Web People Search via Connection Analysis

Dmitri V. Kalashnikov   Zhaoqi Chen   Sharad Mehrotra   Rabia Nuray-Turan

***Abstract*— Nowadays, searches for webpages of a person with a given name constitute a notable fraction of queries to web search engines. Such a query would normally return webpages related to several namesakes, who happened to have the queried name, leaving the burden of disambiguating and collecting pages relevant to a particular person (from among the namesakes) on the user. In this article we develop a Web People Search approach that clusters webpages based on their association to different people. Our method exploits a variety of semantic information extracted from Web pages, such as named entities and hyperlinks, to disambiguate among namesakes referred to on the Web pages. We demonstrate the effectiveness of our approach by testing the efficacy of the disambiguation algorithms and its impact on person search.**

***Index Terms*— Web People Search, Entity Resolution, Graph-based Disambiguation, Social Network Analysis, Clustering**

I. Introduction

Searching for entities is a common activity in Internet search today. Searching for webpages related to a person accounts for over 5% of the current Web searches [24]. Currently, it is done using keywords. A search engine, such as Google or Yahoo, returns a set of Web pages, in ranked order, where each Web page is deemed relevant to the search keyword entered (the person name in this case).[1] A search for a person, such as say "Andrew McCallum" will return pages relevant to *any* person with the name Andrew McCallum.

A next generation search engine can provide significantly more powerful models for person search. Assume (for now) that for each such Web page the search-engine could determine which real entity (i.e., *which* Andrew McCallum) the page refers to. This information can be used to provide a capability of *clustered* person search where instead of a list of Web pages of (possibly) multiple persons with the same name, the results are clustered by associating each cluster to a real person. The clusters can be returned in a ranked order determined by aggregating the rank of the Web pages that constitute the cluster. With each cluster we also provide a summary description that is representative of the real person associated with that cluster (for instance in this example the summary description may be a list of words such as "computer science, machine learning, professor"). The user can hone in on the cluster of interest to her and get all pages in that cluster, i.e., only the pages associated with *that* Andrew McCallum.

Such cluster-based people search could potentially be very useful. Imagine searching for the Web page of "George Bush" who used to live in your neighborhood in Champaign, Illinois using Google today. This is virtually impossible (or at least very tiring) since the first 20-30 pages of a Google search of "George Bush" returns pages only about the President. In the clustered approach, ideally, all of the President's pages will be folded into a single cluster giving his namesakes an opportunity to be displayed in the first page of search results. One might argue that the use of context could improve the results of the standard search engines today and thus there is no need for clustering approaches. However, this is not the case if you have very little knowledge about the person you are searching for. For example, assume that we are searching for "Tom Mitchell, the psychology professor" with his name and keywords "psychology" and "professor". The search engine, e.g., Google, returns more than 2 different people (to be exact 13 different persons in the top 100 pages). Hence, the task of clustering the pages related to different people is still valid even for the queries that include context.

While the example above shows the clustered approach in a positive light, in reality, it is not that obvious that it indeed is a better option compared to searching for people using keyword-based search supported by current search engines. Intuitively, if clusters identified by the search engine corresponded to a single person, then the clustered-based approach would be a good choice. On the other hand, if clusters contained errors (multiple people merged into the same cluster, or alternatively, pages of the same person spread over multiple clusters) the advantages of a cluster-based approach are not obvious. For instance, if the Web pages were randomly assigned to clusters, the cluster-based approach could be worse compared to the state-of-the-art. The key issue is the quality of clustering algorithms in disambiguating different web pages of the namesake.

In this paper, we make the following contributions. First, we develop a novel algorithm for disambiguating among people that have the same name. Our algorithm is based on extracting 'significant' entities such as the names of other persons, organizations, and locations on each Web page, forming relationships between the person associated with the Web page and the entities extracted, and then analyzing the relationships along with features such as TF/IDF, as well as other useful content including hyperlink information to disambiguate the pages. We then design a cluster-based people search approach based on the disambiguation algorithm. We conduct a detailed experimental study to (1) determine the effectiveness of the disambiguation algorithm and, (2) compare traditional people search supported by current search engines with the clustered entity search built on top of the disambiguation algorithm. Our results show that clustered person search offers significant advantages. The main contributions of this article are:

- A new approach for Web People Search that shows high-quality clustering results (Sections II–IV).
- A thorough empirical evaluation of the proposed solution (Sections VII).
- A new study of the impact on search of the proposed approach (Sections VII-C).

The authors are with the University of California, Irvine
Manuscript received May 31, 2007; revised Mar 24, 2008.
This research was supported by NSF Awards 0331707 and 0331690.
A preliminary version of this article has appeared as a short paper [29].

[1]There are other people information search services as well (such as http://people.yahoo.com and http://find.intelius.com) that provide "background information" about people, such as current and previous addresses and a host of other information when available; our interest and focus is on Web pages relevant to a person on the public Internet.

In the subsequent sections we describe the proposed approach in more detail. We start by presenting an overview of the overall approach in the next section.

II. Overview of the Approach

In this section we provide an overview of all the necessary algorithms and components for implementing the Web People Search system. We take the middleware-based approach to develop our algorithms. In the proposed approach the processing of a user query consists of the following steps:

1) *User Input.* A user submits a query to the middleware via a specialized web-based interface.
2) *Webpage Retrieval.* The middleware queries a search engine with this query via the search engine API and retrieves a fixed number (top K) of relevant Web pages.
3) *Preprocessing.* The retrieved Web pages are preprocessed:
    a) *TF/IDF.* Preprocessing steps for computing TF/IDF are carried out. They include: stemming, stop word removal, noun phrase identification, inverted index computations, etc.
    b) *Extraction.* Named entities, and web related information is extracted from the Web pages.
4) *Graph Creation.* The entity-relationship graph is generated based on data extracted on the preprocessing step (Section III).
5) *Clustering.* The clustering algorithm takes the graph, TF/IDF values, and model parameters and disambiguates the set of (K) Web pages (Section IV). The result is a set of clusters of these pages with the aim being to cluster Web pages based on association to real person.
6) *Cluster Processing.* Each resulting cluster is then processed as follows (Section V):
    a) *Sketches.* A set of keywords that represent the web pages within a cluster is computed for each cluster. The goal is that the user should be able to find the person of interest by looking at the sketch.
    b) *Cluster Ranking.* All clusters are ranked by a chosen criterion to be presented in a certain order to the user.
    c) *Web page Ranking.* Once the user hones in on a particular cluster, the Web pages in this cluster are presented in a certain order, computed on this step.
7) *Visualization of Results.* The results are presented to the user in the form of clusters (and their sketches) corresponding to namesakes and which can be explored further.

The following sections will elaborate all of these steps in detail.

III. Generating a Graph Representation

The core of our approach is based on analyzing entities, relationships, and features (instantiated attributes of entities) present in the dataset. For example, the word distribution inside a webpage/document is frequently utilized as a topic feature of that webpage/document in the literature. A webpage topic can be captured using the TF/IDF methodology [5], or by other techniques [8], [39]. Our goal is to exploit entities and relationships for disambiguation. However, unlike the problem settings of many disambiguation methods which work off a normalized database, e.g. [2], [12], [15], [16], [20], in our problem setting we do not have the entities and relationships associated with each Web page already available for use. Rather such entities and

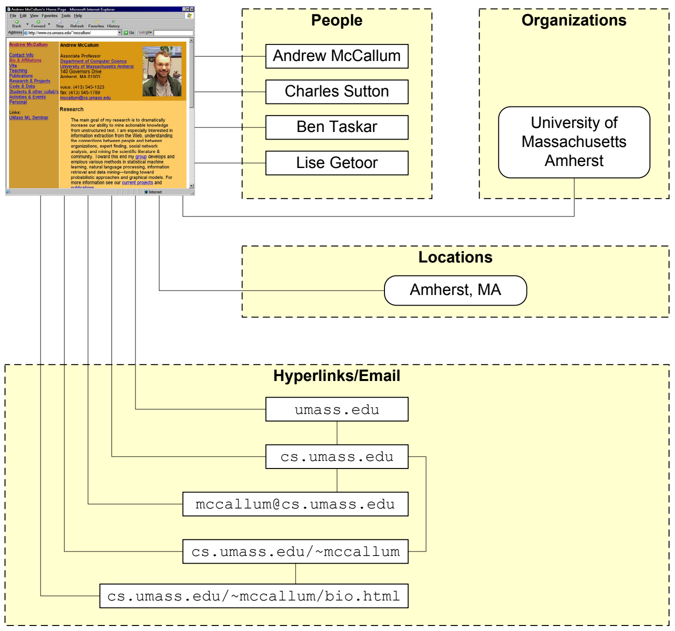


Fig. 1. Extraction of Named Entities and Web-related Info from a Webpage.

relationships need to be *extracted* off the Web pages which we do using information extraction (IE) software. In addition to Named Entities (NEs), we also extract hyperlinks and email addresses from the Web pages, see Figure 1.

The abstract representation we wish to construct is a graph where the nodes correspond to the different Web pages and entities and the edges correspond to the relationships between the Web pages and entities or among entities. The graph creation algorithm is illustrated in Figure 2. When an NE is extracted, a node is created for that NE to represent all NEs with the same name. For example, a person 'John Smith' might be extracted from two different Web pages. A single node will be created for 'John Smith', regardless whether the two pages refer to the same person or to two different people. The node represents the group of persons that share the same name. The same holds for locations and organizations. A node is also created per each of the (top K) Web pages. A relationship edge is created between a node representing a Web page and a node corresponding to each NE extracted from that Web page. The relationship edges are typed. A relationship edge between a Web page (node) and a person (node) will have a type distinct from a relationship edge between a Web page (node) and an organization (node) or a location (node). Any hyperlinks and email addresses extracted from the Web page are handled in an analogous fashion, that is, with nodes being created to correspond to these hyperlinks and email addresses and edges corresponding to the relationship with the page they are extracted from, see Figure 3.

A hyperlink has the form `[www.]`$\mathtt{d}_m$`.`$\cdots$`.`$\mathtt{d}_2$`.`$\mathtt{d}_1$`/`$\mathtt{p}_1$`/`$\mathtt{p}_2$`/`$\cdots$`/`$\mathtt{p}_n$. For example, for the URL `www.cs.umass.edu/˜mccallum/`, we have $\mathtt{d}_3$ =`cs`, $\mathtt{d}_2$ =`umass`, $\mathtt{d}_1$ =`edu`, and $\mathtt{p}_1$ =`˜mccallum`. We create a node for that hyperlink and connect it to the webpage via the edges that correspond to the 'webpage-contains-hyperlink' relationship. Then we create a node for a shorter hyperlink, without $\mathtt{p}_n$: $\mathtt{d}_m$`.`$\cdots$`.`$\mathtt{d}_2$`.`$\mathtt{d}_1$`/`$\mathtt{p}_1$`/`$\mathtt{p}_2$`/`$\cdots$`/`$\mathtt{p}_{n-1}$ and connect it to the node with $\mathtt{p}_n$ via edge of type 'hyperlink-partof-

```
CREATE-GRAPH($D$)
1   $V \leftarrow \emptyset$ //$V$ is the node set
2   $E \leftarrow \emptyset$ //$E$ is the edge set
3   NE $\leftarrow \emptyset$ //NE is the set of named entities
4   $L \leftarrow \emptyset$ //$L$ is the set of hyper links in the document
5   for each $d_i \in D$
6       $v_i \leftarrow$ CREATE-NODE($d_i$)
7       $V \leftarrow V \cup \{v_i\}$
8       NE $\leftarrow$ EXTRACT-ENTITIES($d_i$)
9       for each ne$_j \in$ NE
10          $v_j \leftarrow$ CREATE-NODE(ne$_j$)
11          $V \leftarrow V \cup \{v_j\}$
12          $E \leftarrow E \cup \{(v_i, v_j)\}$
13      $L \leftarrow$ EXTRACT-HYPERLINKS-EMAILS($d_i$)
14      for each link$_k \in L$
15          $v_k \leftarrow$ CREATE-NODE(link$_k$)
16          $V \leftarrow V \cup \{v_k\}$
17          $E \leftarrow E \cup \{(v_i, v_k)\}$ //create an edge
18          $(V_l, E_l) \leftarrow$ CREATE-LINK-GRAPH(link$_k, v_k$)
19          $V \leftarrow V \cup V_l$
20          $E \leftarrow E \cup E_l$
21  $G \leftarrow \{V, E\}$
22  return $G$
```

Fig. 2. Graph Creation Algorithm.

```
CREATE-LINK-GRAPH($L, v$)
1   $V \leftarrow v$ //$V$ is the node set with type
2   $E \leftarrow \emptyset$ //$E$ is the edge set
3   $v_0 \leftarrow v$
4   $i \leftarrow 1$
5   $S \leftarrow$ (SUB-DOMAIN($L$))
6   while $S$ still has sub domain
7       $v_i \leftarrow$ CREATE-NODE($S$, link)
8       $V \leftarrow V \cup \{v_i\}$
9       $E \leftarrow E \cup \{(v_{i-1}, v_i)\}$// type of relation is subdomainof
10      $S \leftarrow$ (SUB-DOMAIN($S$))
11      inc $i$
12  $S \leftarrow$ (PART-OF($L$))
13      $v_i \leftarrow$ CREATE-NODE($S$, link)
14  $V \leftarrow V \cup \{v_i\}$
15  $E \leftarrow E \cup \{(v_0, v_i)\}$// type of relation is partof
16  while $S$ still has sub directories (part of)
17      inc $i$
18      $S \leftarrow$ PART-OF($S$)
19      $v_i \leftarrow$ CREATE-NODE($S$, link)
20      $V \leftarrow V \cup \{v_i\}$
21      $E \leftarrow E \cup \{(v_{i-1}, v_i)\}$// type of relation is partof
22  return $V, E$
```

Fig. 3. Link Graph Creation Algorithm.

hyperlink', see Figure 1. We continue this process for $\mathtt{p}_{n-1}$, $\mathtt{p}_{n-2}$ and so on, until only $\mathtt{d}_m.\cdots.\mathtt{d}_2.\mathtt{d}_1$ part remains. We then create a node for a shorter hyperlink, this time without $\mathtt{d}_m$: $\mathtt{d}_{m-1}.\cdots.\mathtt{d}_2.\mathtt{d}_1$ and connect it to that with d_m via edge of type 'subdomainof'. We continue this process until only $\mathtt{d}_2.\mathtt{d}_1$ remains. Figure 1 illustrate this process for hyperlink `cs.umass.edu/~mccallum/bio.html`, which is linked via a chain of edges to `umass.edu`. Each of the hyperlinks that correspond to the nodes in that chain are also mentioned separately from the webpage itself, thus those nodes are shown to be connected to the webpage.

At the end of this process, we have a complete graph representation of the information that a clustering or disambiguation algorithm can now work with. The algorithm is now abstracted from any of the extraction details and can in fact self-tune itself to optimize based on the nature of the graph.

IV. Disambiguation Algorithm

This section describes the algorithm for clustering Web pages that is employed by the proposed solution. It takes as input the entity relationship graph described in Section III. It then uses a Correlation Clustering (CC) algorithm to cluster the pages, as discussed in Section IV-A. The outcome is a set of clusters with each cluster corresponding to a person. Sections IV-B and IV-C explain how to assign edge label, used by CC, with the help of a carefully designed similarity function. Finally, Sections IV-D and IV-E discuss how to calibrate this similarity function.

A. Correlation Clustering

We group the nodes representing the Web pages that belong to the same person by employing a *Correlation Clustering (CC)* algorithm [7]. Correlation clustering has been applied in the past to group documents of the same topic and to other problems. It assumes that there is a similarity function $s(u, v)$ that for any objects (e.g., documents) u and v returns whether or not it believes that u and v are similar to each other. Such a function is typically learned on the past data. The overall clustering problem is represented as a fully-connected graph, where each object becomes a node in the graph. Each (u, v) edge is assigned "$+$" (similar) or "$-$" (different) label, according to the similarity function $s(u, v)$. The goal is to find the partition of the graph into clusters that agrees the most with the assigned labels. An interesting property of CC is that, unlike many other types of clustering, it does not take k (the number of the resulting clusters) as its input parameter, whereas k is often difficult to determine beforehand. Instead, CC determines k from the labeling itself.

The goal of CC is formulated formally as either to maximize the agreement (the number of positive edges inside the clusters plus the number of negative edges outside the clusters), or to minimize the disagreement (the number of negative edges inside the clusters plus the number of positive edges). If the '$+$' and '$-$' labels are assigned perfectly by $s(u, v)$, the right clustering can be trivially obtained by removing all the negative edges in the graph: the remaining connected components will represent the right clusters. CC is designed for the cases where $s(u, v)$ is not perfect and can mislabel some of the edges (this case is of direct interest to us), or when there is no notion of exact clusters (e.g., clusters are for document topics).

For example, if the labeling is such that edges (u, v) and (v, w) are labeled '$+$', but edge (u, w) is labeled '$-$', there will not be a clustering with perfect agreement with the labeling. In general, the more accurate $s(u, v)$ in its labeling, the higher the quality of the overall clustering.

The problem of CC is known to be NP-hard and various approximation algorithms have been proposed in the literature. We do not propose any new CC algorithm per se, we instead focus on developing and learning a new accurate $s(u, v)$ function.

B. Connection Strength

To define the similarity function $s(u, v)$ we will need to use the notion of the Connection Strength $c(u, v)$ between two objects u and v, which is defined in this section.

Various disambiguation approaches have been developed for a variety of applications. These approaches can be classified along several facets. One of these facets is the *type* of information the approach is capable of analyzing. For example, to decide if two object descriptions (or two tuples in a table) co-refer (i.e., refer to the same entity/object), the traditional approaches would analyze primarily object *features* [22], [35]. Another example are *relational* approaches, that analyze dependencies among co-reference decisions [32], [38]. Our proposed disambiguation algorithm is based on analyzing *two* types of information: object features and the Entity-Relationship graph (ER graph) for the dataset. In [17], [27], [28] it has been shown that complementing the traditional methodology of analyzing object features with analysis of the ER graph can lead to the improved quality of disambiguation.

The idea is that many real datasets are relational, and thus can be viewed as a graph of entities, represented as nodes, interconnected via relationships, represented as edges. To decide whether two object descriptions co-refer, the approach analyzes not only their features, but also the paths that exist in the ER graph between those two object descriptions.

The motivation behind analyzing features of two objects u and v is based on the assumption that similarity of features of u and v defines certain affinity/attraction between those objects $f(u,v)$, and if this attraction is sufficiently large, then the objects are likely to be the same. The intuition behind analyzing paths is similar: the assumption is that each path between two objects carries in itself certain degree of attraction. A path between u and v semantically captures (perhaps complex and indirect) interactions between them via intermediate entities. If the combined attraction of all these paths is sufficiently large, the objects are likely to be the same. An in-depth insight into the motivation for this methodology is elaborated in [27].

Formally, the attraction between two nodes u and v via paths is measured using the *connection strength* measure $c(u,v)$, which is defined as the sum of attractions contributed by each path:

$$c(u,v) = \sum_{p \in P_{uv}} w_p. \tag{1}$$

Here P_{uv} denotes the set of all L-short simple paths between u and v, and w_p denotes the weight contributed by path p. A path is *L-short* if its length does not exceed L and is *simple*, if it does not contain duplicate nodes. The weight path p contributes is derived from the type of that path, and thus paths of the same type contributes the same weight. The sequence of nodes types and edge types determine the type of a path: two paths having the nodes of the same type connected via edges of the same type are considered to be of the same path type. The number of possible path types, for L-short simple path, is limited per each domain. Let w_k be the attraction associated with a path of type k. Let P_{uv} consist of c_1 paths of type 1, c_2 paths of type 2, …, c_n paths of type n. Then Eq. (1) can be equivalently written as:

$$c(u,v) = c_1 w_1 + c_2 w_2 + \cdots + c_n w_n. \tag{2}$$

In the next section we will discuss how the concept of connection strength $c(u,v)$ can help designing a better similarity function $s(u,v)$.

C. Similarity Function

Our goal is to design a powerful similarity function $s(u,v)$ that would minimize mislabeling of the data. We will design a flexible function $s(u,v)$, such that it will be able to automatically self-tune itself to the particular domain being processed. We construct our function $s(u,v)$ as a combination of the connection strength $c(u,v)$ and feature similarity $f(u,v)$:

$$s(u,v) = c(u,v) + \gamma f(u,v). \tag{3}$$

The similarity function $s(u,v)$ labels data by comparing the $s(u,v)$ value against the threshold τ, where τ is a nonnegative real number. Namely, we use the δ-band ("clear margin") approach, which labels each (u,v) edge according to the following rules:

$$\begin{cases} +1 & \text{if } s(u,v) > \tau + \delta; \\ -1 & \text{if } s(u,v) < \tau - \delta; \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

That is, if the value of $s(u,v)$ is inside the δ-band of τ, then the algorithm is uncertain whether u and v are similar and reflects that by assigning the zero ("don't know") label to the (u,v) edge. It assigns the $+1$ label to (u,v), when $s(u,v)$ exceeds the threshold by the clear positive δ margin; and it assigns the -1 label similarly. This labeling scheme allows the algorithm to avoid committing to $+$ or $-$ decision, when it does not have enough evidence for that.

TF/IDF. The proposed solution employs the standard TF/IDF scheme from the area of Information Retrieval to compute its feature-based similarity $f(u,v)$ [5]. First, the standard *preprocessing* steps are applied to all the documents, including elimination of stop words, stemming, using only noun phrases for keywords, and deriving larger terms [5].[2] Assume that the entire document corpus consists of K documents (that is, top K webpages) and contains N distinct terms $T = \{t_1, t_2, \ldots, t_N\}$. Then each document u can be characterized by vector $\mathbf{u} = \{w_{u1}, w_{u2}, \ldots, w_{uN}\}$. Here w_{ui} is the weight assigned to term t_i for document u. This weight is computed as $w_{ui} = \left(\frac{1}{2} + \frac{n_{ui}}{2\max_\ell n_{u\ell}}\right) log\frac{K}{n_i}$, where n_i is the number of documents in the corpus that contain term t_i and n_{ui} is the number of occurrences of term t_i in u. The similarity $f(u,v)$ between two documents u and v is computed using the cosine measure, $f(u,v) = \cos(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{|\mathbf{u}||\mathbf{v}|} = \frac{\sum_{i=1}^{N} w_{ui} w_{vi}}{\sqrt{\sum_{i=1}^{N} w_{ui}^2}\sqrt{\sum_{i=1}^{N} w_{vi}^2}}$.

D. Training the Similarity Function

As in traditional learning, to train the $s(u,v)$ function, we assume the past data is available. That data is fully and accurately labeled with '$+$' and '$-$' for each (u,v) edge. The learning of $s(u,v)$ is carried based on the way the algorithm assigns the labels, that is, according to rules (4) from the previous section. Namely, for each (u,v) edge, we should require that:

$$\begin{cases} s(u,v) > \tau + \delta & \text{if } (u,v) \text{ is labeled '}+\text{';} \\ s(u,v) < \tau - \delta & \text{if } (u,v) \text{ is labeled '}-\text{'.} \end{cases} \tag{5}$$

However, in practice $s(u,v)$ is unlikely to be perfect, and that would manifest itself in cases where Ineqs. (5) will be violated for some of the (u,v) edges, making the whole system (5) intractable. This problem is analogous to that found in SVMs, and it can be resolved in a similar manner: by adding slack to each inequality in (5) and then requiring that the overall slack be minimized. The

[2]The larger terms have been constructed from the neighboring keywords/terms that, when taken together, constitute a distinct concept in an ontology. We have used DMOZ ontology.

overall system becomes:

$$\begin{cases} \text{Constraints:} & \\ \delta = \delta_0 & \\ s(u,v) + \xi_{uv} > \tau + \delta & \text{for all `+'} \\ s(u,v) - \xi_{uv} < \tau - \delta & \text{for all `}-\text{'} \\ 0 \le \xi_{uv} & \text{for all } u, v \\ & \\ \text{Objective: Minimize } \sum_{uv} \xi_{uv} & \end{cases} \quad (6)$$

Here, $s(u,v)$ is computed according to Eq. (3), wherein $c(u,v)$ is computed according to Eq. (2). The task becomes to solve the linear programming problem (6) to determine the optimal values for path type weights $w_1, w_2, \ldots, w_n$ and threshold τ. Linear programming is known to have efficient solutions [25].

There are two interesting properties of Linear Program (6). First, in many disambiguation techniques, and in clustering in general, it is often a nontrivial issue to set the threshold τ, whereas in this case τ is simply learned from data. Second, finding the exact value of the optimal δ_0, to get the best results, turned out to be not a critical issue, as wide range of values neighboring the optimal δ_0 will lead to similar results. The reason is that the constraints of Linear Program (6) do not include standalone constants except for δ_0: they include w's, ξ's, τ's, δ's, which are variables, and γ (a variable) multiplied by various constants that correspond to the according TF/IDF values. This creates the effect where all these variables scale up, if δ_0 is increased, and down if it is decreased, tuning itself to δ_0 and the labeling.

E. Choosing Negative Weight

Loosely speaking, with '+'/'$-$' labeling, a correlation clustering algorithm will assign an entity u to a cluster if the number of positive edges between u and the other entities in the cluster outnumbers that of negative edges. In other words, the number of positive edges is more than half (i.e., 50%). However, we observe that when CC is applied to a particular real-world domain, an entity might need to be assigned to a cluster for a different fraction of positive edges than 50%. For instance, if for a given domain, to keep an entity in a cluster, it is sufficient to have only 25% percent of positive edges, then by using $w^+ = +1$ weight for all positive edges and $w^- = -\frac{1}{3}$ weight for all negative edges will achieve the desired effect (since $0.25 \times 1 = 0.75 \times \frac{1}{3}$). One solution for choosing a good value for the weight of negative edges w^- is to learn it on past data.

It is possible to design a better solution, based on the following observation. Assume for now that we know the number of namesakes n in the top k Web pages being processed by the algorithm. If $n = 1$ then choosing w^- as small as possible, that is $w^- = 0$, is likely to produce the best result. This is because when $w^- = 0$, there will be no 'negative weight' for CC to prevent merging and all the pair connected via positive edges will be merged. Similarly, if $n = k$, it is best to choose $w^- = -1$. This would produce maximum negative evidence for pairs not to be merged. Thus, instead of using a fixed value for w^-, it might be possible to pick a good value for w^- specifically for the top k webpages being processed, based on a function of n: $w^- = w^-(n)$.

This observation raises two issues. The first one is that n is not known to the algorithm beforehand. The second is how to choose $w^-(n)$ function.

While n is not known, we can compute its estimated value $\hat{n}$, by running the disambiguation algorithm with a fixed value of w^-. The algorithm would output certain number of clusters $\hat{n}$, which can be employed as an estimation of n. It should be noted that the overhead for this extra computation is minimal: the paths once discovered, need not be rediscovered second time. Thus the extra cost of such an estimation is equivalent to the cost of running pure CC algorithms on already labeled graph, which is less than a millisecond.[3]

The next question we need to address is how to choose $w^-(\hat{n})$ function. A straightforward solution would be to try to fit a curve to data. While this approach succeeded for smaller web datasets, in practice the following simple function has proven to work well across all the web datasets. The value of $w^-(\hat{n})$ is chosen to be zero when $\hat{n}$ is less than a certain threshold, and it is chosen to be -1 when it is above this threshold. The value for this threshold itself is learned from data.

V. Interpreting Clustering Results

Given a set of Web pages related to a particular name the disambiguation approach above generates a set of clusters. We now describe how these clusters are used to build people search. Recall that for people search our goal is to first provide the user with a set of clusters based on association to real person. The task is now to: (i) Rank the clusters. (ii) Provide a summary description with each cluster. Ranking and summarization are defined as follows:

Cluster rank: Search engines (e.g., Google) return pages in order of relevance to the query based on the algorithm they use. For each cluster we simply select the highest ranked page (i.e., the page with the numerically least order based on standard search-engine result) and use that as the order of the cluster. The cluster orders now form the basis for cluster ranks.

Cluster sketch: We coalesce all pages in the cluster into a single page. Then after removing the stop words we compute the TF/IDF of the remaining words for the coalesced page. The set of terms above a certain threshold (or top N terms) is selected and used as a summary for the cluster.

Web page rank: When the user explores a particular cluster we first display all pages in that cluster. These pages are displayed according to their original search engine order. We also take the remainder pages (i.e., pages in the top K not in the selected cluster) and compute their affinity to the selected cluster. The remainder pages are then displayed in order of the affinity to the selected cluster. Affinity is defined as:

Affinity to cluster: The affinity between a Web page p and a cluster C is defined as the sum of the similarity values between the page p and each page v in the cluster C:

$$\mathit{affinity}(p, C) = \sum_{v \in C} s(p, v).$$

The clustering is not always perfect and it may be the case that the pages for one real individual are actually spread across multiple clusters. However, since remainder pages are displayed as well, the user has the option to get to these Web pages too ultimately. Also it may be that based on the cluster summaries the

[3]We believe that the value of n can serve as an important factor for choosing values for other parameters of any web disambiguation techniques, because web data can be quite diverse. Thus, the described techniques for picking w^- based on n might be helpful in choosing other parameters in general as well.

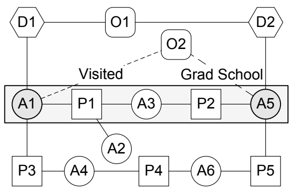


Fig. 4. Sample G^L_{uv}: plotted $G^6_{A_1 A_5}$. 'Pruned' G^L_{uv} is obtained by removing certain edge types: say by removing all 'visited' edges.

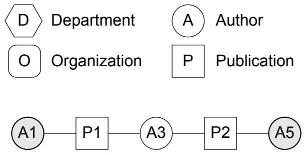


Fig. 5. Name co-occurrence subgraph for $G^6_{A_1 A_5}$: used by [10]–[13], which rely on the co-occurrence property. If it does not hold for a domain, then no graph will be extracted.

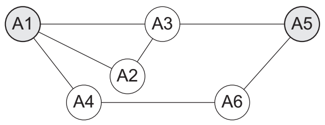


Fig. 6. Analyst-derived graph out of $G^6_{A_1 A_5}$: People connected via 'are-related' edges that correspond to co-authorships. Used by [31]. All information not about "author-writes-paper" is lost.

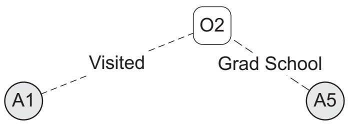


Fig. 7. $G^2_{A_1 A_5}$: this is what is used by [33]. The graph for the shortest path, used by [26], happens to be the same.

user may not be able to identify the cluster of pages associated with the real person she is looking for. We also provide the original list of (unclustered) pages from the standard search-engine and in this case the user can examine this list of pages.

VI. Related Work

Disambiguation and entity resolution techniques are key to any Web people search applications. In Section IV-B we have already overviewed several existing entity resolution approaches, pointing out that they rely primarily on analyzing object features for making their co-reference decisions. In this section we first overview our past work and compare it with existing disambiguation work in Section VI-A. We then discuss existing Web people search applications in Section VI-B.

A. Disambiguation

We have developed several disambiguation approaches in the past. The approaches in [27], [28], [36] solve a disambiguation challenge known as Fuzzy Lookup. To address the web page clustering problem, studied in this article, one needs to address a different type of disambiguation, known as Fuzzy Grouping. These disambiguation challenges are related but different, and we are unaware of any work that would solve the former using the latter. In Lookup, the algorithm is given a list of objects and the goal is for each references in the dataset to identify which object from that list it refers to [27], [28]. For grouping, no such list is available, and the goal is to simply group all of the references that co-refer [10], [13].

Besides the differences in the types of problems, the solution in [27], [28], [36] is also completely different: it reduces the disambiguation challenge into a global optimization problem whereas in this article a clustering approach is employed. While the solution in [17], [18] also addresses the entity resolution problem, the clustering algorithm proposed in this article, however, is different: it is based on correlation clustering (Section IV-A) and it employs a supervised learning approach for tuning to the dataset being processed (Section IV-D).

Most related techniques. The differences among our disambiguation methodology and most related existing work are multi-level (see Table I). Critical to understanding the differences is the notion of the *connection subgraph* G^L_{uv} for two nodes u and v, which is defined as the subgraph of the entity-relationship graph formed by the nodes and edges of all L-short simple paths between u and v [21]. The differences can be summarized as:

- **Level 1: Problem Type.** There are two different common types of the disambiguation challenge: (fuzzy) Lookup [27], [28], and (fuzzy) Grouping [10], [13].
- **Level 2: Data wrt G^L_{uv}.** Most of the existing techniques are different from our methodology as they do not analyze the same type of data: specifically our methodology is based on analyzing G^L_{uv} and the majority of the existing techniques do not analyze G^L_{uv} at all. For example, in the recent Web People Search Task (WEPS) at SemEval workshop [3], the participated algorithms that achieved the top three places all exploit extended rich features such as Named Entities or URLs extracted from the web pages, while no relationships are analyzed as in our approach. Then, there are some recent techniques that might be able to analyze portions of G^L_{uv} if certain conditions are met, e.g. see Table I. Let us take, for instance [12] by Bhattacharya and Getoor, which summarizes the approaches covered in [10], [11], [13]:
  - **Name co-occurrence.** The approach in [12] analyzes only co-occurrences of names of authors via publications for a publication dataset. Figure 4 illustrates a sample G^L_{uv} for the scenario where authors write publications and can be associated with some departments and organizations. When analyzing authors A_1 and A_4, the approach in [10], [11], [13] would only be interested in author A_3, which is a co-occurring author in publications P_1 and P_2, which are connected to A_1 and A_4 respectively. That is, [12] would be interested only in the subgraph shown in Figure 5. Our methodology instead analyzes the whole G^L_{uv}.
  - **Restrictions on types.** The approach in [12] understands only one type of relationship ("writes" in this case) and only two types of entities: person ("author") and container ("publication"). Our approach can analyze all of the types of relationships and entities present in G^L_{uv}.

  There is also recent work, e.g. [26], [31], [33], which builds on our work, but often still analyzes just portions of G^L_{uv}. For instance, [26] analyzes only the shortest path between u and v. The adaptive approach in [33] analyzes G^2_{uv}, see Figure 7. Another interesting solution [31] simply looks at people and connect then via 'are-related' relationships, see Figure 6. There, people can be 'related' if for instance their names co-occur on the same webpage. The solution however can analyze only one type of the 'are-related' relationship, whereas there can be different types of such relationships in a given domain, since people can be related for different reasons.
- **Level 3: Analysis of G^L_{uv}.** Those existing research efforts that analyze G^L_{uv} do it differently from our methodology. Our methodology is based on analyzing paths in P_{uv} and building mathematical models for $c(u, v)$, which are affinity-on RandomWalk-based models. The existing work (e.g.,

TABLE I
DIFFERENCES AMONG EXISTING TECHNIQUES WITH REGARD TO THE WAY THEY ANALYZE G_{uv}^L.

Approach	**Problem**	**Data wrt G_{uv}^L**	**Analysis of G_{uv}^L**	**Overall Solution**	**Dom. Dep.**
Proposed	Grouping	G_{uv}^L	Supervised Learning	Correlation Clustering	Indep.
[27], [28]	Lookup	Pruned G_{uv}^L	RandomWalk	Optimization Problem	Indep.
[36]	Lookup	G_{uv}^L	Adaptive	Optimization Problem	Indep.
[17]	Restricted Grouping	Pruned G_{uv}^L	RandomWalk	Partitioning	Indep.
[18]	Grouping	G_{uv}^L	Adaptive	Merging	Indep.
[10], [11]	Grouping	Name co-occurrence	Group Distance	Merging	Some domains
[13]	Grouping	Name co-occurrence	Group Probability	LDA, Gibbs Sampling	Some domains
[12]	Grouping	Name co-occurrence	Summary of [10], [13]	Summary of [10], [13]	Some domains
[31]	Grouping	[Are-related Graph][a]	Limited RandomWalk	Hierarchical Clustering	Some domains
[20]	Grouping	[Co-ref. Dep. Graph][b]	Analysis of co-reference dependencies	Merging	Indep.
[8]	Group Identification	[Web Graph][c]	Analysis of 'Link Structure' intersections	LS + A/CDC	Some domains
[26]	Lookup	Shortest Path	Length of SP	Ranking based on LSP	Some domains
[33]	Lookup	G_{uv}^2 (e.i., $L = 2$)	Adaptive RandomWalk	Sparse Matrix Multiplication, Kernel, Ranking	Indep.
[24]	Version of Lookup	Name co-occurrence	Similarities among the nodes	Minimum spanning tree, ranked output	Some domains, requires knowledge bases

[a]*Are-related Graph* is an analyst-derived graph of people (nodes), connected via "are-related" edges.
[b]*Co-ref. Dependence Graph* is an analyst-derived graph, encoding dependencies (edges) among co-reference decisions (nodes).
[c]*Web Graph* is a graph with one type of nodes (webpages) and one type of edges (hyperlinks).

[27], [28]) is often not path-centric and uses domain-specific or probabilistic (e.g., [13]) techniques to analyze the direct neighbors. Some techniques are based on just analyzing the shortest u-v path [26].

- **Level 4: Way to use $c(u, v)$.** Finally, once G_{uv}^L is analyzed, disambiguation approaches have to use the results of this analysis in making their co-reference decisions. The way we use it is also different. For instance, [10], [11] employ agglomerative clustering. In our previous work [27], [28] the disambiguation problem is converted into an optimization problem, which is then solved iteratively. In this article, a correlation clustering approach is employed (Section IV-A) which utilizes supervised learning for tuning itself to the dataset being processed (Section IV-A).
- **Level 5: Domain-independence.** Once our framework is provided with the entity-relationship graph for a dataset, it processes it the same way, regardless of the domain, that is, it is domain-independent. Some of the existing techniques are applicable to only certain types of domains or just one domain. For instance, the approach in [12] only applies to datasets where "noisy references to person entities (e.g. author names) are observed together (e.g. in publications)", i.e. domains where the co-occurrence property holds.

WSD. Natural Language Processing area studies related problems of "Word Sense Disambiguation" and "Word Sense Discrimination" [34], [37]. The goal of the first problem is to determine the exact sense of an ambiguous word given a list of word senses. The task of the second is to determine which instances of the ambiguous word can be clustered as sharing the same meaning.

The research on WSD mostly focuses on how to match the contextual features of word with the knowledge of word senses. It is important to decide which information to include for the context features to best represent the ambiguous word. On the other hand, how to use the external knowledge sources and what knowledge to exploit is a fundamental problem to solve in WSD. Many researchers have proposed various approaches, such as using lexical knowledge associated with a dictionary, building semantic network as is done by WordNet, etc. There are both supervised and unsupervised approaches for WSD problem, depending on whether or not there is a sense-tagged corpus available as training dataset. For unsupervised approaches, a trend is to use iterative or recursive algorithms to sense-tag the words with a finite number of processing cycles. In each step, such algorithms would either remove irrelevant senses or tag some words by synthesizing the information from previous steps. For supervised approaches, both hidden models (e.g., EM) and explicit models (e.g., Log Linear Model) have been used, depending on whether the features are directly associated with the word sense in training data.

If we view the ambiguous word as a reference and the word sense as an entity, then the two instances of WSD problem are similar to the Lookup and Grouping instances of Entity Resolution/WePS. Because of this similarity, the proposed approaches are frequently similar at a high level. There are some lower-level differences among the WSD and WePS problems. For instance, for WSD we can often assume that there is a dictionary of all word senses (perhaps imperfect), which can be employed sometimes quite effectively. Currently, such a complete dictionary is infeasible for WePS.[4] In addition, while a word typically has only a few semantic meanings, a reference to a person, e.g.,

[4]That is, there is no complete list of all people in the world and no dataset exists that associates the right keywords/tags/information with every person.

"J. Smith", can be much more uncertain. The different natures of domains also contribute to the differences of the WSD and WePS problems and solutions. For example, the part of speech tag associated with a word can significantly help in disambiguating the word sense in WSD. On the other hand, the POS tag assigned to a reference play much less significant role in the case of WePS.

There are many other interesting related disambiguation techniques, and we could not mention all of them in this article. Instead, we next describe the techniques and applications that deal directly with Web Search.

B. Web People Search

There are some research efforts [1], [4], [8], [14], [40], [41] that have explored the problem of entity disambiguation in the Web setting. We empirically compared our approach to some of the state of the art techniques in Section VII. Web people search applications can be implemented in two different settings. One is a *server-side* setting where the disambiguation mechanism is integrated into the search-engine directly. The other setting is a *middleware* approach where we build people search capabilities on top of an existing search-engine such as Google by "wrapping" the original engine. The middleware would take a user query, use the search engine API to retrieve top K Web pages most relevant to the user query, and then cluster those Web pages based on their associations to real people. The middleware approach is more common, as it is difficult to conduct realistic testing of the server-side approach, due to the lack of direct access to the search engine internal data. In this paper, we also take the middleware based approach to develop our algorithms.

There are a few publicly available Web search engines that offer related functionality in that Web search results are returned in clusters. Clusty (http://www.clusty.com) from Vivisimo Inc., Grokker (http://www.grokker.com), and Kartoo (http://www.kartoo.com) are search engines that return clustered results. However the clusters are determined based on intersection of broad topics (for instance research related pages could form one cluster and family pages could form another cluster) or page source, also the clustering does not take into account the fact that multiple persons can have the same name. For all of these engines, clustering is done based on entire Web page content or based on the title and abstract from a standard search-engine result.

ZoomInfo (http://www.zoominfo.com) search engine is an example of person search on the web. This search engine is similar to the one proposed in this article. It also extracts the named-entities and after that applies some machine learning and data mining algorithms to identify different people on the web. But this system has high cost and low scalability because the person information in the systems is collected primarily manually.

Among research efforts, such as [1], [4], [8], [11], [14], [40], [41], the approach of [24] is somewhat similar to our approach in that there is exploitation of relationships for disambiguation; however the assembly of relationships and approach to exploiting such relationships are quite different as we now explain. The approach in [24] starts with constructing a 'sketch' of each Web page (representative of a person with the name) which is essentially a set of attribute-value pairs for 'common' distinguishing attributes of a person such as his affiliation, job title etc. To construct the sketch however a variety of existing data sources (such as DBLP) and some pre-constructed specialized knowledge bases (such as TAP) are used. This approach is thus restricted to person searches where the persons are famous or prominent (famous enough for us to have compiled information about them in advance), whereas our approach does not rely on any such pre-compiled knowledge and thus will scale to person search for any person on the Web. Even in the case where pre-compiled knowledge exists the sketch comparison approach of [24] is limited since it relies on name co-occurrence, see Table I.

The approach of [8] is based on exploiting the link structure of pages on the Web, with the hypotheses that Web pages belonging to the same real person are more likely to be linked together. Three algorithms are presented for disambiguation, the first is just exploiting the link structure of Web pages, the second algorithm is based on word similarities between documents and does clustering using Agglomerative/Conglomerative Double Clustering (A/DC), the third approach combines link analysis with A/DC clustering.

VII. Experimental Results

In this section we empirically evaluate the proposed approach. First, in Section VII-A we describe the experimental setup. Next, Section VII-B covers experiments that evaluate the overall disambiguation quality of various algorithms. Then, Section VII-C studies the impact of the new cluster-based interface on web search. Finally, Section VII-D concludes the experimental evaluation with a study of the efficiency of the approach. Specifically, it shows that the overall query response time is largely determined by the time needed to preprocess the webpages, and that the clustering time itself is just a small fraction of the response time.

A. Experimental setup

Datasets. We conduct experiments on several real data sets for disambiguation of people on the Web. Each dataset has been created by querying the web using the Google or Yahoo search engine with a number of different queries. A query is either a person name, or a person name along with context keywords. The top 100 returned webpages of the Web search were gathered for each person. To get the "ground truth" for these datasets, the pages for each person name have then been assigned to distinct real persons by manual examination. The three datasets we have at our disposal are:

1) **WWW'05 Dataset.** Dataset used by Ron Bekkerman and Andrew McCallum in WWW'05 [8]. It contains webpages for 12 different people names.
2) **WEPS Dataset.** Dataset used in Web People Search Task (WEPS) at SemEval workshop [3]. The original WEPS data consist of the *Trial*, *Training*, and *Test* portions. The WEPS Trial portion contains webpages for 9 person names, and it is the same dataset used by Artiles et al. in [4]. The WEPS Training consists of webpages for 49 person names: 7 from Wikipedia, 10 from ECDL, and 32 from the U.S. Census. The WEPS Test part consists of 30 person names: 10 from Wikipedia, 10 from ACL06, and 10 from U.S. Census.
3) **Context Dataset.** This dataset is generated by us, by issuing 9 queries to Google, each in the form of a person name along with context keywords.

From the pages of the datasets, we constructed the graph to be analyzed, by extracting entities and creating the relationships as described in Section II. We used GATE [19] system for the extraction of named-entities (NEs) from the Web pages in the dataset. We used the system "as-is" i.e., without providing any

TABLE II
OVERALL QUALITY COMPARISON

Method	*WWW'05 Dataset*		*WEPS Training dataset*		*WEPS Test dataset*	
	B-Cubed	$\mathbf{F_P}$	**B-Cubed**	$\mathbf{F_P}$	**B-Cubed**	$\mathbf{F_P}$
Baseline	0.746	0.821	0.719	0.791	0.663	0.732
$s(u,v) = c(u,v)$	0.795	0.844	0.757	0.816	0.739	0.791
$s(u,v) = c(u,v) + \gamma f(u,v)$	0.805	0.850	0.771	0.837	0.763	0.814
$s(u,v) = c(u,v) + \gamma f(u,v), w^- = w^-(\hat{n})$	0.824	0.864	0.780	0.843	0.770	0.820

additional training, rules or data. The extraction of entities, while not perfect, is of reasonably high accuracy. We also employed some standard word stemming and fuzzy matching (consolidating "U.S." and "United States", etc.) over the extracted entities as a cleaning step.

To train the free parameters of our algorithm we apply leave-one-out cross-validation on smaller datasets, including WWW'05, WEPS Trial, and Context datasets. For the full WEPS dataset, before the "ground truth" for its WEPS Test portion was released by the organizers of the workshop, we tested our approach on the WEPS Training set by two-fold cross-validation. That is, we randomly divided the dataset into two halves, such that each of the subsets (i.e., Wikipedia, ECDL and US Census) are divided randomly into two halves. Then we trained on the 1st half and tested on the 2nd, and vice versa, and then we reported the average of the results. After the "ground truth" of WEPS Test portion became available, we trained our algorithm on the whole WEPS Training portion and tested on the WEPS Test portion.

Quality Evaluation Measures. Following suit of WEPS challenge [3] and Artiles et al. [4], we use the B-cubed [6] and F_P measures for assessing the quality of disambiguation.[5] B-cubed is considered to be a better measure than F_P and many other measures, as it is more fine-grained and it does not have as many measuring anomalies (counterintuitive measuring outcomes). Thus we will use B-cubed as our primary measure. More detailed discussion of quality metrics is beyond the scope of this paper.

Baseline Methods. In addition to comparing our algorithm to prominent solutions and the state of the art, we also use the Agglomerative Vector Space clustering algorithm with TF/IDF as our Baseline method. This method is widely employed as a benchmark to evaluate similar tasks, e.g. in [4], [8]. The threshold parameter for this method is trained the same way as discussed above.

Statistical significance test. We used the standard 1-tailed paired t-test, with $\alpha = 0.05$, to measure the statistical significance of our results when compared to other approaches. All of the results have been found to be significant, even for α as low as 0.001 for some experiments. The exception is the context experiment, where the results have been found to be significant for $\alpha = 0.07$.

[5]*B-cubed:* For each reference r (where references are webpages in this case) B-cubed computes set S_r of references that co-refer with r according to the ground truth. The term "co-refer" means refer to the same object. It also computes set A_r of references that co-refer with r according to the clustering produced by the algorithm. For reference r it computes $Precision_r = \frac{|A_r \cap S_r|}{|A_r|}$ and $Recall_r = \frac{|A_r \cap S_r|}{|S_r|}$. It then computes $Precision$ ($Recall$) as average $Precision_r$ ($Recall_r$) over all references. Finally, it computes F_B as the harmonic mean of $Precision$ and $Recall$.

F_P: F_P is computed as a harmonic mean of $Purity$ and $InversePurity$. Let $S = \{S_1, S_2, \ldots, S_{|S|}\}$ be the set of the original (ground truth) clusters of references (webpages in this case). Let $A = \{A_1, A_2, \ldots, A_{|A|}\}$ be the set of clusters according to a disambiguation algorithm. Then $Purity = \sum_{A_i \in A} \frac{|A_i|}{|R|} \max_{S_j \in S} \frac{|A_i \cap S_j|}{|A_i|}$ and $InversePurity = \sum_{S_j \in S} \frac{|S_j|}{|R|} \max_{A_i \in A} \frac{|A_i \cap S_j|}{|S_j|}$, where R is he set of all references (all webpages).

B. Testing Disambiguation Quality

In this section we present a set of experiments that study the quality aspect of the proposed approach. Experiment 1 assesses the overall quality of the proposed approach. Experiment 2 evaluates its quality on a disambiguation problem known as the Group Identification. Experiment 3 studies the quality on queries with context. The last experiment in this section evaluates the quality of the algorithm for generating cluster sketches.

Experiment 1 (Disambiguation Quality: Overall). Table II demonstrates the overall disambiguation quality results on WWW'05 and WEPS datasets. Here, $s(u,v) = c(u,v)$ represents the approach where only the connection strength is employed for disambiguation. That approach relies only on the extracted named entities and hyperlink information, and it does not use TF/IDF. Method $s(u,v) = c(u,v) + \gamma f(u,v)$ complements the previous method with the analysis of the features of webpages $f(u,v)$, in the form of their TF/IDF similarity. The last row in the table represent the approach which, in addition to the above, also picks w^- according to the function $w^-(\hat{n})$ of the predicted number of namesakes, as has been discussed in Section IV-E.

The table shows that, as expected, each subsequent method achieves better results than the previous one. The proposed approach gains 7.8% improvement in terms of B-cubed measure over the baseline approach on WWW'05 dataset and it gets 6.1% improvement on WEPS Training dataset (training is by two-fold cross-validation) and 10.7% improvement on WEPS Test dataset (training is on the whole WEPS Training set). The improvement is statistically significant at the 0.05 level for WWW'05 dataset and at the 0.001 level for WEPS datasets. The improvement is also evident in terms of F_p measure.

We also compare the results with the top runners in WEPS challenge [3]. The first runner in the challenge reports 0.78 for Fp and 0.70 for B-cubed measures. The proposed algorithm outperforms all of the WEPS challenge algorithms. Some of the learning approaches in WEPS challenge have not shown as good results as anticipated. This has been attributed to the fact that (1) the WEPS Training and Test datasets are small, and (2) these datasets have different properties such as different average ambiguity. These factors might have resulted in overfitting for the models used. On the other hand, Table II shows that the proposed learning model is stable on the WEPS dataset.

Disambiguation Quality per Namesake. Tables IV and V demonstrate more detailed (per queried name) results for the experiments on WEPS trial and and WWW'05 datasets. WEPS

TABLE III

QUALITY OF GROUP IDENTIFICATION. VALUES IN BRACKETS SHOW THE ABSOLUTE IMPROVEMENT OVER [8].

Name	#W	*WWW'05 Algo.*			*Baseline Algo.*			*New Algo.*		
		#C	**#I**	**F-measure**	**#C**	**#I**	**F-measure**	**#C**	**#I**	**F-measure**
Adam Cheyer	96	62	0	78.5	75	1	87.2(+8.7)	94	0	98.9(+20.4)
William Cohen	6	6	4	75.0	5	0	90.9(+15.9)	4	0	80.0(+5.0)
Steve Hardt	64	16	2	39.0	40	7	72.1(+33.1)	51	2	87.2(+48.2)
David Israel	20	19	4	88.4	14	2	77.8(-10.6)	17	2	87.2(-1.2)
Leslie Kaelbling	88	84	1	97.1	66	0	85.7(-11.4)	88	1	99.4(+2.3)
Bill Mark	11	6	9	46.2	9	17	48.6(+2.4)	8	1	80.0(+33.8)
Andrew McCallum	54	54	2	98.2	52	0	98.1(-0.1)	54	1	99.1(+0.9)
Tom Mitchell	15	14	5	82.4	15	2	93.8(+11.4)	12	5	75.0(-7.4)
David Mulford	1	1	0	100.0	0	1	0.0(-100.0)	1	0	100.0(+0.0)
Andrew Ng	32	30	6	88.2	27	1	90.0(+1.8)	25	1	86.2(-2.0)
Fernando Pereira	32	21	14	62.7	23	17	63.9(+1.2)	25	11	73.5(+10.8)
Lynn Voss	1	0	1	0.0	1	0	100.0(+100.0)	0	0	0.0(+0.0)
Overall	**455**	**313**	**47**	**80.3**	**327**	**47**	**82.4(+2.1)**	**379**	**24**	**92.1(+11.8)**

TABLE IV

QUALITY ON WEPS TRIAL DATASET. VALUES IN BRACKETS SHOW THE ABSOLUTE IMPROVEMENT OVER [4].

Name	**#**	**B-Cubed**	$\mathbf{F_P}$
Ann Hill	55	92.0	92.9(+4.9)
Brenda Clark	23	88.1	93.2(+5.0)
Christine King	29	79.0	84.6(+17.6)
Helen Miller	38	92.8	93.9(+31.9)
Lisa Harris	30	74.2	76.3(-6.7)
Mary Johnson	54	90.6	90.6(+15.6)
Nancy Thompson	47	78.6	81.7(+0.7)
Samuel Baker	38	70.8	72.8(-6.2)
Sarah Wilson	62	91.4	93.0(+23.0)
Mean/Overall	**42**	**84.2**	**86.5(+9.5)**

TABLE V

QUALITY ON WWW'05 DATASET.

Name	**#**	**B-Cubed**	$\mathbf{F_P}$
Adam Cheyer	2	97.9	99.0
William Cohen	10	87.3	93.1
Steve Hardt	6	80.4	88.3
David Israel	19	85.0	85.5
Leslie Kaelbling	2	98.9	99.4
Bill Mark	8	89.2	79.5
Andrew McCallum	16	93.3	95.2
Tom Mitchell	37	83.1	85.5
David Mulford	13	78.8	86.5
Andrew Ng	29	82.0	86.2
Fernando Pereira	19	71.4	78.8
Lynn Voss	52	42.1	59.6
Mean/Overall	**18**	**82.4**	**86.4**

trial dataset has also been used by Javier Artiles et al. in SIGIR'05 [4]. From these tables we can see that that 9 person names are queried in WEPS trial dataset and 12 names in WWW'05 dataset. The '#' field shows the number of namesakes for a particular name in the corresponding 100 webpages. Table IV compares the results of the proposed approach with those of [4]. In [4] the authors employ an enhanced version of agglomerative clustering based on TF/IDF. The table shows that the proposed approach outperforms that of [4] by 9.5% in terms of F_P measure.[6] The achieved improvement is statistically significant at the 0.05 level. Table V demonstrates the results for a similar experiment, but on WWW'05 dataset. □

The improvement is achieved since the proposed disambiguation method is simply capable of analyzing more information, hidden in the datasets, and which [4], [8] do not analyze.

Experiment 2 (Disambiguation Quality: Group Identification). In [8] the authors propose an unsupervised approach for Group Identification: a related-but-different problem to the one studied in this paper. In that problem the algorithm is given N names of N people that are somehow related, e.g., these names are found in somebody's address book. The task is to find the webpages related to the meant N people.

We have modified our algorithm to apply it to that problem, as explained in the electronic appendix in more detail. That algorithm outperforms [8] by 11.8% of F-measure as illustrated in Table III. In this experiment F-measure is computed the same way as in [8].[7] The field "#W" in Table III is the number of the to-be-found webpages related to the namesake of interest. The field '#C' is the number of webpages found correctly and the field '#I' is the number of pages found incorrectly in the resulting groups. We can see that the baseline algorithm also outperforms the algorithm proposed in [8]. The baseline algorithm utilizes only one free parameter, the threshold, which in our case is trained from data. The difference between WWW'05 algorithm and our new algorithm is statistically significant at the 0.05 level.

The work in [9] is the latest follow up to the work in [8] we are aware of. In it, the authors have extracted all the hyperlinks contained in the 1085 webpages of WWW'05 dataset and crawled the Web three hops from those links, retrieving additional webpages. This costly process has resulted in 669,847 webpages overall. By analyzing all these webpages, the authors achieved 83.9% F-measure. This is still 8.2% less than the results of the approach proposed in this paper, which achieves 92.1% and it does not yet analyze all this additional data. □

[6] F_P is referred to as $F_{\alpha=0.5}$ in [4].

[7] F-measure: let S_i be the set of the correct webpages for cluster-i, and A_i be the set of the webpages assigned to cluster-i by the algorithm. Then, $Precision_i = \frac{|A_i \cap S_i|}{|A_i|}$, $Recall_i = \frac{|A_i \cap S_i|}{|S_i|}$, and F is their harmonic mean.

TABLE VI
RESULTS FOR THE CONTEXT DATASET.

				Baseline		New Alg	
Query	**Query Text**	**# Pages**	**#**	**B-Cubed**	**F_P**	**B-Cubed**	**F_P**
Q_1	"Andrew McCallum" music	100	29	53.5	69.6	73.8	81.7
Q_2	"Andrew McCallum" poster	100	4	87.6	93.5	76.5	86.6
Q_3	"Andrew McCallum" dance	100	30	57.7	68.7	65.0	75.1
Q_4	"Andrew McCallum" uci	100	1	78.3	88.9	95.9	98.0
Q_5	"George Bush" bible scholar	98	13	61.7	77.5	74.1	84.6
Q_6	"William Cohen" cmu	100	7	89.4	94.5	91.1	95.4
Q_7	"William Cohen" uci	100	17	71.7	82.4	55.1	73.8
Q_8	"Tom Mitchell" psychology	98	17	63.2	78.4	76.0	85.5
Q_9	"Tom Mitchell" soccer	97	40	56.6	69.3	69.6	77.9
Mean		**99**	**18**	**68.9**	**80.3**	**75.2**	**84.3**

Experiment 3 (Disambiguation Quality: Queries with Context). We generated a dataset by querying Google with a person name and context keyword(s) that is related to that person. We used 9 different queries. The statistics for this dataset is illustrated in Table VI. For instance, the table shows that for query $Q_8 =$ *"Tom Mitchell" psychology*, 98 meaningful pages were found (the rest are empty pages) and there are 13 namesakes for Tom Mitchell in those pages. Table VI presents the disambiguation quality results for the proposed and baseline algorithms. The proposed approach outperforms the baseline by 6.3% of B-cubed measure. The difference between the baseline and the new algorithm is significant at the 0.07 level. □

While the proposed algorithm factors in the same information used by the baseline, it ultimately makes its own decisions, which are largely driven by analyzing the connections. Table VI demonstrates that point: the results are worse than those of the baseline for the two cases, but they are better on average.

Experiment 4 (Quality of Generating Cluster Sketches). In Section II we have presented an algorithm for generating representative keywords to summarize each cluster. Table VII illustrates the output of that algorithm for 'Andrew McCallum' query on WWW'05 dataset. The keywords and phrases are shown in their stemmed versions. The table shows only the top-10 keywords for each cluster for the sake of clarity. Each cluster has different set of keywords. So if the search is for UMass professor Andrew McCallum, his cluster can easily be identified with the terms like *"machine learning"* and *"artificial intelligence"*, as well as with the keywords like *extract*, *model*, and *classification*. □

TABLE VII
RESULTS FOR "ANDREW MCCALLUM" QUERY IN WWW'05 DATASET.

Group Name	**Cluster Summary**
UMASS Professor	learn, artifici intellig, machin, proceed, machin learn, extract, model, classif, comput, data
ACOSS President3	student, incom, univers, educ, fee, famili, low, east timor, timor, cent
ACOSS President11	acoss, childcar, welfar, australian, council, servic, social servic, incom, famili, presid,
Teacher	aclandburghlei, camden, burghlei, sch, acland, school, english, month, biographi, uk
Writer	competit, stori, read, toowrit, winner, author, mous, nep, toowritepoetri, children
Artist3	philosophi, mentalfloss, festiv, scienc, blog, pietersen, ultim, coburg, guthri, flyer
Artist1	rockbox, jukebox, archo, studio, tedford, stuart, bod, boru, donaghi, melih
Photographer	amico, imag, collect, librari, conspir, davidrumsei, penitentiari, trial, williamstown, court
Kid	theatr, shakespear, tempest, grouch, juliet, romeo, crew, dream, festiv, night
Medical Professor1	ccfp, kari, kgh, leroyv, med, puddi, queensu, jennif, em, md
Customer Support	initil, opensr, domain, tucow, loui, dn, chronolog, protect, sent, client
Humanist	dreambook, humanist, color, human, ge, homepag, plz, secular, vacat, individu
Painter	height, imag, larger, price, sherwin, keith, cub, leopard, sefton, richard
ACOSS President1	hospit, nurs, health, australian, servic, treatment, kingston, accid, local, care
Medical Professor1	inquest, ontario, coron, ministri, death, eastern ontario rugbi union, ontario rugbi union, ottawa, rugbi union, union and leagu,
Poll Analyst	declan, zealand, fc, jul, video, game, horn, censorship, fitug, offici
Poll Analyst	regul, electron, transact, swain, act, notic, paper, disclosur, internetnz, discuss
Economist	acidif, soil, cost, farm, acid, agricultur, land, econom, lime, research
Technician	chemistri, depart, otago, chemic, comput, laboratori, univers, calm, comput support cooper work

C. Impact on Search

Comparing the effectiveness of cluster-based people search to the traditional search is a complex task, as it implies too many unknowns: what the user is looking for exactly, which background/context information she knows and intends to use in her query, how the user will react on partially examined output in the returned results, and so on. To perform a quantitative comparison, we used the following methodology.

User Observations. A user is interested in retrieving the webpages of a particular person. The user queries the search engine with the name of that person, e.g. William Cohen, and then scans through the top K pages in order to satisfy the objective of finding all the webpages of that person among the top K pages. In case of a traditional search interface, at each **observation** i, where $i = 1, 2, \dots, K$, the user looks at the sketch provided for the i-th returned webpage. We assume that by doing so the user can decide whether the page is *relevant* to the person she was looking for or *irrelevant*.

For the new interface, supported by a cluster-based people search, the user first looks at the "people search" interface. The user sequentially reads cluster sketches/descriptions, until on the m-th observation the user find the cluster of interest. The user then clicks on that cluster, and the systems then shows the original set of K Web pages returned by the search engine, except that the webpages are ordered differently. Specifically, first are the set of pages S that our algorithm identified for that namesake,

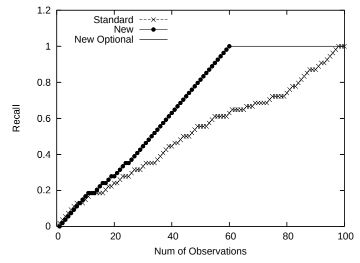


Fig. 8. UMass Prof: Recall

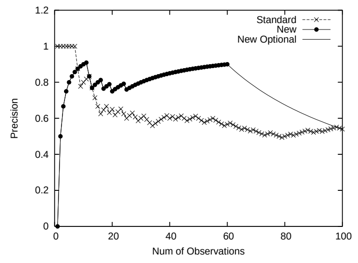


Fig. 9. UMass Prof: Precision

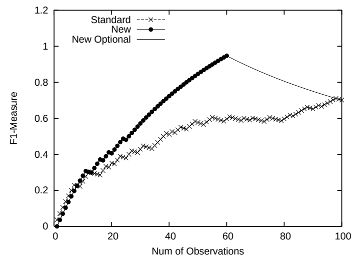


Fig. 10. UMass Prof: F1-measure

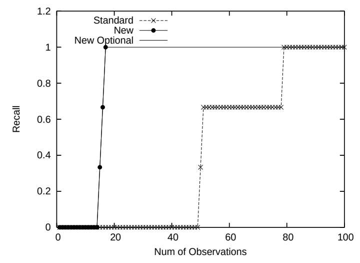


Fig. 11. Cust. Sup.: Recall

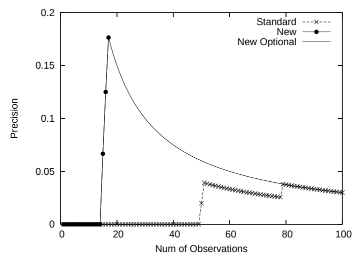


Fig. 12. Cust. Sup.: Precision

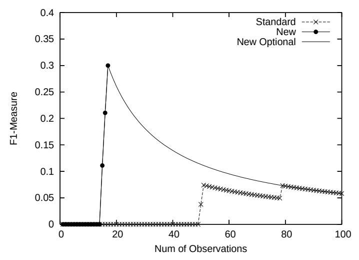


Fig. 13. Cust. Sup.: F1-measure

next are the pages in the order of their similarity to the pages in S. At each subsequent observation, the user examines the sketch for each page, as in the standard interface, and decides relevant/irrelevant in a similar fashion. Notice that in a cluster-based search, it is possible that none of the cluster definitions satisfies the user (i.e., matches the person she is interested in). In such a case, the user can retrieve the original K webpages returned by the search engine. Notice that in practice the user can make mistakes in deciding relevant/irrelevant based on sketches. Thus, the reported quality results will be optimistic for both the new and standard search interfaces.

Measures. We compare the quality of the new and standard interface using Precision, Recall, and F-measure. On the i-th observation the precision shows the fraction of relevant pages among all the webpages examined so far. The recall on the i-th observation shows the fraction of related webpages out of all the related pages, discovered so far. Notice that using the new interface the user starts examining the 1-st webpage only on the $(m+1)$-th step, after locating the right cluster on the m-th step.

Recall plots are useful in computing another metric: how many observations are needed to discover a certain fraction of relevant pages. In general, the fewer observations are needed in a given interface, the faster the user can find the related pages, and thus the better is the interface. Each figure in this section shows three curves: one for the standard interface and two for the new interface. The new interface knows the number of webpages $|S|$ in the cluster S the user chooses to explore. The user may opt to examine only those $|S|$ webpages suggested by the algorithm and then stop. This case is represented by 'New' curve in the figure. Optionally, the user might choose to continue exploring the rest of the webpages. The latter situation is represented with 'New Optional' curve.

Experiment 5 (Impact on Search). This experiment consists of three parts. The first two parts study two common cases (1) A search for a namesake whose webpages form the largest cluster, these webpages also tend to be first pages in search, and (2) A search for a regular cluster. The third part studies the overall performance averaged over all the namesakes in the dataset.

Case 1: First-Dominant Cluster. Figures 8–10 plot the measures for Andrew McCallum the UMass Professor. His pages tend to appear first in Google, they form the first group, which is also the largest one. The Recall figure shows that one needs to do 44 observations in the standard interface to discover half (50%) of the pages (27 out of 54) of the UMass Prof., while in the new interface one need to do just 33 observations total. To discover 90% of the relevant pages, one needs to do 92 observations in the standard interface and only 55 in the new one. The general trend in the plots in this section is that the Precision, Recall, and F-measure for the new interface either dominate, or are comparable to, those of the standard interface.

Case 2: Regular Cluster. Figures 11–13 plot the same measures for Andrew McCallum the Customer Support person. His cluster consists of 3 pages that appear more toward the end in Google search. His group is one of the last groups. To get 50% of his cluster one needs to do 51 observations in the standard interface and only 16 observations in the new interface. For 90%, it is 79 observations in for the standard interface and 17 observations for the new interface.

Case 3: Average. Figures 14–16 plot the average of Recall, Precision, and F measures for search impact on WWW'05 dataset, by averaging over all the namesakes in this dataset. It should be noted that some of the person names have many namesakes, e.g., David Israel has 45 namesakes, Bill Mark has 52 namesakes, etc. Therefore, for some of the namesakes, both the standard and new interfaces would require first doing many observations to find even the first relevant webpage. After averaging, this leads to small overall values for measures. Figures 14–16 show that, even with the imperfect clustering, the curves for the new interface largely dominate those for the standard interface. The figures do not capture another advantage of the new interface: its ability to suggest when to stop the search, since the algorithm knows the number of elements in each cluster. □

Impact on Search With Context. In general it is not hard to imagine scenarios where a good choice of keywords would identify a person really well, so that all the returned top K webpages would belong to just one namesake. In that case one can expect to see no difference between the new and the

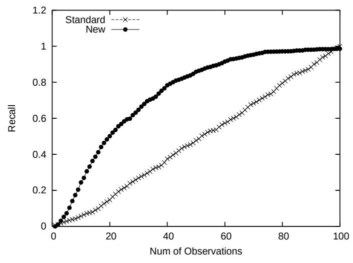


Fig. 14. Average Recall.

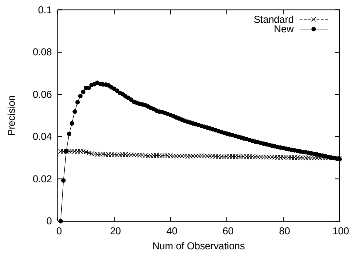


Fig. 15. Average Precision.

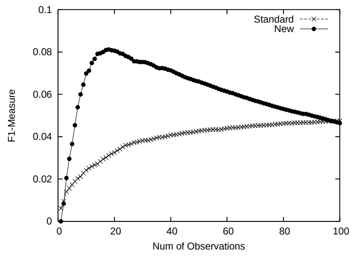


Fig. 16. Average F1-measure.

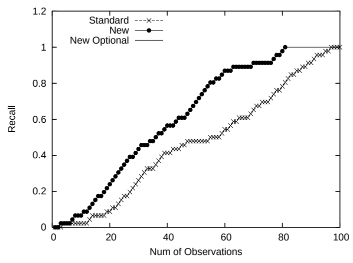


Fig. 17. Umass, Music: Recall

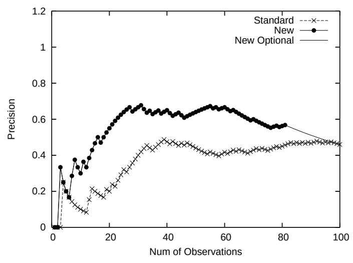


Fig. 18. Umass, Music: Precision

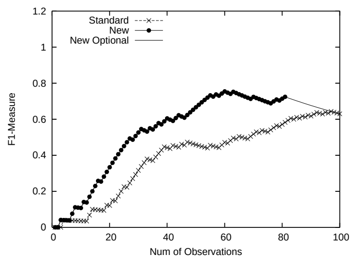


Fig. 19. Umass, Music: F1-measure

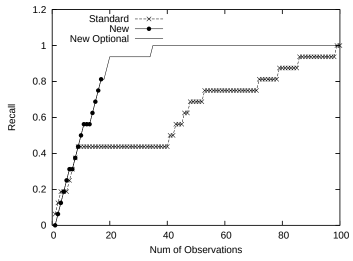


Fig. 20. Musician: Recall

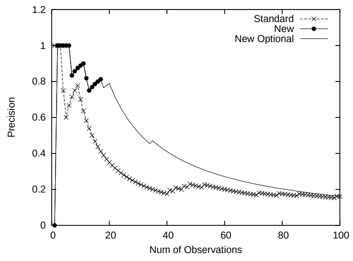


Fig. 21. Musician: Precision

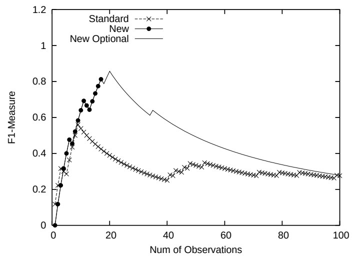


Fig. 22. Musician: F1-measure

standard interface. But quite often searches with context still return results that contain several namesakes, see for example Table VI. Figures 17–22 plot the impact on search when context is used. In this case the query is: *"Andrew McCallum" music*. The number of namesakes for that query is surprisingly large: 23. The reason is that webpages often contain advertisements, e.g. links to websites that sell music. Figures 17–19 plot the impact results for the case where the user in his query has meant Andrew McCallum the UMass professor, who is interested in music. Figures 20–22 plot the same for the case where the user meant Andrew McCallum the DJ/musician. In both cases the new interface performs better than the standard one. The user finds the 90% of the documents related to DJ/musician in 20 observations with the new interface, whereas it takes 90 observations with the standard interface. On the other hand to find the 90% of the documents for the UMass professor, user should examine the 60 pages in the new interface (90 pages with the current interface).

D. Efficiency

Experiment 6 (Efficiency). The overall approach first downloads and preprocesses pages before applying the actual clustering algorithm. That takes 3.82 seconds per webpage mainly due to the fact that we use a third party Named Entity extractor, GATE, to extract named entities, the speed of which we cannot control.[8] However, the preprocessing cost disappears if the server-side approach is employed instead of the wrapper approach, since this preprocessing can be done off line beforehand. The clustering algorithm itself executes in 4.7 seconds on average per queried name. □

[8]Our preliminary experiments indicate that it takes 0.36 seconds per webpage for another extractor, called Stanford Named Entity Recognizer [23]. The quality of the results of SNER vs. GATE are comparable.

VIII. Conclusions and Future Work

In this paper we only evaluated our core technique. We have attempted to answer the question of which maximum quality our approach can get if it uses only the information stored in the top-k webpages being processed. There are several interesting directions for future work. One of the most promising directions is to employ external data sources for disambiguation as well. This includes using ontologies, encyclopedias, and the Web [30]. Another direction is to use more advances extraction capabilities that would allow: (a) a better interpretation of extracted entities by taking into account the roles they play with respect to each other (boss of somebody, student of somebody) (b) extraction of relationships, as currently the algorithm relies primarily on co-occurrence relationships only. We plan to develop disambiguation algorithms for other people search problems that have different settings. Finally, we would like to work on algorithms for a generic *entity* search, where entities are not limited to people.

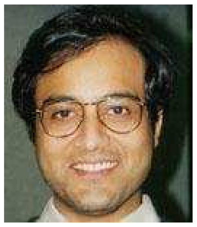
References

[1] R. Al-Kamha and D. W. Embley. Grouping search-engine returned citations for person-name queries. In *Proc. of WIDM*, 2004.
[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.
[3] J. Artiles, J. Gonzalo, and S. Sekine. The SemEval-2007 WePS Evaluation: Establishing a benchmark for the Web People Search Task. In *Proc. of Intl. Wrks. on Semantic Evaluations (SemEval)*, June 2007.

[4] J. Artiles, J. Gonzalo, and F. Verdejo. A testbed for people searching strategies in the WWW. In *SIGIR*, 2005.
[5] R. Baeza-Yates and B. Riberto-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
[6] A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. http://citeseer.ist.psu.edu/153897.html, 1998.
[7] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Foundations of Computer Science*, pages 238–247, 2002.
[8] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *WWW*, 2005.
[9] R. Bekkerman, S. Zilberstein, and J. Allan. Web page clustering using heuristic search in the web graph. In *Proc. of IJCAI*, 2007.
[10] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD Workshop*, 2004.
[11] I. Bhattacharya and L. Getoor. Relational clustering for multi-type entity resolution. In *Proc. of MRDM Workshop*, 2005.
[12] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 29(2):4–12, June 2006.
[13] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *Proc. of SIAM Data Mining Conf.*, 2006.
[14] D. Bollegala, Y. Matsuo, and M. Ishizuka. Extracting key phrases to disambiguate personal names on the web. In *CICLing*, 2006.
[15] S. Chaudhuri, K. Ganjam, V. Ganti, R. Kapoor, V. Narasayya, and T. Vassilakis. Data cleaning in Microsoft SQL Server 2005. In *ACM SIGMOD Conference*, 2005.
[16] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, 2003.
[17] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Exploiting relationships for object consolidation. In *Proc. of International ACM SIGMOD Workshop on Information Quality in Information Systems (ACM IQIS 2005)*, 2005.
[18] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Adaptive graphical approach to entity resolution. In *Proc. of ACM IEEE Joint Conference on Digital Libraries (ACM IEEE JCDL 2007)*, 2007.
[19] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of ACL*, 2002.
[20] X. Dong, A. Y. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.
[21] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *SIGKDD*, 2004.
[22] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of Amer. Statistical Association*, 64(328):1183–1210, 1969.
[23] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of ACL*, 2005.
[24] R. Guha and A. Garg. Disambiguating people in search. In *Stanford University*, 2004.
[25] F. Hillier and G. Lieberman. *Introduction to operations research*. McGraw-Hill, 2001.
[26] R. Holzer, B. Malin, and L. Sweeney. Email alias detection using social network analysis. In *Proceedings of the ACM SIGKDD Workshop on Link Discovery: Issues, Approaches, and Applications*, 2005.
[27] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems (ACM TODS)*, 31(2):716–767, June 2006.
[28] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM International Conference on Data Mining (SIAM Data Mining 2005)*, Newport Beach, CA, USA, April 21–23 2005.
[29] D. V. Kalashnikov, S. Mehrotra, Z. Chen, R. Nuray-Turan, and N. Ashish. Disambiguation algorithm for people search on the web. In *IEEE ICDE Conference*, Istanbul, Turkey, April 16–20 2007.
[30] D. V. Kalashnikov, R. Nuray-Turan, and S. Mehrotra. Towards breaking the quality curse. A web-querying approach to Web People Search. In *Proc. of Annual International ACM SIGIR Conference*, July 20–24 2008.
[31] B. Malin. Unsupervised name disambiguation via social network similarity. In *Ws. on Link Analysis, Counterterrorism, & Security*, 2005.
[32] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*, 2004.
[33] E. Minkov, W. Cohen, and A. Ng. Contextual search and name disambiguation in email using graphs. In *SIGIR*, 2006.
[34] I. Nancy and V. Jean. Word sense disambiguation: The state of the art, 1998.
[35] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.
[36] R. Nuray-Turan, D. V. Kalashnikov, and S. Mehrotra. Self-tuning in graph-based reference disambiguation. In *DASFAA Conference*, Bangkok, Thailand, April 9–12 2007.
[37] H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.
[38] P. Singla and P. Domingos. Multi-relational record linkage. In *MRDM Workshop*, 2004.
[39] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths. Probabilistic author-topic models for information discovery. In *SIGKDD*, 2004.
[40] C. Tiwari. Entity identification on the web. Technical report, I.I.T. Bombay, 2006.
[41] X. Wan, J. Gao, M. Li, and B. Ding. Person resolution in person search results: Webhawk. In *ACM CIKM*, 2005.

Dmitri V. Kalashnikov received the Diploma (summa cum laude) in Applied Mathematics and Computer Science from Moscow State University, Russia, in 1999 and the PhD degree in Computer Science from Purdue University in 2003. Currently, he is a Researcher at the University of California, Irvine. He has received several scholarships, awards, and honors, including an Intel Fellowship and Intel Scholarship. His current research interests are in the areas of entity resolution & disambiguation, web people search, spatial situational awareness, moving-object databases, spatial databases, and GIS.

Zhaoqi (Stella) Chen received the BSc in Computer Science from Fudan University, China, in 1998 and the MSc in Computer Science from Peking University, China, in 2001. She has received multiple scholarships, awards and honors including an IBM Scholarship and Motorola Scholarship. She is currently a Graduate Student Researcher at University of California at Irvine. Her research interests are in entity resolution, web people search and data mining techniques.



Sharad Mehrotra is currently a Professor in the Department of Computer Science at the University of California, Irvine (UCI) and the Director of the Center for Emergency Response Technologies. Previously, he was a Professor at the University of Illinois at Urbana-Champaign (UIUC). He received his Ph.D. in Computer Science from the University of Texas at Austin in 1993. He has received numerous awards, and honors,including SIGMOD best paper award 2001, DASFAA best paper award 2004, and CAREER Award 1998 from NSF. His primary research interests are in the area of database management, distributed systems, and data analysis.

Rabia Nuray-Turan is a PhD Candidate in the Computer Science Department of University of California, Irvine, USA. Her research interests include information retrieval, web mining, data cleaning and machine learning. She received her MSc in computer engineering from Bilkent University, Ankara, Turkey.

Appendix

Group Identification

In [8] the authors have developed an unsupervised Web disambiguation algorithm that exploits Web links present in the data set. Since we had adopted the data set produced by [8], we wanted to compare the quality of the two disambiguation algorithms. The problem we faced was that the disambiguation problem studied in this article is different from the one studied in [8]. This article solves a well-recognized Web Page Clustering problem, while the approach in [8] addresses less studied *Group Identification* problem.

In the formulation of the Group Identification problem, it is assumed that N (e.g., N=12) people names are found together, for instance, on a conference Web page, or in somebody's email folder, etc. The mere fact that those names are mentioned together indicate that specific real people are meant there, which are related in some unknown way. The task is to gain more information about those N meant people.

Specifically, Google is queried N times, for each person, with the person's name as the query. The top-100 web pages returned by Google are stored per each person. The goal is to identify among those Web pages only those pages that refer to the N specific people, that were meant by the list of names.

```
GROUP-IDENTIFICATION($Q, K,$ Ontology)

1   $C \leftarrow \emptyset$ // sets of set of clusters
2   $Q \leftarrow$ Person-Names
3   $R \leftarrow \emptyset$ // pages mentioning the persons in the groups
4   Ontology $\leftarrow$ LOAD-ONTOLOGY()
5   for each person query $q_i \in Q$
6       $C_i \leftarrow$ PROCESS-QUERY($q_i, K,$ Ontology)
7   for each person query $q_i \in Q$
8       $X \leftarrow$ Search the occurrence of other names $(Q - q_i)$ in $C_i$
9       $R \leftarrow R \cup X$
10  return $R$
```

Fig. 23. Naive Algorithm for Group Identification Problem.

Even though the two disambiguation challenges are different, it is easy to design a naive algorithm for solving the Group Identification problem, that would build on any Web Page Clustering algorithm. The pseudo-code for one such algorithm is illustrated in Figure 23. The naive approach first performs web page clustering on each set of the top-100 web pages, per each of N persons, to determine their namesakes. After that, the task becomes to identify the right (meant) namesake per each name. That is, for namesakes of Andrew McCallum, the algorithm should identify which namesake is the meant one (the UMass Professor in this case). The naive algorithm does this by counting which groups contain web pages that mention at least one of the other $N - 1$ people names (William Cohen, Tom Mitchell) from the original list of N people. If two or more groups have such pages, those groups are merged into one. For instance, the UMass Professor might be split into two groups by mistake of the grouping algorithm. But if the two groups both mention, say, William Cohen, then they will be merged into one group.

We used the above approach to compare the proposed disambiguation algorithm with the approach proposed in [8].[9] The proposed disambiguation algorithm achieves the F-measure of 92.1%, which is 11.8% improvement over the best result of 80.3% reported in [8].

Also notice that the task of Group Identification, addressed in [8], can be procedurally viewed as an optimization over *one* specific namesake in each group of namesakes with the same name (e.g., optimization over only Cohen the CMU Prof, in all Cohen's Web pages). The task of Web page clustering, addressed in this paper, can be viewed as an optimization over *all* the namesakes, which is more challenging. Thus, one would expect that any approach solving only Group Identification would get better results on the namesake of interest, than those of a general grouping algorithm, since the former has the advantage over the latter.

[9]We note that this straightforward algorithm could be perhaps improved further by using more robust criteria for grouping individuals, e.g., by using similarity joins over clusters.