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, 2006.

[23] C. Wang, F. Jing, L. Zhang, and H.-J. Zhang. Image annotation refinement using random walk with restarts. In *ACM MM*, 2006.

[24] J. Yi and F. Maghoul. Query clustering using click-through graph. In *WWW*, 2009.