Web Based Linkage

Ergin Elmacioglu Min-Yen Kan† Dongwon Lee* Yi Zhang‡
The Pennsylvania State University, PA, USA †National University of Singapore, Singapore
‡University of California, Santa Cruz, CA, USA
{ergin,dongwon}@psu.edu, kanmy@comp.nus.edu.sg, yiz@soe.ucsc.edu

ABSTRACT

When a variety of names are used for the same real-world entity, the problem of detecting all such variants has been known as the *(record) linkage* or *entity resolution* problem. In this paper, toward this problem, we propose a novel approach that uses the Web as the collective knowledge source in addition to contents of entities. Our hypothesis is that if an entity e_1 is a duplicate of another entity e_2, and if e_1 frequently appears together with information I on the Web, then e_2 may appear frequently with I on the Web. By using search engines, we analyze the frequency, URLs, or contents of the returned web pages to capture the information I of an entity. Extensive experiments verify that our hypothesis holds in many real settings, and the idea of using the Web as the additional source for the linkage problem is promising. Our proposal shows 51% (on average) and 193% (at best) improvement in precision/recall compared to a baseline approach.

Categories and Subject Descriptors

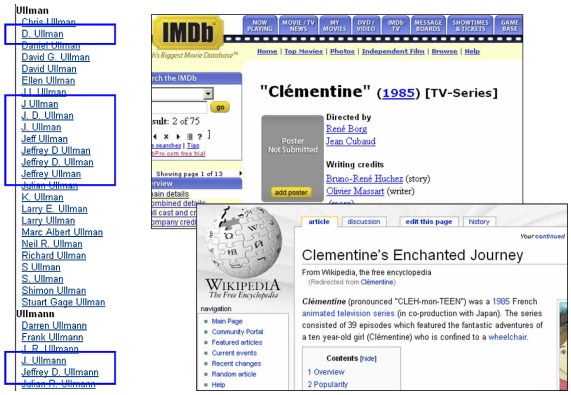
H.3.3 [**Information Search and Retrieval**]: Selection process; H.3.7 [**Information Search and Retrieval**]: Digital Libraries; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Design, Experimentation

1. INTRODUCTION

Large-scale data repositories often suffer from duplicate entities that have different identifiers, but refer to the same real-world object. Part of the problems often stem from the fact that the chosen identifier is not unique (i.e., bad choice of primary key in database jargon). A good example of this scenario is the use of *names* as identifiers. Names of people, movies, or organizations are frequently used as unique identifiers for its simplicity and intuitive association. However, due to various reasons, a variety of names may be used to refer to the same entity.



(a) ACM (b) IMDB & Wikipedia

Figure 1: Duplicate names: (a) ten duplicate entities of the author "J. D. Ullman" in ACM digital library, and (b) two duplicate titles of the same movie in IMDB and Wikipedia.

Example 1. In the ACM digital library, the full name of an author is used as an identifier (Figure 1(a)) so that in order to access citations of the author "J. D. Ullman", one has to look for his name. However, since the ACM digital library was constructed by software that extracts metadata (e.g., author, title) from articles, it tends to have errors. Therefore citations of the author "J. D. Ullman" are scattered under ten duplicate names in the ACM digital library, causing a confusion. Similarly, Figure 1(b) shows the example where two repositories use different titles (i.e., names) of the same movie "Clementine". Such examples of duplicate entities on the Web are common, too. For instance, suppose a person "John Doe" moves from an organization x to y and have different home pages at each organization, one using "John Doe" and the other using "J. Doe". Then, being able to identify if two home pages are for the same person or not is an interesting problem. □

Let us denote that, among the duplicate entities, the single authoritative one as *canonical entity* while the rest as *duplicate entities* or *duplicates* in short. Such a problem is, in general, known as the **(record) linkage** or *entity*

*Contact author. His research was partially supported by IBM Eclipse Innovation Award (2004, 2006) and Microsoft SciData Award (2005).

WIDM'07, November 9, 2007, Lisboa, Portugal.

resolution (ER) problem. The linkage problem frequently occurs in many applications and is exacerbated especially when data are integrated from heterogeneous sources. Note that in this paper our focus is on how to detect duplicate entities with similar names. Another important problem that can arise is the confusion due to homonyms – if two people have the same name spelling "John Doe", how to distinguish them? This problem is often referred to as the *name disambiguation* problem and has been extensively studied (e.g., [16, 3]). Since two problems are different in nature, we do not consider the name disambiguation problem any further in this paper.

With an array of extensive research on the linkage problem (to be surveyed in Section 5), in general, there are various efficient and effective methods to identify duplicate entities. By and large, previous approaches to the linkage problem (e.g., [6, 17, 9]) work as follows: (1) the information (e.g., name, metadata, or contents) of an entity, e, is captured in a data structure, $D(e)$, such as a multi-attribute tuple or an entropy vector; (2) a distance function that takes two inputs, f, is prepared; (3) the distance of two entities, e_1 and e_2, is measured as that of the corresponding data structures, $D(e_1)$ and $D(e_2)$, using the function f: $dist(e_1, e_2) = f(D(e_1), D(e_2))$; and (4) finally, if the result, $r = dist(e_1, e_2)$, is less than certain threshold, ϕ, then the two entities are deemed to be duplicates: $r < \phi \rightarrow e_1 \sim e_2$. Although this works well in many scenarios, this approach often suffers when the entity e in question does not have enough information or has only poor information for linkage.

How can one determine then if two entities are the same or not if the data to compare are not sufficient to decide? One solution is to ask people what they think. If many people collectively agree that two entities are the same, then two entities are probably the same. We argue that the Web is a good representation of what people think. That is, we propose to seek for additional information of an entity from the Web. Our hypothesis is the following:

Hypothesis 1. *If an entity e_1 is a duplicate of another entity e_2, and if e_1 frequently appears together with information I on the Web, then e_2 may appear frequently with I on the Web, too.* ■

Therefore, by measuring how the information I appears together with e_2 on the Web, we can determine if e_2 is a duplicate of e_1 or not. In particular, we propose various methods to capture the information I of an entity from the Web. Since our methods rely on search engines such as Google or MSN to draw additional information of an entity, compared to a baseline method, it is our hope that ours be simpler and more practical.

The contribution of our work is: (1) we propose the idea of using the *Web* as a source for additional knowledge of an entity for the linkage problem, (2) to capture the additional information of an entity returned from a search engine, we propose various methods such as using the frequency of representative terms, URL information or the content of returned web pages, and (3) we empirically validate our hypothesis with extensive experiments.

2. OVERVIEW OF THE PROBLEM

To make the presentation simple, let us assume that our problem setting has a single canonical entity e_c and the goal is to find all duplicate entities of e_c. Formally, the linkage problem in our setting is defined as follows:

> Given a set of entities, E, where each entity e_i has a name and information $I(e_i)$ (e.g., contents, affiliation, social network knowledge), for each canonical entity, e_c ($\in E$), identify all duplicate entities, e_v ($\in E$) by consulting the collective knowledge of the Web, such that $sim(I(e_c), I(e_v)) > \phi$, where ϕ is a pre-set threshold.

A naive algorithm with $O(|E|^2)$ running time for the general linkage problem is:

```
for each entity e_c ∈ E:
  for each entity e_v ∈ E, where e_c ≠ e_v:
    if sim(I(e_c), I(e_v)) > φ, then e_v ≈ e_c
```

Note that what we study in this paper is *not* the algorithmic improvement of the general linkage problem. Rather, we study how to devise better similarity function, $sim()$, utilizing additional information I from the Web. Throughout the paper, we use the notation in Table 1.

Table 1: Notations used.

Notation	Description
e_1, e_2, ..	an entity
e_c	a canonical entity
E	an entity set
t_1, t_2 ..	token
tf	term frequency
tf*idf	term freq.* inverse document freq.
w_tf*idf	web information based $tf * idf$
$name(e_i)$	name of entity e_i
$token(e_i)$	all tokens of entity e_i's content
$data(e_i)$	representative data piece of entity e_i's content
$count(x)$	number of pages returned by query string "x"
$host_k(x)$	top-k host names returned by query string "x"
$x \cap y$	equivalent to "x `AND` y" in a query string
$x \cup y$	equivalent to "x `OR` y" in a query string

3. WEB BASED LINKAGE

The basic framework of our approach is as follows. Consider a canonical entity e_c and a candidate entity e_1 that is potentially a duplicate of e_c. Name descriptions of entities are available as $name(e_c)$ and $name(e_1)$. Then,

1. We identify the information I as the best representative data piece of e_c. This data piece could be anything – from as simple as a single token of e_c to tuple(s) or as long as the entire contents of e_c. Suppose we get such a data piece from e_c, called $data(e_c)$.

2. We acquire a collective knowledge of people to see how "often" $data(e_c)$ is used together with $name(e_1)$. If e_1 is a duplicate of e_c, it should have a frequent appearance with $data(e_c)$ on the Web. To check this, we submit two queries to a search engine and collect the results.

3. By analyzing the returned pages in various aspects (e.g., frequency, URLs, documents), then we measure how similar e_1 is to e_c.

3.1 Representative Data of an Entity

Suppose one wants to determine if an author "John McCarthy" is a duplicate entity of an author "McCarthy, N. J.". Depending on the context, this task may not be easy to resolve. However, if one knows that the representative data of "John McCarthy" is, say, "LISP" or "Time Sharing", then one may measure how often "McCarthy, N. J." is used together with "LISP" or "Time Sharing" on the Web, and conclude that "McCarthy, N. J." is not a duplicate of "John McCarthy". Therefore, the first step of our framework is to identify the representative data of an entity e, denoted as $data(e)$.

Ideally, $data(e)$ helps discriminate duplicates of e. If $data(e)$ itself occurs too often on the Web, then it does not help link duplicates. On the other hand, if $data(e)$ itself occurs too rarely on the Web, then its discriminating power might be too weak to be used, and can be easily influenced by noise. We use a single token for $data(e)$ in our experiments.

By adopting conventional approaches in information retrieval, we propose to use three measures to select the best representative token of an entity as follows:

- *tf*: One may assume that the more frequently a token appears in an entity, the more representative the token gets. Therefore, we may use the traditional *term frequency (tf)* as a metric[1], and use the token that occurs most frequently as the representation.
- *tf*idf*: When the token selected by *tf* too common like "computer" or "web", its discriminating power diminishes. Even if we get relevant web pages for an entity e, such a common token may also frequently appear with many other candidate entities, and thus may not help find the actual duplicates. Hence the popular *term frequency * inverse document frequency (tf*idf)* [29] scheme is more appealing since it can find more discriminative tokens in an entity's data contents[2].
- *w_tf*idf*: Finally, we propose a web based scheme called *w_tf*idf*, which resembles *tf*idf* but gets the frequency information from the Web. It is defined as:

$$w\_tf * idf(e, t_i) = \frac{count(name(e) \cap t_i)}{count(t_i)}$$

where e is an entity, t_i is a token from e's contents, and $count(x)$ is a function returning the number of occurrences of the term x on the Web. That is, *w_tf*idf* is based on the observation that the more t_i appears with e and the less t_i independently appears on the Web, the more valuable t_i gets to discriminate e.

Our experiments in section 4 show that using a single token, if selected carefully, could be very effective to uniquely discriminate the appearance of an entity name on the web. There are abundant possibilities for utilizing more than one token towards a better solution for the problem. We also show the effect of using multiple tokens in our current framework in section 4.

[1]From the corpora (i.e., contents or metadata of entities), we first remove all stop-words (e.g., "a" or "the").

[2]Even further, probabilistic topic models such as LDA [7] can be used on the content to discover a pre-specified number of latent topics, each of which has a particular probability of emitting a specific token. Then one or more of such topics could be utilized as a representative data for the entity in question. We leave this exploration as future work.

3.2 Knowledge Acquisition from the Web

Once one representative token t_c is selected for a canonical entity e_c, suppose we want to determine if another candidate entity e_i is a duplicate entity of e_c or not. Then, we form two queries Q_1 and Q_2 to submit to a search engine:

Q_1: "*name(e_c)* `AND` t_c" and Q_2: "*name(e_i)* `AND` t_c"

In return, we receive two separate search engine result pages (for Q_1 and Q_2) of relevant web pages that contain both entity names and the representative token in question.

3.3 Interpreting the Collective Knowledge

We propose three methods to analyze the returned results to unearth the collective knowledge of the Web.

(1) ***Page Count***. Search engines return the number of web pages that satisfy the constraints of the query. Suppose an entity e_2 is a duplicate of an entity e_1, but e_3 is not. Further, assume that a token t_i is selected as the representative of e_1. Then it is plausible that the appearance of $e_1 \cap t_i$ and $e_2 \cap t_i$ on the Web will be *relatively* similar than that of $e_1 \cap t_i$ and $e_3 \cap t_i$. That is, $|count(e_1 \cap t_i) - count(e_2 \cap t_i)| \ll |count(e_1 \cap t_i) - count(e_3 \cap t_i)|$. Formally, $sim(e_c, e_i) = 1$ if $count(name(e_c) \cap t_c) = count(name(e_i) \cap t_c)$, and otherwise $sim(e_c, e_i) = \frac{1}{|count(name(e_c) \cap t_c) - count(name(e_i) \cap t_c)|}$, where t_c is the best representative token of e_c.

(2) ***URL***. From top-k results returned from search engines, we extract host addresses of the URLs, and calculate the Jaccard similarity as the similarity:

$$sim(e_c, e_i) = \frac{|host_k(name(e_c) \cap t_c) \bigcap host_k(name(e_i) \cap t_c)|}{|host_k(name(e_c) \cap t_c) \bigcup host_k(name(e_i) \cap t_c)|}$$

where $host_k(x)$ is the collection of host names of top-k URLs returned for the search query x.

(3) ***Web Pages***. If one views the results of Q_1 and Q_2 as *virtual documents*, then their similarity can be computed using document to document similarity measures. Then, depending on the virtual document is created among a set of returned web pages, a variety of alternatives can be used. Table 2 lists a set of heuristics that we used to create a virtual document. Suppose we created two virtual documents, D_i and D_j, using one of approaches in Table 1. Then, we propose to use three methods to measure the similarity of D_i and D_j – Jaccard, cosine, and language models, as follows.

First, *Jaccard similarity* of D_i and D_j is the ratio between intersected vs. unioned token sets of two documents: $sim_{jaccard}(D_i, D_j) = \frac{|tokens(D_i) \bigcap tokens(D_j)|}{|tokens(D_i) \bigcup tokens(D_j)|}$. Second, *cosine similarity* of D_i and D_j uses the cosine of the angle between two m-dimensional vectors where m is the number of distinct tokens in corpora) as the similarity: $sim_{cosine}(D_i, D_j) = \frac{\sum_{k=1}^{n} token_k(D_i) token_k(D_j)}{\|D_i\| \|D_j\|}$. Finally, *probabilistic language models* have been successfully applied to the ad-hoc IR tasks such as [35]. In the language model approach, each document is represented by a unigram word distribution θ_d. Then, the *Kullback-Leibler divergence* can be used to measure how the language models of two documents differ: $KL(\theta_i \| \theta_j) = \sum_k p(k|\theta_i) log \frac{p(k|\theta_i)}{p(k|\theta_j)}$. Here, θ_d can be found by the *maximum likelihood estimation* (MLE): $P(token_i|D) = \frac{tf(token_i, D)}{\sum_{token_j} tf(token_j, D)}$. In [35], authors use the KL divergence on document models to detect the novelty of relevant documents in an adaptive filtering system and report promising

Table 2: Heuristics to create a virtual document v from top-k returned web pages.

Notation	Description
$D(m)$	Top m ($\leq k$) documents are concatenated
$D(sim, m)$	m of the top-k web pages where the contents have the highest similarity with the current contents of the entity in question (i.e., publication information available in the data set)
$T(1, n)$	Top n ($\in \{50, 100, 200, 1000\}$) tokens with the highest *tf*idf* weight from the top ranked web page
$T(all, n)$	Top n ($\in \{100, 200, 400, 1000, 10000\}$) tokens with the highest *tf*idf* weight from all top-k web pages
$T(each, n)$	Top n ($\in \{10, 20, 40, 100\}$) tokens with the highest *tf*idf* weight from each of top-k web pages
$T(common, p)$	Common tokens in at least p ($\in \{2, 3, 4\}$) documents from top-k web pages
$T(common, p, n)$	Common tokens in the top-n ($\in \{10, 20, 50\}$) tokens with the highest *tf*idf* weight of at least p ($\in \{2, 3, 4\}$) documents from top-k web pages
$S(m, n)$	A set of sentences of the top-m ranked web pages such that the sentence contains one of the top-n ($\in \{100, 200\}$) tokens with the highest *tf*idf* weight
$Snippet(m)$	Snippets of top-m ($\leq k$) web pages (i.e., short summary-like information provided by search engines) are concatenated

Table 3: Summary of data sets. Numbers in parenthesis represent average # of elements (i.e., citations, movie titles, etc.) for each entity

Data set	# of e_c	avg. # of e_v	avg. # of duplicates
ACM	43 (14.2)	21 (3.1)	1.8 (6.7)
ArXiv	64 (3.4)	9 (8.1)	1.3 (27)
IMDB	50 (24)	20 (24)	1 (23.5)

results compared to a simple content overlap based metric like Jaccard. For our language model based measures, we may adopt the same approach by measuring the novelty of canonical entity's virtual document, $D_i(e_c)$, against each candidate entity's virtual document, $D_j(e_v)$. Then, the lower the novelty score of $(D_i(e_c), D_j(e_v))$ is, the higher the similarity between two entities e_c and e_v. Moreover, we adopt to use the two smoothing techniques to adjust the MLE to assign non-zero probabilities to unseen tokens to make KL-based measure more appropriate as used by [35]:

- *Bayesian Smoothing using Dirichlet Priors.* This technique uses the conjugate prior for a multinomial distribution (i.e., the Dirichlet distribution) with parameters: $(\lambda p(token_1), ..., \lambda p(token_n))$. Then, the model is given by $P_\lambda(token_i|D) = \frac{tf(token_i,D)+\lambda p(token_i)}{\sum_{token_j}(tf(token_j,D)+\lambda p(token_j))}$. In consistent with [35], in our experiments, if $token_i \in D$, then we set $\lambda p(token_i) = 0.5$, and 0 otherwise.
- *Smoothing using Shrinkage.* This technique models each document from the language model of the document θ_{D_MLE} and a model for general English θ_{E_MLE} built from the tokens in all documents of the data set, by: $\theta_D = \lambda_D \theta_{D\_MLE} + \lambda_E \theta_{E\_MLE}$, where $\lambda_D + \lambda_E = 1$. We experimentally determine optimal value for λ_D and λ_E.

4. EXPERIMENTAL VALIDATION

In experiments, we seek to answer the following questions:

- Q1: Is Hypothesis 1 valid? That is, how do web based linkage schemes perform?
- Q2: Which of the proposed web based linkage schemes performs the best? Which variations?
- Q3: Which method to interpret the collective knowledge is better?
- Q4: How do results change for different search engines?

All experimentation was done using MySQL Server 5.0 on a desktop with AMD Opteron 2GHZ and 4GB RAM. As search engines for the web knowledge, we used both Google and Microsoft Live Search.

4.1 Set-up

We used three data sets, as summarized in Table 3 – *ACM Digital Library* for computer science authors with each authors respective citation list, *ArXiv Digital Library* for general science authors with her citation list, and *Internet Movie Database (IMDB)* for actors with movie-related data in which she stars. In this context, the linkage problem is to identify duplicate author entities in ACM and ArXiv and actor entities in IMDB.

For each canonical entity e_c, to find its duplicates, comparing e_c to each entity e_v in E ($e_c \neq e_v$) is prohibitively expensive (recall the naive algorithm in Section 2). Therefore, often, a pre-processing step, called *blocking*, pre-filters entities in E into a small set of candidate entities, called a *block*. In the experiments, we used a heuristic blocking rule that showed good performance for name-related entity resolution problem in [27] – i.e., all (author or actor) entities that share the same last name as the canonical entity are clustered into the same block. Then, duplicate entities are injected into the block. This block may contain as small as a few dozen entities to as large as hundreds of entities, depending on the last name in question. Hence, the goal is that for each block, a scheme is to detect all duplicates correctly.

Case I. From the *ACM Digital Library*, we have randomly gathered 43 real cases of duplicate names (thus 43 blocks). These real cases include duplicates as simple as "Dongwon Lee" vs. "D. Lee" (i.e., abbreviation) and "Alon Levy" vs. "Alon Halevy" (i.e., last name change) to as challenging as "Shlomo Argamon" vs. "Sean Engelson" (i.e., little similarity) or even ten duplicates of "Jeffrey D. Ullman". All 43 cases were verified by directly asking authors themselves or checking their home pages.

Case II. Second, we gathered 64 real cases from the *ArXiv Digital Library*. To make the case more challenging, for each block, we selected an author entity with the smallest number of citations as the canonical entity. One important difference from ACM case is that each entity block in ArXiv case tends to have larger number of citations.

Case III. Here we attempt to simulate a scenario in which

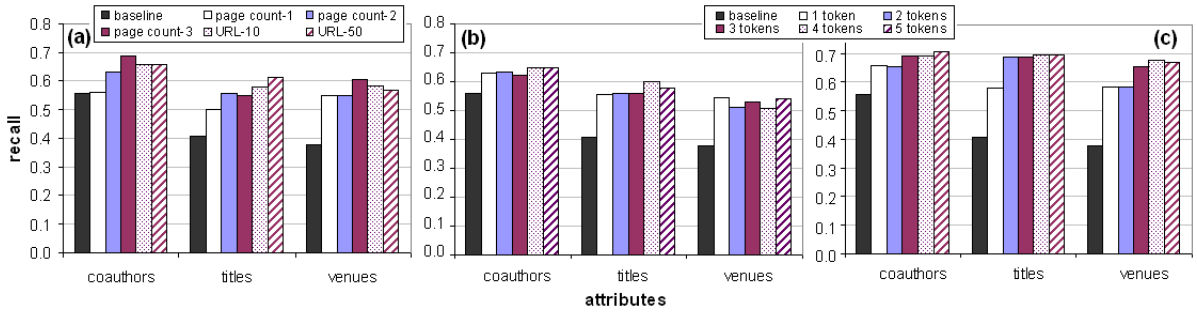


Figure 2: (a) ACM test case (using `Google` as search engine). Page count-1,2,3 – use *tf*, *tf*idf* and *w_tf*idf* for token selection, respectively. URL-10, 50 – use *tf*idf* for token selection and evaluate host names of top-10 and 50 URLs returned, respectively. (b, c) Effect of using multiple ($k = 2, 3, 4, 5$) tokens on Page count-2 and URL-10, respectively.

entities have large associated contents but they are not discriminative enough to help linkage. For this, we collected 50 real actor names and the titles of movies they appeared in from *Internet Movie Database* (IMDB). Since IMDB is a commercial database, it has little errors or noise, compared to ACM and ArXiv data sets. Therefore, we created 50 test cases such that each actor (i.e., canonical entity) has exactly one "a.k.a." name (i.e., duplicate entity). Since "a.k.a." name usually does not have any contents in IMDB, we randomly split the original contents of the actor, and assign each to an actor and her "a.k.a." entity. Due to the nature of the data set, the intersection of the two halves sharing the same content might be small – i.e., movie titles of an actor do not generally have a common theme.

In this context, the linkage problem is to identify duplicate author entities in ACM and ArXiv and actor entities in IMDB. Note that similar data sets have been used in the related work (e.g., [31, 27]).

Evaluation Metrics. We use traditional precision/recall with the window size equal to the number of duplicates. For instance, for a canonical entity e_c, suppose that there are k duplicates in the block. When a scheme returns top-k candidate duplicates, consider that only r of them are correct duplicates and the remaining $k - r$ candidates are false positives. Then, the precision and recall become the identical and are calculated as: $Precision = Recall = \frac{r}{k}$.

As a baseline approach for the linkage problem, we use the Jaccard similarity that measure the overlapping ratio of the contents of two entities ([27] reported that although simple, Jaccard similarity showed good performance among various standard distance functions):

$$sim_{jaccard}(e_c, e_i) = \frac{|token(e_c) \bigcap token(e_i)|}{|token(e_c) \bigcup token(e_i)|}$$

For the first two cases, our study focuses on three attributes of citations only – co-author, title, and venue – since (1) too many tokens from too many attributes slow down the process, (2) recent study [16] shows these three play a major role, and (3) other attributes have often missing values. Since ArXiv is a pre-print repository, it often does not have venue information. In the IMDB case, we only use the movie title attribute.

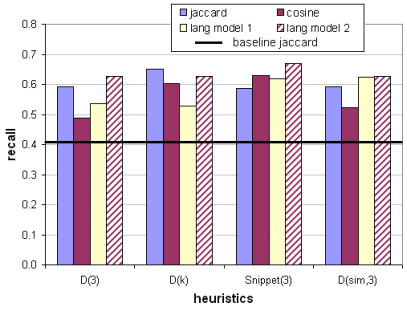


Figure 4: Results on the ACM title attribute using `Microsoft Live Search` as search engine.

4.2 Accuracy

Results of Page Count and URL Methods. Using the ACM data set, we first experiment with variations of the first two basic web based linkage methods: page count and URL methods. Figure 2(a) shows the results of these methods using `Google` against Jaccard metric on all three attributes of the ACM data set individually. All five variations of the basic methods performs better than the baseline Jaccard metric. Among three token selection schemes used by page count based methods, *w_tf*idf* yields better performance compared to the other two schemes, finding more discriminative tokens for entities. The two URL based methods show stable and near top performance on all cases probably due to the large volume of information used and the quality of the top returned documents by search engine. Figures 2(b,c) show the effect of using multiple tokens on the methods page count-2 and URL-10, respectively. Here, for each token selected in the top-k tokens of an entity, an experiment is run, and the similarity is calculated by averaging the individual similarities of all k experiments. The results are mostly parallel, and do not show significant improvement.

2. Web Page Based Methods. All experiments for the web page based methods were done using top-10 results from `Google` as the search engine. For the token selection method, we use *tf*idf* only, since it usually shows better discriminating power than *tf*, and *w_tf*idf* adds up extra time cost although it may have a better performance. Web page based

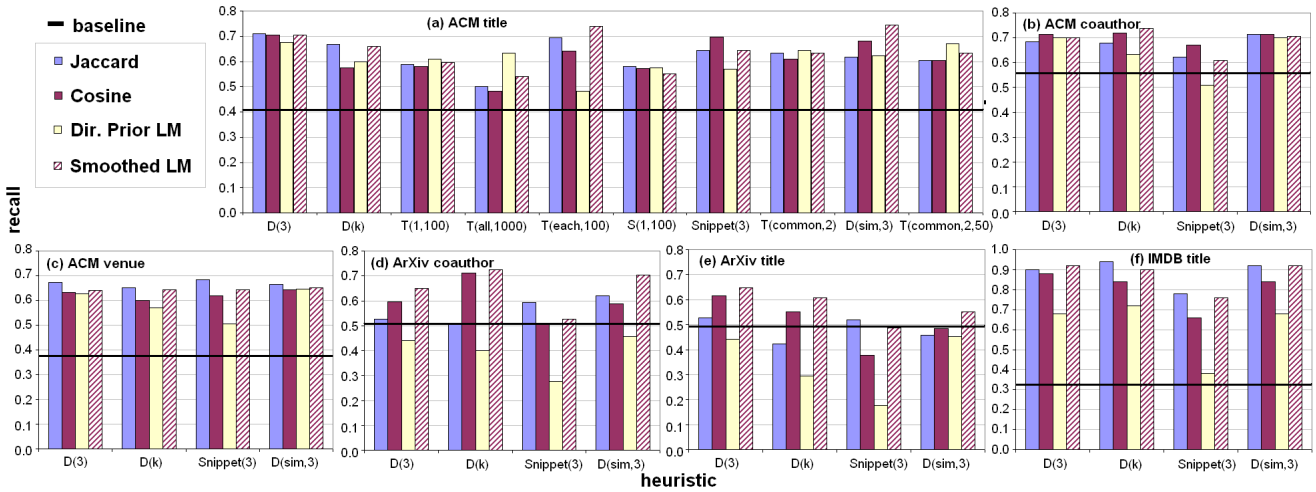


Figure 3: Results on the (a) ACM title, (b) ACM coauthor, (c) ACM venue, (d) ArXiv coauthor, (e) ArXiv title, and (f) IMDB title attributes (all using `Google` as search engine and *tf*idf* scheme for token selection).

methods are based on the creation of a virtual document from the web page results for each entity and then comparing these virtual documents to make a decision for the linkage. To create a virtual document, we defined a number of heuristics and their variations as listed in Table 2. Due to space constraints, we will only show a few variations for each experiment.

To determine which one to use, Figure 3(a) shows the results of 10 variations of our web page based heuristics on the title attribute of the ACM data set. In Figures 3(a-f), note that the *baseline jaccard* (i.e., the straight line) is the overlapping ratio of the "contents" of two entities while the plain *jaccard* is the overlapping ratio of the "virtual documents" of two entities. In our extensive experimentation, for the majority of cases, we saw that $top - 3$ returned pages are usually the most relevant ones and effective if used in virtual document creation. In other cases, using as many as returned web pages helped to achieve better precision/recall. Therefore, among the various heuristics, we selected *D(3)*, *Snippet(3)*, *D(sim,3)* due to their good performance. We also included one more variation of the first heuristic, *D(k)*, to illustrate the performance when all returned documents are used.

Figures 3(a-c) show the ACM results on the title, coauthor, and venue attributes, respectively. The baseline method performs well on the coauthor attribute but degrades on the other two. The reason for the degradation is that there are usually many tokens in the title attribute of an entity content, and for the venue field there are many common tokens like "conference", "ieee", which increase the false positive rate. On the other hand, all of our variations perform better and show around 60-70% recall on all attributes. Our methods show the best on the coauthor attribute too, due to the a more discriminative token selection coming from the last names of the author's collaborators.

Figures 3(d,e) show the results of the ArXiv data set on the coauthor and title attributes, respectively. The baseline Jaccard performs around 50% recall on both attributes. Note that most entities to be linked in this set have larger content and a lesser number of candidates in each block (Table 3), compared to the ACM case. Therefore, it should be easy for any metric to perform well. However, we here attempt to simulate a case where the canonical entities do not have much content and select the canonical entities from each block accordingly, explaining the mediocre recall value of the baseline approach. Although not as good as in the ACM case, our web based linkage schemes can still have a better recall than the baseline approach, yielding 72% recall at best for coauthor attribute. The smaller improvement on the title attribute stems from undesirable token selection from a small and not-so-good representative title tokens for the entities.

Figure 3(f) show the IMDB result on the movie title attribute. Expectedly, the baseline method shows very low performance having a poor recall value of 32%. Although each actor has enough content (about 24 titles), such content does not well identify an actor. However, this problem does not apply for the web based linkage methods which yield 94% recall at best implying that the approach is very promising for such cases.

3. Using A Different Search Engine. Our proposed web based linkage of using the collective knowledge from the Web is a general approach. As long as it can acquire such knowledge through a search engine, it can have a promising result on the linkage problem. Thus, we tested how results would change if we use different search engines – `Microsoft Live Search` instead of `Google`. For this set of experiments, we again use the ACM data set and tested the performance on the title attribute using `MSN`. Figure 4 shows the new results. The results of the heuristics are consistent with those of Google. While being relatively smaller, all four sets of experiments still outperform the baseline Jaccard. The results indicate that as long as we have access to some portion of the Web, we can utilize it to some extend as a collective knowledge source for the linkage problem.

4.3 Scalability

Since our methods rely on the knowledge acquired from the Web, each linkage decision process may incur a large

Table 4: Running times of the experiments on the ACM data set using the *tf* scheme on the title attribute.

Methods	Recall	Running time	# of Web accesses
Baseline - jaccard on the current content	0.405	0.09 min	0
Web based scheme - cosine using *D(3)* and Google	0.653	290.44 min	7951
Web based scheme - jaccard using *D(3)* and Google	0.711	289.32 min	7951
Web based scheme - lang. model 1 using *D(3)* and Google	0.573	289.29 min	7951
Web based scheme - lang. model 2 using *D(3)* and Google	0.756	289.47 min	7951
Web based scheme - cosine using *D(3)* and the local snapshot	0.463	9.53 min	0
Web based scheme - jaccard using *D(3)* and the local snapshot	0.443	9.38 min	0
Web based scheme - lang. model 1 using *D(3)* and the local snapshot	0.457	9.37 min	0
Web based scheme - lang. model 2 using *D(3)* and the local snapshot	0.494	9.39 min	0

number of Web accesses. Therefore scalability is a crucial problem. Apart from the computation time, the time spent on accessing the Web information introduces a considerable lag. As an example, the statistics of response time and the number of required web accesses of an experiment using the ACM data set and the tf scheme to select a representative token are shown in Table 4. Furthermore, the response time may be affected by many factors – network traffic, load of search engines, and web sites, etc. One solution is to reduce the *time* to access search engines – i.e., instead of using external search engines (i.e., Google or MSN), we can use local ones. As long as the local search engines have sufficient coverage of the Web, their behavior may parallel external search engines. To validate this idea, we use the WebBase data set [18] which contains about 100 million pages from 50,000 web sites. We built a local search engine using Apache Nutch from 3.5 million web pages after filtering out irrelevant 96.5 million pages. Then, all requests to external search engines are re-directed to this "local" search engine. Table 4 highlights the difference. Although the running time of the same experiments decreased dramatically, the recall of the web based linkage idea is still better than the baseline results, approximately 6% higher. Considering the small size of the local snapshot, however, the local search engine result proves that a sufficient performance can be achieved in a reasonable time once a reasonable size of data sources are used as the collective knowledge source like the Web.

5. RELATED WORK

The general linkage problem has been known as various names in many disciplines – record linkage (e.g., [14, 6]), citation matching (e.g., [26]), identity uncertainty (e.g., [28]), merge-purge (e.g., [17]), object matching (e.g., [9]), entity resolution (e.g., [30, 2]), authority control (e.g., [34, 19]), and approximate string join (e.g., [15]) etc.

Bilenko et al. [6] have studied name matching for information integration using string-based and token-based methods. Cohen et al. [12] have also compared the efficacy of string-distance metrics, like JaroWinkler, for the name matching task. [22] experimented with various distance-based algorithms for citation matching, with a conclusion that word based matching performs well. [27] conducted an in-depth study on the split entities case from the blocking point of view.

Unlike the traditional methods exploiting textual similarity, Constraint-Based Entity Matching (CME) [31] examines "semantic constraints" in an unsupervised way. They use two popular data mining techniques, Expectation-Maximization (EM) and relaxation labeling for exploiting the constraints. [4] presents a generic framework, *Swoosh* algorithms, for the entity resolution problem. The recent work by [13] proposes an iterative linkage solution for complex personal information management. Their work reports good performance for its unique framework where different linkage results mutually reinforce each other (e.g., the resolved co-author names are used in resolving venue names).

The recent trend in the linkage problem shows similar direction to ours (e.g., [6, 5, 20]). Although each work calls its proposal under different names, by and large, most are trying to "exploit additional information beyond string comparison." For instance, [20] presents a relationship-based data cleaning (RelDC) which exploits context information for entity resolution, sharing similar idea to ours. RelDC constructs a graph of entities connected through relationships and compares the connection strengths across the entities on the graph to determine correspondences. A more extensive and systematic study is needed to investigate the usefulness and limitations of the context in a multitude of the linkage problem. The main difference from ours to these approaches lies in the simplicity of our idea of using the Web to extract additional information.

Finally, to overcome the data incompleteness problem, several recent studies use the Web through search engines, similar to our proposal. Among those, [33] uses the statistical data by querying the Web for recognizing synonyms, and in [24], syntactic patterns are searched in the Web by using the Google API in order to acquire background knowledge for anaphora resolution. In addition, in [1], related texts are crawled from the Web to enrich a given ontology. In [11], another approach is employed to find the best concept for an unknown instance in a given ontology. [25] uses page counts to determine the strength of relations in a social network. A formal study by [10] defines a semantic distance metric between two terms using page counts from the Web. Similarly, [8] exploits the usage of page counts and lexical patterns from search engine snippets to measure semantic similarity between words. In [32], authors tackle another type of entity resolution problem known as "mixed citation problem" [23] using URL information. Finally, [21] independently explores the similar idea of using the Web to augment incomplete information for entity resolution. Since their focus is more on theoretical framework called "resource-bounded information gathering," our study on various methods to find the best representative tokens or scalability via local caching is complementary to their findings.

6. CONCLUSION

In this paper, we propose a novel approach toward the (record) linkage problem to identify duplicate named enti-

ties with insufficient description or contents. Unlike other approaches that use textual similarity of contents or name, our proposal unearths the hidden knowledge from the Web. Experimental results verify that our proposal improves the recall as high as 193% at best, and outperforms the baseline approach for a majority of test cases.

Our proposal relies on the information on the Web. As the Web evolves, we expect the information to get better. However it does not mean that it can always find reliable information. Moreover, the main limitation is that entities to be linked should have a good presence on the Web. Otherwise, even if the Web based linkage is used, the lack of information deters improvement.

Many directions are ahead for the future work. The proposed methods can be tested on more and larger data sets. More sophisticated approaches such as Information Gain, Chi-square etc., can be used to find better representative tokens or key phrases from contents of canonical entity or both entities to be linked. Using multiple representative data pieces, individual decisions or rankings can be combined to make a more accurate linkage decision by different schemes, i.e., voting, correlation etc. It is also important to be able to quantify the signal to noise ratio in selecting multiple information and combining them in queries to acquire the most representative data.

7. REFERENCES

[1] E. Agirre, O. Ansa, E. Hovy, and D. Martinez. "Enriching Very Large Ontologies Using the WWW". In *ECAI Ontology Learning Workshop*, Berlin, Germany, 2000.

[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. "Eliminating Fuzzy Duplicates in Data Warehouses". In *VLDB*, 2002.

[3] R. Bekkerman and A. McCallum. "Disambiguating Web Appearances of People in a Social Network". In *Int'l World Wide Web Conf. (WWW)*, 2005.

[4] O. Benjelloun, H. Garcia-Molina, Q. Su, and J. Widom. "Swoosh: A Generic Approach to Entity Resolution". Technical report, Stanford University, 2005.

[5] I. Bhattacharya and L. Getoor. "Iterative Record Linkage for Cleaning and Integration". In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2004.

[6] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. "Adaptive Name-Matching in Information Integration". *IEEE Intelligent System*, 18(5):16–23, 2003.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". *Journal of Machine Learning Research*, (3):993–1022, May 2003.

[8] D. Bollegala, Y. Matsuo, and M. Ishizuka. "Measuring Semantic Similarity between Words Using Web Search Engines". In *Int'l World Wide Web Conf. (WWW)*, pages 757–766, 2007.

[9] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. "Robust and Efficient Fuzzy Match for Online Data Cleaning". In *ACM SIGMOD*, 2003.

[10] R. Cilibrasi and P. M. B. Vitanyi. "Automatic Meaning Discovery Using Google", 2004.

[11] P. Cimiano, S. Handschuh, and S. Staab. "Towards the Self-annotating Web". In *Int'l World Wide Web Conf. (WWW)*, pages 462–471, 2004.

[12] W. Cohen, P. Ravikumar, and S. Fienberg. "A Comparison of String Distance Metrics for Name-matching tasks". In *IIWeb Workshop held in conjunction with IJCAI*, 2003.

[13] X. Dong, A. Y. Halevy, and J. Madhavan. "Reference Reconciliation in Complex Information Spaces". In *ACM SIGMOD*, 2005.

[14] I. P. Fellegi and A. B. Sunter. "A Theory for Record Linkage". *J. of the American Statistical Society*, 64:1183–1210, 1969.

[15] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. "Text Joins in an RDBMS for Web Data Integration". In *Int'l World Wide Web Conf. (WWW)*, 2003.

[16] H. Han, C. L. Giles, and H. Z. et al. "Two Supervised Learning Approaches for Name Disambiguation in Author Citations". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2004.

[17] M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". In *ACM SIGMOD*, 1995.

[18] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. "WebBase: A Repository of Web Pages". *Computer Networks*, 33(1–6):277–293, 2000.

[19] Y. Hong, B.-W. On, and D. Lee. "System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach". In *European Conf. on Digital Libraries (ECDL)*, Bath, UK, Sep. 2004.

[20] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. "Exploiting Relationships for Domain-independent Data Cleaning". In *SIAM Data Mining (SDM) Conf.*, 2005.

[21] P. Kanani, A. McCallum, and C. Pal. "Improving Author Coreference by Resource-bounded Information Gathering from the Web". In *AAAI IJCAI*, 2007.

[22] S. Lawrence, C. L. Giles, and K. Bollacker. "Digital Libraries and Autonomous Citation Indexing". *IEEE Computer*, 32(6):67–71, 1999.

[23] D. Lee, B.-W. On, J. Kang, and S. Park. "Effective and Scalable Solutions for Mixed and Split Citation Problems in Digital Libraries". In *ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS)*, Jun. 2005.

[24] K. Markert, N. Modjeska, and M. Nissim. "Using the Web for Nominal Anaphora Resolution". In *EACL Workshop on the Computational Treatment of Anaphora*, 2003.

[25] 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 *Int'l World Wide Web Conf. (WWW)*, pages 397–406, 2006.

[26] A. McCallum, K. Nigam, and L. H. Ungar. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching". In *ACM KDD*, Boston, MA, Aug. 2000.

[27] B.-W. On, D. Lee, J. Kang, and P. Mitra. "Comparative Study of Name Disambiguation Problem using a Scalable Blocking-based Framework". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2005.

[28] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. "Identity Uncertainty and Citation Matching". In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[29] G. Salton and M. McGill. *"Introduction to Modern Information Retrieval"*. McGraw-Hill, 1983.

[30] S. Sarawagi and A. Bhamidipaty. "Interactive Deduplication using Active Learning". In *ACM KDD*, 2002.

[31] W. Shen, X. Li, and A. Doan. "Constraint-Based Entity Matching". In *AAAI*, 2005.

[32] Y. F. Tan, M.-Y. Kan, and D. Lee. "Search Engine Driven Author Disambiguation". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2006.

[33] P. D. Turney. Mining the Web for synonyms: PMI–IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, 2167.

[34] J. W. Warnner and E. W. Brown. "Automated Name Authority Control". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, 2001.

[35] Y. Zhang, J. Callan, and T. Minka. "Novelty and Redundancy Detection in Adaptive Filtering". In *ACM SIGIR*, 2002.