Towards Breaking the Quality Curse. A Web-Querying Approach to Web People Search.*

Dmitri V. Kalashnikov
dvk@ics.uci.edu

Rabia Nuray-Turan
rnuray@ics.uci.edu

Sharad Mehrotra
sharad@ics.uci.edu

Department of Computer Science
University of California, Irvine
Irvine, CA 92697, USA

ABSTRACT

Searching for people on the Web is one of the most common query types to the web search engines today. However, when a person name is queried, the returned webpages often contain documents related to several distinct namesakes who have the queried name. The task of disambiguating and finding the webpages related to the specific person of interest is left to the user. Many Web People Search (WePS) approaches have been developed recently that attempt to automate this disambiguation process. Nevertheless, the disambiguation quality of these techniques leaves a major room for improvement. This paper presents a new server-side WePS approach. It is based on collecting co-occurrence information from the Web and thus it uses the Web as an external data source. A skyline-based classification technique is developed for classifying the collected co-occurrence information in order to make clustering decisions. The clustering technique is specifically designed to (a) handle the dominance that exists in data and (b) to adapt to a given clustering quality measure. These properties allow the framework to get a major advantage in terms of result quality over all the latest WePS techniques we are aware of, including all the 18 methods covered in the recent WePS competition [2].

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]:Information Search and Retrieval; H.3.5[**Information Storage and Retrieval**]:Online Information Services- *Web-based services*

General Terms: Algorithms, Experimentation, Measurement

Keywords: Clustering, Skyline based classifier, WEPS, Web People Search, Named Entity Co-Occurrences, Social Network, Web Querying

1. INTRODUCTION

The rapid growth of the Internet has made the Web a popular place for collecting information. Today, Internet users access billions of web pages online using search engines. Searching for a specific person is one of the most popular search queries. Some statistics suggest that such searches account for as much as 5-10% of all search queries [14].

When a web search engine is queried with a person name, it returns a collection of webpages. Let $D = \{d_1, d_2, \ldots, d_k\}$ be the set of the top k returned results. The webpages in D are related to the set of one or more namesakes who have the queried name, where these namesakes are unknown beforehand. The goal of a Web People Search (WePS) system is to automatically cluster the webpages in D such that each cluster corresponds to a namesake.

Most popular search engines of today such as Google and Yahoo! do not yet provide WePS capabilities, leaving the disambiguation task of finding the webpages of a person of interest to the user. For instance, when Google is queried with the name "William Cohen", the top 100 returned webpages refer to at least 10 different namesakes whose name is "William Cohen". Thus the task of locating all the webpages about a specific William Cohen, for instance the CMU professor, can present a challenge to the user. Frequently the problem does not disappear even if the context keywords are provided along with the person name. For instance, the returned webpages for the above query with the additional keyword "CMU" (a) will not include his webpages that do not include "CMU", making the recall lower, and (b) will actually still refer to more than one namesake, thus not leading to perfect precision either. Such issues explain the motivation and the ever increasing interest in WePS systems, whose goal is to provide the user with more structured and more powerful web search capabilities for this specific type of query.

One of the key challenges that needs to be overcome to make the WePS functionality a reality, is to build a WePS system that is capable of reaching high disambiguation quality. This has proven to be a nontrivial task, as corroborated by the results of a recent WePS competition [2].

Given that there is already a significant number of WePS techniques it is important to understand what distinguishes this work from other solutions. The proposed approach is a two-step clustering method. First the Named Entities (NEs) are extracted from each webpage $d \in D$. The extracted NEs are people names and organizations that represent the social network of the namesake referred to by d. Then, for each distinct pair of webpages $d_i, d_j \in D$ the framework computes TF/IDF similarity based on NEs only. If this similarity is sufficiently large, the two pages are merged into one cluster.

*This research was supported by NSF Awards number 0331707 and 0331690.

SIGIR'08, July 20–24, 2008, Singapore.

In the second step of the approach, for each d_i, d_j pair that is still unmerged, the algorithm forms queries to a Web search engine that combine the NEs from d_i and d_j. The web search engine results are the co-occurrence counts that tell how often elements of the two social networks co-occur on the web and thus how strongly they are related. These counts are computed both, in the context of the queried name and without it. The counts are then transformed to form similarity features for the d_i, d_j pair. While there is research efforts on using related feature types for some domains, e.g. [7, 11, 19], we are unaware of any work that employs such features for WePS. Collecting such features is a time consuming operation, unless it is performed at a server (web search engine) side. Furthermore, web search engine APIs have significant restrictions on the number of queries allowed per day. Thus, the proposed approach in its current form is inherently a server-side solution.

Proper features selection is one of the important contributions of this paper. However, the key is to develop an algorithm to utilize the chosen features well in order to build a WePS system that gets high disambiguation quality results. We have developed a skyline-based feature classification algorithm for this purpose. This algorithm is the **main contribution** of this paper. The algorithm gains its advantage by (a) taking into account dominance that exists in the co-occurrence data and (b) by using a supervised learning to tune itself to both, the WePS domain and a given quality metric.

The rest of this paper is organized as follows. Section 2 summarizes the related research work. Section 3 provides an overview of the proposed approach. It is followed by Section 4 that covers the steps of the algorithm in more detail. The proposed approach is empirically evaluated in Section 5 and compared to some of the state of the art solutions. Section 6 concludes the paper by highlighting the impact of the proposed solution.

2. RELATED WORK

The Web People Search challenge is closely related to the well-studied Entity Resolution problem. In our previous work we also have developed interrelated techniques to solve various Entity Resolution challenges, e.g. [9, 15–18, 23]. The approach covered in the paper, however, is *not* related to those techniques. The key algorithm for training the skyline-based classifier is new. In fact, we are unaware of any Entity Resolution technique that would use a similar approach under any context.

There are several research efforts that address specifically Web Person Search and related challenges [1–3, 5, 6, 8, 22, 25]. The approach of [5] 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 and forming clusters based on link analysis, 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. The work in [21] clusters documents based on the entity (person, organization, and location) names and can be applied to the disambiguation of semi-structured documents, such as Web pages. The primary new contribution is the development of a document generation model that explains, for a given document how entities of various types (other person names, locations, and organizations) are "sprinkled" onto the document.

In Sem-Eval 2007, a workshop on WePS task was held [2]. Sixteen different teams from different universities have participated to the task. The participated systems utilized named entities, tokens, URLs, etc that exist in the documents. It has been shown that as the extracted information increases the quality of the clustering increases. Use of different NE recognition tools affects the results of clustering as well. The NE based single link clustering was the one of the top three systems [12], because the named entity network alone enables to identify most of the individuals.

There are also 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. 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/) and Spock (http://www.spock.com/) are commercially available people search engines.

Recently, researchers have started to use external databases, such as ontology and Web Search engine in order to improve the classification and clustering qualities in different domains [7, 11, 13, 19]. For example, querying the Web and utilizing the search results are used for Word Sense Disambiguation (WSD) [7] and record linkage in publications domain [11, 19]. The number of queries to a search engine is a bottleneck in these approaches. Hence, the study in Kanani and McCallum [19] tries to solve the problem in the case of limited resources. The suggested algorithm increased the accuracy of data cleaning while keeping the number of queries to a search engine minimal. Similarly the approach in [11] uses the web as a knowledge source for data cleaning. The study proposed a way to formulate queries and used some standard measures like TF/IDF similarity to compute the similarity of two different references to an entity. The work in [12] is a complementary work to the one in [19]. In [7] the authors proposed to use the co-occurrence counts for disambiguation of different meanings of words. The authors used web-based similarity measures like WebJaccard, WebDice, and so on. These measures are also utilized as features for the SVM based trainer along with a set of token based features, where the trainer learns the probability of two terms being the same.

In this paper, we study a similar approach, where our aim is to increase the quality of the Web Person search. Our study differs from the others in the way we utilize the web search results. In addition, we have used a different way to formulate queries.

3. APPROACH OVERVIEW

The main task of a WePS system is to accurately cluster the webpages in $D = \{d_1, d_2, \ldots, d_k\}$, such that each resulting cluster corresponds to a namesake. There are several possible ways of implementing a WePS engine, such as

```
PREPROCESSING(AllWebPages, m)
1  for each webpage d ∈ AllWebPages
2     P ← EXTRACT-PEOPLENE-SET(d)
3     P ← CLEAN-PEOPLENE-SET (P)
4     P_d ← PICK-M-PEOPLENES (P, d, m)
5     O ← EXTRACT-ORGNE-SET(d)
6     O ← CLEAN-ORGNE-SET (O)
7     O_d ← PICK-M-ORGNES(O, d, m)
8     TF_d ← PRECOMPUTE-TF/IDF-FACTORS(d, P, O)
9  TRAIN-CLASSIFICATION-SKYLINE(TrainingData)
```

Figure 1: Webpage Preprocessing.

client-side, third party proxy, or server-side approaches. In a third party proxy approach, the user query is issued first to a proxy, which in turn quires a web search engine and then clusters the returned results. The advantage of such an approach is that a third (independent) party could implement it. A client-side solution is similar, except for the software installed on the client acts as the proxy.

In a server-side approach, the user queried a web search engine directly. The server also does the clustering and returns the result to the user. The advantage of such an approach is that it is likely to be more efficient, as many webpage preprocessing steps can be done before any user query is issued. In addition, web querying is done internally instead of querying over the Internet. The disadvantage is that only a web search company could do that, or alternatively a third party should maintain a snapshot of the entire Web, e.g. using technology similar to that of WebBase developed at Stanford [10].

The solution proposed in this paper can potentially work with any of these architectures. However, currently it is more feasible as a server-side approach.

3.1 Preprocessing

The pseudo code in Figure 1 demonstrates the webpage preprocessing steps carried out on the server. They are performed in advance for the entire web collection, before the system starts accepting user queries. For each webpage, its Named Entities (NEs) are extracted and processed. The TF/IDF factors are precomputed. This is done to speed up the future computations of TF/IDF similarities between pairs of webpages. TF/IDF will be computed during actual query processing and will use these precomputed values. Finally, the classifier is trained on the training data via supervised learning. The latter is a key step which will be covered in detail in Section 4.3.

The pseudo code demonstrates that the extracted NEs are first cleaned. This is done using a set of filters. The idea is that NEs will be used by the framework as the context that identifies a namesake to some degree. However, certain types of NEs are too ambiguous for that purpose, in which case they are filtered out from further consideration. Given that the goal is to minimize the number of queries to the web search engine, the filters are specifically designed not to use any extra queries.

Filter 1. For instance, location names have been found to be too ambiguous to be used as context, as too many namesakes may be mentioned in the context of the same location. Therefore, the algorithm does not use location information as one of the social network components. Moreover, NE extractors sometimes wrongly extract the location names as organization names. This also creates the same problem mentioned above. Thus, to eliminate the ambiguity, the preprocessing step filters out the locations extracted as organizations. It does so by performing a lookup in a locally stored gazetteer with the organization name as the query. If there is a matching location in the gazetteer, then the organization name is simply filtered out.

```
PROCESS-QUERY(Q, k)
1  D ← GET-TOPK-WEBPAGES(Q, k)
2  for i ← 1 to k − 1 do
3     for j ← i + 1 to k do
4        if AlreadyMerged(d_i, d_j) = true then
5           break
6        s ← COMPUTE-TF/IDF(TF_{d_i}, TF_{d_j})
7        if s > τ then
8           MergedInOneCluster(d_i, d_j)

   // All pairs are "unprocessed" initially.
9  for each distinct unmerged unprocessed pair d_i, d_j ∈ D do
10    c_ij ← GET-WEB-COUNTS(d_i, d_j)
11    f_ij ← CONVERT-TO-FEATURES(c_ij)
12    if CLASSIFY-FEATURE(f_ij) = merge then
13       MergedInOneCluster(d_i, d_j)
14    MarkAsProcessed(d_i, d_j)

15 VISUALIZE-RESULTS-TO-USER(D)
```

Figure 2: Online User Query Processing.

Filter 2. The second filter deals with the NEs that consist of one-word person names, such as "John". Such NEs are highly ambiguous since they can appear in the context of many namesakes on the Web. Consequently, the algorithm prunes away the NEs consisting of one-word names. This filter works by performing a lookup into the dataset that store first names.

Filter 3. Similarly, the third filter handles NEs that are common English words. For example, word "defense" might be extracted from a webpage as an organization by the extraction software. However, it is a commonly used word, which can appear in the context of many namesakes. To detect common words the algorithm selects the most frequent 5000 terms from Wikipedia (http://www.wikipedia.org) as common English words. If an NE is a common English word, it is filtered out.

Filter 4. Suppose that we are disambiguating webpages for "Jack Smith". It is not rare to find out that two or more distinct "Jack Smith" namesakes are related to two distinct namesakes "John Smith". These two Jacks might for example be relatives of the two Johns. Thus, "John Smith" cannot serve as a good context to identify a particular "Jack Smith" namesake. To capture this intuition, the algorithm filters out people NEs whose last name is the same as the last name specified in the original query.

3.2 User Query Processing

A user query processing consists of three logical steps illustrated in pseudo code in Figure 2: (1) TF/IDF Clustering (Lines 1–8), (2) WebFeature Clustering (Lines 9–14), and (3) Visualizing the results to the user (Lines 15).

The first step of the algorithm merges all pairs of webpages whose TF/IDF similarities exceed a threshold. The threshold value is learned during the supervised learning process. TF/IDF is computed only on NEs extracted from the webpages, using the standard cosine similarity formula [24]. This initial clustering achieves two purposes: it de-

GET-WEB-COUNTS(d_i, d_j)
Let:
N be the queried name
$\mathcal{P}_{i1}, \mathcal{P}_{i2}, \ldots, \mathcal{P}_{im}$ be the people NEs extracted from d_i
$\mathcal{P}_{j1}, \mathcal{P}_{j2}, \ldots, \mathcal{P}_{jm}$ be the people NEs extracted from d_j
$\mathcal{O}_{i1}, \mathcal{O}_{i2}, \ldots, \mathcal{O}_{im}$ be the org. NEs extracted from d_i
$\mathcal{O}_{j1}, \mathcal{O}_{j2}, \ldots, \mathcal{O}_{jm}$ be the org. NEs extracted from d_j

1 $\mathcal{P}_i \leftarrow (\mathcal{P}_{i1}$ OR $\mathcal{P}_{i2}$ OR $\cdots$ OR $\mathcal{P}_{im})$
2 $\mathcal{P}_j \leftarrow (\mathcal{P}_{j1}$ OR $\mathcal{P}_{j2}$ OR $\cdots$ OR $\mathcal{P}_{jm})$
3 $\mathcal{O}_i \leftarrow (\mathcal{O}_{i1}$ OR $\mathcal{O}_{i2}$ OR $\cdots$ OR $\mathcal{O}_{im})$
4 $\mathcal{O}_j \leftarrow (\mathcal{O}_{j1}$ OR $\mathcal{O}_{j2}$ OR $\cdots$ OR $\mathcal{O}_{jm})$

5 $c_{ij1} \leftarrow$ GetWebCount(N AND $\mathcal{P}_i$ AND $\mathcal{P}_j$)
6 $c_{ij2} \leftarrow$ GetWebCount($\mathcal{P}_i$ AND $\mathcal{P}_j$)
7 $c_{ij3} \leftarrow$ GetWebCount(N AND $\mathcal{P}_i$ AND $\mathcal{O}_j$)
8 $c_{ij4} \leftarrow$ GetWebCount($\mathcal{P}_i$ AND $\mathcal{O}_j$)
9 $c_{ij5} \leftarrow$ GetWebCount(N AND $\mathcal{O}_i$ AND $\mathcal{P}_j$)
10 $c_{ij6} \leftarrow$ GetWebCount($\mathcal{O}_i$ AND $\mathcal{P}_j$)
11 $c_{ij7} \leftarrow$ GetWebCount(N AND $\mathcal{O}_i$ AND $\mathcal{O}_j$)
12 $c_{ij8} \leftarrow$ GetWebCount($\mathcal{O}_i$ AND $\mathcal{O}_j$)

Figure 3: Algorithm for Querying the Web.

creases the number of queries to the search engine [19], while at the same time it improves the quality of the final clustering, as will be discussed in Section 5.

The second step employs the Web to gain additional data about the interactions between the webpages in D. For each distinct unprocessed and unmerged pair of webpages $d_i, d_j \in D$, it utilizes the search engine to collect the co-occurrence information c_{ij} of the social networks of d_i and d_j. This will be explained in more detail in Section 4.1. The algorithm then transforms the co-occurrences into the corresponding similarity features (Section 4.2). It then uses the skyline-based classifier on those features to predict whether the d_i, d_j pair should be merged (Section 4.3). The algorithm continues such iterations until no unprocessed pairs are left to be considered. After the second step, the final clustering is ready. The third step presents the computed final clustering results to the user.

4. QUERY PROCESSING ALGORITHM

4.1 Queries to the Web Search Engine

Figure 2 demonstrates that for each unprocessed pair of webpages d_i and d_j the algorithm forms queries to the Web Search engine to compute co-occurrence statistics. This section explains the motivation behind using the queries as well as the procedure for forming such queries.

The purpose of using Web queries is to evaluate the degree of interaction of the social networks for two namesakes represented by webpages d_i and d_j. If there is evidence on the web that two social networks are closely related, then two webpages are merged into one cluster. The guiding principles in formulating the queries are:

- *Quality.* The queries should be chosen such that their results should allow creating a set of features that would enable high quality disambiguation.
- *Efficiency.* The overall number of the required queries should be minimized for efficiency reasons.

The pseudo code in Figure 3 illustrates the procedure for formulating the queries. It demonstrates that two major types of queries are utilized:

1. N AND $\mathcal{C}_i$ AND $\mathcal{C}_j$
2. $\mathcal{C}_i$ AND $\mathcal{C}_j$.

Here, $\mathcal{C}_i$ represents the context for d_i. It can be either the set of people NEs $\mathcal{P}_i$, or organization NEs $\mathcal{O}_i$. Context $\mathcal{C}_j$ is defined similarly for document d_j. Since $\mathcal{C}_i$ and $\mathcal{C}_j$ can have two possible assignments each, this creates 4 context combinations. Given that there are 2 types of queries, this leads to 8 queries in total, shown in Figure 3.

For example, assume that the user searches for the webpages related to "William Cohen". Suppose that the algorithm extracts 2 namesakes of each type per webpage, that is, $m = 2$. Assume that webpage d_i contains names "Jamie Callan" and "Tom Mitchell", and webpage d_j contains names "Andrew McCallum" and "Andrew Ng". Then the first query will be:

> "William Cohen" AND ("Jamie Callan" OR "Tom Mitchell") AND ("Andrew McCallum" OR "Andrew Ng").

The web search engine API has a function call that computes the number of webpages relevant to the query, without actually retrieving those webpages.

Observe that dataset D alone might not have any evidence to merge d_i and d_j. For instance, the TF/IDF similarity between d_i and d_j might be low. Also, among the webpages in D, names "Jamie Callan" and "Tom Mitchell" might be only mentioned in d_i, whereas "Andrew McCallum" and "Andrew Ng" only in d_j, and otherwise D might not contain any information revealing interactions among these people. However, querying the Web allows the algorithm to gain *additional information* to support the merge. In this case, the counts will be high enough to indicate that the people mentioned in the query are closely related.

4.2 Creating Features

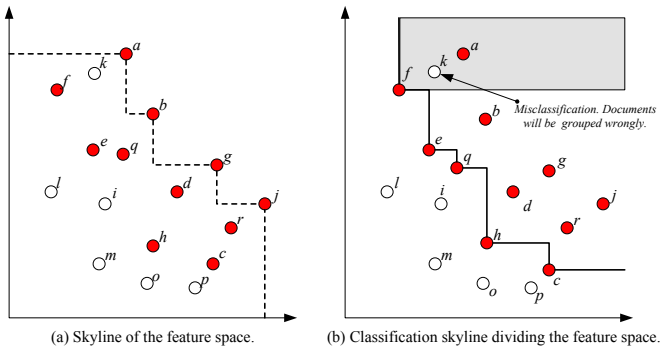
To estimate the degree of overlap of two contexts $\mathcal{C}_i$ and $\mathcal{C}_j$ for webpages d_i and d_j for the queried name N we can compute the co-occurrence count $|N \cdot \mathcal{C}_i \cdot \mathcal{C}_j|$ for query $N \cdot \mathcal{C}_i \cdot \mathcal{C}_j$. However, it might be difficult to interpret this absolute value without comparing it to certain other values. For instance, if this count value is high, does it mean the contexts overlap significantly and thus d_i and d_j should be merged? Or, is it simply because N is a common name and thus there are lots of webpages that contains it under many contexts? Or, is it because contexts $\mathcal{C}_i$ and $\mathcal{C}_j$ are too unspecific, and thus too many webpages contain them?

Similar questions have been studied in the past [7]. The solution proposed there advocates normalizing such values, based on either Jaccard or Dice similarities. The Jaccard similarity between two sets A and B computes the fraction of common elements in A and B among all the distinct elements:

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \quad (1)$$

The Dice similarity between two sets A and B computes the fraction of common elements in A and B among all the elements (including non distinct) in A and B, and normalizes it to $[0, 1]$ interval:

$$Dice(A, B) = \frac{2|A \cap B|}{|A| + |B|}. \quad (2)$$



(a) Skyline of the feature space. (b) Classification skyline dividing the feature space.

Figure 4: Example of a SkyLine.

There have been studies showing that the differences in retrieval quality, when using these measures, is insignificant and furthermore these measures are monotone with respect to each other [20].

The proposed algorithm uses the Dice similarity to get the normalized version of $|N \cdot \mathcal{C}_i \cdot \mathcal{C}_j|$ count. Two ways to normalize it have been examined:

$$Dice_1(N \cdot \mathcal{C}_i, N \cdot \mathcal{C}_j) = \frac{2|N \cdot \mathcal{C}_i \cdot \mathcal{C}_j|}{|N \cdot \mathcal{C}_i| + |N \cdot \mathcal{C}_j|},$$

and

$$Dice_2(N, \mathcal{C}_i \cdot \mathcal{C}_j) = \frac{2|N \cdot \mathcal{C}_i \cdot \mathcal{C}_j|}{|N| + |\mathcal{C}_i \cdot \mathcal{C}_j|}.$$

The algorithm employs the second formula, as it has proven to capture the ambiguity of context better. The following example provides just one type of scenario to illustrate the choice of the formula. Assume that the namesakes mentioned in two webpages d_i and d_j are different. Suppose that the extractor wrongly extracts "This" as the only person NE from d_i, and "That" as the only person NE from d_j. Assume that all (or, most of) the webpages of the two namesakes contain both "this" and "that". Then, $Dice_1$ similarity will be 1 (or, very high), causing the wrong merge of the webpages. However, $Dice_2$ similarity will be low, since $|\mathcal{C}_i \cdot \mathcal{C}_j|$ will be large. That is, $Dice_2$ will automatically capture that "this" and "that" is not a good choice to be used as the contexts.

This observation raises an interesting point. At first glance it might seem the algorithm employs two different strategies to achieve the same goal. That is, it detects ambiguous context by using the filtering strategy covered in Section 3.1. But, frequently the same can be achieved by simply using the $|\mathcal{C}_i \cdot \mathcal{C}_j|$ part of the Dice similarity. Notice, however, the Dice similarity can be low, for instance, because there is no evidence on the web of the context co-occurrence. But it also can be low because some of the context terms were too ambiguous. In the latter case, perhaps if some of the ambiguous terms are removed, there actually will be sufficient evidence to merge d_i and d_j. This is what precisely the filtering strategy attempts to do, by filtering out potentially ambiguous context terms upfront.

Hence, for each of the 4 possible $N \cdot \mathcal{C}_i \cdot \mathcal{C}_j$ combinations (i.e., P-P, P-O, O-P, O-O), the algorithm generates two features. The first one is the raw $|N \cdot \mathcal{C}_i \cdot \mathcal{C}_j|$ count. The second is its normalized version computed using the $Dice_2$ formula. Therefore, there are 8 features in total.

```
TRAIN-CLASSIFICATION-SKYLINE(D)
S_pool // Skyline 1: a pool of points to choose from
S_clas // Skyline 2: current classification skyline
S_best // Skyline 3: best classification skyline observed
1  S_clas ← {(∞, ∞, ∞, ∞, ∞, ∞, ∞, ∞)}
2  S_best ← S_clas
3  S_pool ← COMPUTE-SKYLINE(P+_actv)
4  while S_pool ≠ ∅ do
5     P_lsterr ← GET-BESTNEGQUAL-POINTSET(S_pool, S_clas, P_actv)
6     p_best ← GET-BESTQUAL-POINT(P_lsterr, S_clas, P_actv)
7     S_clas ← UPDATE-SKYLINE(S_clas, p_best)
8     if QUAL(S_class) > QUAL(S_best)
9        S_best ← S_class
10    P_actv ← UPDATE-ACTIVE-POINTSET(P_actv, p_best)
11    S_pool ← UPDATE-POOL-POINTSET(S_pool, p_best)
12 return S_best
```

Figure 5: Algorithm for training.

```
GET-BESTNEGQUAL-POINTSET(S_pool, S_clas, P_actv)
1  Q_best ← 0 // best quality observed
2  P_best ← ∅ // set of best points
3  for each p ∈ S_pool
4     S ← UPDATE-SKYLINE(S_clas, p)
5     Q ← QUAL(S, P−_actv)
6     if Q > Q_best then
7        P_best ← {p}
8        Q_best ← Q
9     else if Q = Q_best then
10       P_best ← P_best ∪ {p}
11 return P_best
```

Figure 6: GET-BESTNEGQUAL-POINTSET.

4.3 SkyLine-Based Classification

Observe that the features are chosen such that there is *dominance* in data in terms of merge decisions. First, let us make a few auxiliary definitions. We will say point $\mathbf{f} = (f_1, f_2, \ldots, f_8)$ *up-dominates* point $\mathbf{g} = (g_1, g_2, \ldots, g_8)$, if $f_1 \leq g_1, f_2 \leq g_2, \ldots, f_8 \leq g_8$, and will denote it as $\mathbf{f} \leq \mathbf{g}$. Similarly, we will say $\mathbf{f}$ *down-dominates* $\mathbf{g}$, if $f_1 \geq g_1, f_2 \geq g_2, \ldots, f_8 \geq g_8$, and will denote it as $\mathbf{f} \geq \mathbf{g}$. Similarly, there is a notion of *strict* domination where '$\leq$' and '$\geq$' are substituted with '$<$' and '$>$'.

Assume that d_i, d_j pair is characterized by the feature vector $\mathbf{f} = (f_1, f_2, \ldots, f_8)$. Suppose that based on $\mathbf{f}$ the algorithm decides that d_i and d_j should be merged. Assume that there is another pair of webpages d_k and d_ℓ, which is characterized by vector $\mathbf{g} = (g_1, g_2, \ldots, g_8)$. Then, if $\mathbf{f} \leq \mathbf{g}$, then d_k and d_ℓ should also be merged, since their social networks contain even more evidence that the two namesakes are the same person. Thus, there is dominance in feature data in terms of merge decisions.

The proposed algorithm uses a well-studied notion of SkyLine for classifying points as "merge" or "do not merge". A skyline of a set of points P is the subset S of all points from P such that each point $p \in S$ is not strictly dominated by any other point from P. Figure 4(a) illustrates an example of a down-dominating skyline consisting of points $\{a, b, g, j\}$.

Figure 4(b) plots the up-dominating skyline for all the merge ('+') points, plotted as filled circles. The do-not-merge ('−') points are plotted as empty circles in that figure. A **classification skyline** is an up-dominating skyline on all the points the classifier declares to be 'merge' points. Given a classification skyline, the classifier will classify any point this skyline up-dominates (i.e., any point that is 'on'

GET-BESTQUAL-POINT($P_{lsterr}, S_{clas}, P_{actv}$)
1 $Q_{best} \leftarrow 0$ // *best quality observed*
2 p_{best} // *best point observed*
3 **for each** $p \in P_{lsterr}$
4 $S \leftarrow$ UPDATE-SKYLINE(S_{clas}, p)
5 $Q \leftarrow$ QUAL(S, P_{actv})
6 **if** $Q > Q_{best}$ **then**
7 $p_{best} \leftarrow p$
8 $Q_{best} \leftarrow Q$
9 **return** p_{best}

Figure 7: GET-BESTQUAL-POINT.

UPDATE-SKYLINE(S_{clas}, p)
1 $S \leftarrow S_{clas}$
2 remove from S all points dominated by p // *use indexing*
3 $S \leftarrow S \cup \{p\}$
4 **return** S

Figure 8: UPDATE-SKYLINE.

or 'above' the skyline) as a 'merge' point, and any other point – as a do-not-merge point. Figure 4(b) illustrates one possible classification skyline for the plotted dataset. As any classifier, a skyline-based classifier can make a mistake. The figure shows that this choice of skyline will cause a do-not-merge point k to be wrongly classified as a merge point.

4.4 Training Classification SkyLine

The clustering algorithm works by merging pairs of webpages whose co-occurrence features are above the classification skyline. This section covers a greedy algorithm for learning such a skyline, illustrated in Figure 5.

The training algorithm is given a training dataset wherein each point that should be merged is labeled with '+', and each point that should not – with '−'. It is also given a quality metric, such as the pairwise F measure or B-cubed, so that at any point in time it can measure which quality it can get by making certain types of decisions.

S_{clas} initially consists of one point $(\infty, \infty, \ldots, \infty)$, which causes no points to be classified as '+'. At each iteration, the algorithm tries to greedily locate the best point to add to S_{clas}, to make it better. For that it examines all the points from a certain pool of '+' points, S_{pool}. Among them it first select the subset of points P_{lsterr} such that adding a point $p \in P_{lsterr}$ to S_{clas} would cause the least error, see Figure 6. It does so by examining the quality of the clustering when using S_{clas} and using $p \in S_{pool}$ to classify only the '−' points. Then among the points in P_{lsterr} it selects a point p_{best} adding which to S_{clas} would lead to the best overall quality, see Figure 7. Point p_{best} is then added to the classification skyline S_{clas}. It keeps track of the best skyline observed so far by storing and updating it in S_{best}.

Adding point p_{best} to the classification skyline S_{clas} can make several (at least one) yet-unclassified points to be classified as '+' by the algorithm. That can happen for two reasons:

- *Direct Merges.* The points dominated by p_{best} will be classified as '+'.
- *Transitive Merges.* Classifying p_{best} as 'merge' can cause multiple clusters to merge, triggering the merges of the elements of those clusters due to the transitivity. This corresponds to classifying the corresponding points as '+'.

UPDATE-ACTIVE-POINTSET(P_{actv}, p)
1 Use indexing to remove all points in P_{actv} dominated by p.
//*Remove points in the region with diagonal* $p - (1, 1, \ldots, 1)$.
2 **return** P_{actv}

Figure 9: UPDATE-ACTIVE-POINTSET.

UPDATE-POOL-POINTSET(S_{pool}, p)
1 $S_{pool} \leftarrow S_{pool} \setminus \{p\}$
2 Use indexing to find points P from P^+_{act} (if any) that now should be in S_{pool} due to removal of p. This procedure incrementally maintains the skyline of P^+_{act} in S_{pool}.
3 $S_{pool} \leftarrow S_{pool} \cup P$
4 **return** S_{pool}

Figure 10: UPDATE-POOL-POINTSET .

The algorithm maintains in P_{actv} the set of yet-unclassified points. We will use notation P^+_{actv} and P^-_{actv} to denote the '+' and '−' points in this set. The pool of points S_{pool} from which P_{lsterr} is constructed is the skyline of P^+_{actv}. The choice of S_{pool} is motivated by several factors. First, for the efficiency reasons we want to avoid using large sets, like for instance P_{actv}. The skyline tends to be much smaller than P_{actv}. Second, since it is a set of +'s, this guarantees making at least one correct merge decision. Third, if we look to add a '+' point to S_{clas} that would minimize the number of misclassifications of −'s (false positives), then that skyline will contain all such points. Figures 6, 7, 8, 9, and 10 demonstrate the steps of the learning algorithm in more detail. The feature space is indexed for efficient processing, such as incremental maintenance of the skylines instead of rebuilding them from scratch.

The example in Figure 11 illustrates one iteration of the algorithm. Assume that the current classification skyline is $S_{clas} = \{q, d, c\}$, as shown in Figure 11(a). Then S_{pool} will be the down-dominating skyline on yet-unclassified '+' points, which is $S_{pool} = \{f, e, u, h\}$. Notice that among yet-unclassified '−' points (the ones below the classification skyline) f dominates 1 point (k), e – 2 points, u – 1 point, and h – 2 points. If the goal of our quality measure to minimize the number false positives, then P_{lsterr} will be chosen as $P_{lsterr} = \{f, u\}$ since they introduce only 1 new error. Now, given a choice of adding f or u to the skyline S_{clas}, assume that adding f would lead to the best classification quality. Then the algorithm will choose $p_{best} = f$. Figure 11(b) demonstrates what happens after adding f to the skyline.

Discussion. Observe that one of the solutions would be just to use any classifier, such as SVM, to classify the points as +'s and −'s. Unlike that solution, the skyline-based classier utilizes several additional factors to its advantage. First, it explicitly takes into account the dominance that exists in data. Second, it is aware that there is a clustering underneath the +'s and −'s and that classifying a point as a + can cause several other points be classified as + due to transitivity. Third, it greedily fine-tunes itself to a given quality metric; whereas the first approach is more geared toward specifically pairwise F-measure. As we shall see in the experimental section, these qualities allow the skyline-based solution to outperform SVM/DTC based classifiers.

5. EXPERIMENTAL RESULTS

In this section we empirically evaluate our approach and compare it with some previously proposed algorithms.

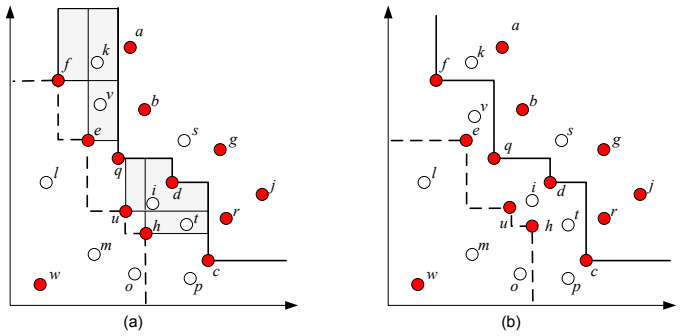


Figure 11: Skyline learning steps.

Datasets. We tested our algorithms using three different publicly available datasets. The first one, `WWW05`, which is used in [5] contains 12 different people. The second one, `SIGIR05`, which is used in [3] is composed of 9 different people. This dataset was also a trial dataset for [2]. The third data set `WePS` is the dataset that is used in the WEPS task [2]. This dataset is composed of training and testing datasets. In the training there are 49 different people, whereas in testing dataset there are 30 people. On average, each person dataset has 100 web pages.

We first combined both data sets `WWW05` and `SIGIR05` to have more training examples with different aspects. Dataset `SIGIR05` contains web pages for very common person names, which are formed using the census data. Dataset `WWW05` is composed of the people who are related to each other due to their research areas. Most of the people in this dataset are well represented on the Web. The names in `SIGIR05` are very common English names, as a result each person is represented with 1 or 2 pages, on the other hand dataset `WWW05` is more diverse. For example, there are two sub datasets, "Adam Cheyer" and "Leslie Kaelbling" in `WWW05`, where pages represent only two people with the same name. Another sub dataset, "William Cohen", contains a popular person with a very common name, majority of the pages are about the secretary of defense William Cohen. We then applied leave-one-out cross validation to demonstrate the effectiveness of our algorithm. We learned the skyline point with the 20 different names and tested the accuracy of those skyline points on the left-out dataset. We used Yahoo! API to collect the co-occurrence statistics.

Quality Measures. We use two measures that are commonly used for assessing the quality of WePS applications: F_P and F_B [2,3]. Here, F_P is the harmonic mean of *Purity*, *Inverse Purity*. F_B is the B-cubed F-measure [4].

Baseline Methods. We use two methods as baseline: the Named Entity based Clustering (NE) and Web based co-occurrence similarity measure (WebDice). NE is a TF/IDF merging scheme that uses only Named Entities from the webpages to compute the TF/IDF similarity values, which are compared against a threshold. This algorithm was the second runner-up in the WEPS competition. The second baseline, WebDice computes the similarity between two pages by summing up the individual dice similarities of People-People, People-Organization similarities of web pages. Single-link clustering approach is used and the optimal threshold is selected by applying leave-one-out cross validation. The threshold for the NE-clustering is learned to be between 0.1 and 0.2 on the `WWW05`'s cross-validation experiments. On the other hand the threshold for the WebDice clustering is 0.001 for `WWW05`. On the `WePS` dataset the threshold is learned to be 0.002 and 0.2 for WebDice Clustering and NE-Clustering respectively.

	SkyLine	WebDice	NE
Name	$\mathbf{F_{P/B}}$	$\mathbf{F_{P/B}}$	$\mathbf{F_{P/B}}$
Adam Cheyer	92.8/85.8	**95.7/91.5**	86.5/73.9
Andrew McCallum	95.0/89.5	**95.6/93.1**	92.5/88.2
Andrew Ng	88.4/84.1	**89.1/84.2**	82.4/74.6
Ann Hill	**91.2/89.6**	76.3/73.2	87.8/85.0
Bill Mark	**89.2/84.2**	86.4/81.4	77.7/69.7
Brenda Clark	96.6/94.4	**97.2/95.5**	93.1/89.7
Christine King	88.1/82.8	**90.2/86.2**	86.9/91.0
David Israel	**86.9/80.7**	79.5/72.4	80.3/73.3
David Mulford	**85.9/78.8**	83.9/74.9	77.6/65.9
Fernando Pereira	82.3/71.9	**86.3/77.9**	83.3/73.5
Helen Miller	**92.1/91.0**	88.7/87.2	90.7/88.5
Leslie Kaelbling	97.7/95.4	**98.3/96.6**	86.6/74.0
Lisa Harris	83.7/79.6	**84.5/79.8**	84.4/79.8
Lynn Voss	**84.7/79.5**	72.0/61.6	66.3/60.5
Mary Johnson	**89.0/88.8**	83.5/83.2	79.1/77.3
Nancy Thompson	**93.9/ 91.8**	87.0/80.8	77.7/74.5
Samuel Baker	**89.8/88.6**	71.9/67.0	69.5/69.0
Sarah Wilson	**90.3/87.6**	84.8/81.2	71.9/68.7
Steve Hardt	76.6/61.0	**84.0/72.1**	43.9/30.6
Tom Mitchell	89.1/86.2	**89.8/87.5**	86.9/83.5
William Cohen	87.9/79.2	**89.9/82.9**	86.1/76.7
Mean	**89.1/84.4**	86.4/81.4	80.5/74.2

Table 1: Results

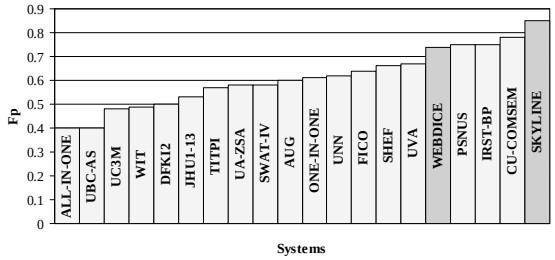


Figure 12: Results on the WEPS Test data.

Quality on WePS dataset. We used the training data of `WePS` to learn the skyline points and tested it on its test portion, as required by the WEPS task. Figure 12 compares the quality of our algorithm with the quality of the systems participated to the WEPS task. The figure shows that the proposed approach outperforms the best solution by 7%.

Detailed Clustering Results Table 1 compare the results of the proposed algorithms and of the baselines solutions. We used paired t-test to measure the statistical significance of the improvement. As Table 1 shows improvement to the baseline methods is statistically significant.[1]

We performed another set of experiments with the `WePS`. First we compute the cross-validation results on the training dataset, and after that we used the training and testing algorithms to test the effectiveness of our algorithm on the

[1]The improvement of both skyline and dice similarity based algorithms with respect to NE based clustering method is statistically significant for $\alpha = 0.01$. The improvement of skyline algorithm with respect to dice similarity is statistically significant for $\alpha = 0.05$.

	SkyLine	WebDice	NE
Name	$\mathbf{F_{P/B}}$	$\mathbf{F_{P/B}}$	$\mathbf{F_{P/B}}$
train	**85.7/78.8**	85.5/78.5	79.4/71.5
test	**84.8/78.7**	73.5/63.2	78.0/70.2

Table 2: Results on WEPS

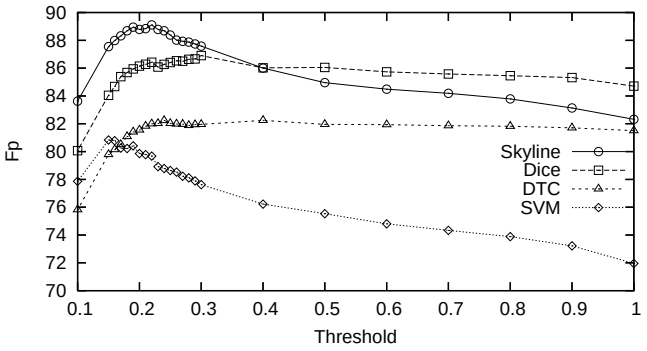


Figure 13: Initial Clustering effect on final Clustering.

WEPS dataset. Table 2 shows the cross validation results of WEPS training dataset. The improvement to the baseline algorithms is also statistically significant. In the table we also show the clustering quality results for the test dataset. It improves the quality as well. The threshold for initial clustering is empirically estimated on the training data and then used on the testing. The threshold is chosen as 0.2 for all three datasets.

Effect of Initial Clustering. In this experiment, we demonstrate the effect of the threshold chosen for the initial TF/IDF clustering to the overall quality. While this threshold is learned from data in the previous experiments, in this experiment it is varied to analyze the dependence. Figure 13 shows the quality change as the threshold increases. As the threshold approaches 0.2 the skyline based algorithm significantly outperforms the dice similarity approach. We also compare the clustering quality of our algorithm with a Decision Tree based Classifier (DTC) and with SVM. SkyLine based method outperforms them both, since it takes into account the dominance in data and tunes itself to F_P.

Efficiency. The proposed approach is likely to be less efficient than any of the 16 WePS task approaches, if implemented as a third party proxy solution. The reason is that it requires on average 4360×8 web search queries to the API per person name. Many search engines do not allow that many queries per day. Notice, however, the number of queries is relatively small, and thus a server-side solution is a real possibility.

6. CONCLUSIONS AND FUTURE WORK

This paper proposes a Web People Search approach that is based on collecting the co-occurrence information from the Web in order to get a better disambiguation quality. A skyline based supervised learning methodology has been developed. The proposed methodology takes into account the dominance that exits in the feature space as well as greedily fine-tunes itself to a given quality measure. These qualities allow it to outperform other state of the art WePS solutions. The limitation of the proposed solution is that as of currently it is more feasible a server-side approach due to the present day restrictions of web search engine querying APIs.

7. REFERENCES

[1] R. Al-Kamha and D. W. Embley. Grouping search-engine returned citations for person-name queries. In *WIDM04*, 2004.

[2] J. Artiles, J. Gonzalo, and S. Sekine. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *SemEval*, 2007.

[3] J. Artiles, J. Gonzalo, and F. Verdejo. A testbed for people searching strategies in the WWW. In *SIGIR*, 2005.

[4] A. Bagga and B. Baldwin. Entity-based cross-document co-referencing using the vector space model. In *COLING*, 1998.

[5] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *WWW*, 2005.

[6] D. Bollegala, Y. Matsuo, and M. Ishizuka. Extracting key phrases to disambiguate personal names on the web. In *CICLing*, 2006.

[7] D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *WWW'07*, 2007.

[8] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Comput. Netw. ISDN Syst.*, 29(8-13):1157–1166, 1997.

[9] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. An adaptive graphical approach to entity resolution. In *Proc. of ACM IEEE Joint Conference on Digital Libraries (ACM IEEE JCDL)*, June 17-23 2007.

[10] J. Cho, H. Garcia-Molina, T. Haveliwala, W. Lam, A. Paepcke, S. Raghavan, and G. Wesley. Stanford webbase components and applications. In *ACM TOIT*, 2006.

[11] E. Elmacioglu, M.-Y. Kan, D. Lee, and Y. Zhang. Web based linkage. In *WIDM07*, 2007.

[12] E. Elmacioglu, Y. F. Tan, S. Yan, M.-Y. Kan, and D. Lee. Psnus: Web people name disambiguation by simple clustering with rich features. In *SemEval*, 2007.

[13] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, 2007.

[14] R. Guha and A. Garg. Disambiguating people in search. In *Stanford University*, 2004.

[15] D. V. Kalashnikov, Z. Chen, R. Nuray-Turan, and S. Mehrotra. Web people search via connection analysis. *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE),to appear.*

[16] 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.

[17] 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.

[18] D. V. Kalashnikov, S. Mehrotra, Z. Chen, R. Nuray-Turan, and N. Ashish. Entity search: A new paradigm for people search on the web. In *IEEE ICDE*, Istanbul, Turkey, April 16–20 2007.

[19] P. Kanani, A. McCallum, and C. Pal. Improving author coreference by resource-bounded information gathering from the web. In *IJCAI*, 2007.

[20] I. C. Lerman. *Les Bases de la Classification Automatique.* Gauthier-Villars, 1970.

[21] X. Li, P. Morie, and D. Roth. Semantic integration in text: From ambiguous names to identifiable entities. *AI Magazine. Special Issue on Semantic Integration*, pages 45–68, 2005.

[22] Y. Matsuo, J. Mori, M. Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka. Polyphonet: an advanced social network extraction system from the web. In *WWW*, 2006.

[23] R. Nuray-Turan, D. V. Kalashnikov, and S. Mehrotra. Self-tuning in graph-based reference disambiguation. In *Proc. of Int'l Conf. on Database Systems for Advanced Applications (DASFAA)*, April 9-12 2007.

[24] C. J. van Rijsbergen. *Information Retrieval.* Butterworth, 1979.

[25] X. Wan, J. Gao, M. Li, and B. Ding. Person resolution in person search results: Webhawk. In *CIKM*, 2005.