Efficient Web-Based Linkage of Short to Long Forms

Yee Fan Tan[1] Ergin Elmacioglu[2] Min-Yen Kan[1] Dongwon Lee[3]

[1]Department of Computer Science, School of Computing, National University of Singapore, Singapore
[2]Department of Computer Science and Engineering, The Pennsylvania State University, USA
[3]College of Information Sciences and Technology, The Pennsylvania State University, USA

{tanyeefa,kanmy}@comp.nus.edu.sg {ergin,dongwon}@psu.edu

ABSTRACT

Abbreviations, acronyms, initialisms, and shortenings frequently occur in many texts found on the Web, such as publication metadata, stock ticker codes, and biological articles. To connect these disparate forms together for knowledge discovery, short forms must be properly linked to their canonical long forms. In this paper, we demonstrate how a search engine can be efficiently utilized in mining the required contextual information, so that short forms can be effectively linked to long forms. We show that a count-based method consistently outperforms other methods, and that using the snippets is better than using the full web pages. We also consider adaptively combining a query probing algorithm together with our count-based method. This reduces running time and network bandwidth, while maintaining the strong linkage performance.

Keywords

abbreviation matching, web as information resource, query probing, record linkage

1. INTRODUCTION

Proper nouns, technical terms, and long words are often shortened for saving space or improving clarity, due to writing style or convenience. Figure 1 shows typical examples of bibliographic references, randomly selected from [11], where publication venues have been abbreviated ("WebDB", "SIGIR", and "AAAI"). Figure 2 shows more examples in bibliographic publication venues, stock ticker symbols, and human genome research. However, it is not obvious how short forms are generated from long forms. For example, "MOU" includes the word "of" in its abbreviation, while "MIT" does not. Also, "UbiComp" is generated phonetically rather than by selecting initial letters of words. Further, different long forms can have the same short form, such as "ACSAC".

Short forms are a principal way in which variations are introduced to string representations of names, contributing to data quality issues when mining the Web. To aid knowledge discovery and uncovering implicit linkages, resolving short and long forms plays an important role in many data applications. In this paper, we study the problem of linking short forms to long forms. We believe that resolving short forms

E. Agichtein, P. G. Ipeirotis, and L. Gravano. Modeling query-based access to text databases. In *WebDB*, 2003.
J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, 1995.
W. W. Cohen. Learning trees and rules with set-valued features. In *AAAI*, 1996.

Figure 1: References with publication venues abbreviated.

DBLP Computer Science Conferences and Workshops	
ACSAC	Annual Computer Security Applications Conference
ACSAC	Asia-Pacific Computer Systems Architecture Conference
KDD	Knowledge Discovery and Data Mining
KDID	Knowledge Discovery in Inductive Databases
UbiComp	Ubiquitous Computing
WebDB	International Workshop on Web and Databases
NASDAQ Composite	
AAPL	Apple Inc.
CSCO	Cisco Systems, Inc.
DELL	Dell Inc.
INTC	Intel Corporation
MSFT	Microsoft Corporation
XRAY	DENSPLY International Inc.
Human Genome Acronym List	
MALDI	matrix-assisted laser desorption ionization
Mb	megabase
MGI	Microbial Genome Initiative
MHC	major histocompatibility complex
MIT	Massachusetts Institute of Technology
MOU	Memorandum of Understanding

Figure 2: Various examples of abbreviations.

to long forms is harder than the other way around, because the former requires the interpolation of missing data from abbreviations that are typically 3 to 5 letters long, while the latter only needs to discard extraneous data. Our problem statement is as follows:

> Given a set of short forms SF and a set of long forms LF, for each short form in SF, find the corresponding matching long forms in LF.

In this paper, we assume that no contextual information is available when linking short forms to long forms. For example, in Figure 1, we see the short forms of publication venues ("WebDB", "SIGIR", and "AAAI") but not their corresponding long forms. Hence, algorithms that detect short forms and their corresponding long forms in full-text documents (e.g., [3, 16, 18]) are inappropriate for our problem setting as they depend on the missing contextual information. Also, while a number of short to long form lists (e.g., acronym lists) are freely available, they are usually incomplete and quickly outdated, as new short forms are continually created. To remedy these problems, we explore

Proceedings of the 11th International Workshop on Web and Databases (WebDB 2008), June 13, 2008, Vancouver, Canada

the use of the Web as an additional knowledge source. In particular, we propose to use a *search engine* such as Google, Yahoo, or Live Search to obtain additional information that facilitates linking decisions. Our major contributions are as follows:

- We propose to exploit the external knowledge from the Web to obtain the required contextual information that is typically missing from such data. While similar work has employed the Web for other tasks, our work is unique, in being the first to specifically tackle the problem of short-to-long form matching and in unifying related threads of research on this theme. We view the task as two related facets: a) query composition, and b) search engine evidence analysis. In particular, we propose a count-based method that is effective for linking short forms to long forms.
- While our web-based method achieves good linkage accuracy, the search engine queries issued require a long running time to complete. Instead of taking an ad hoc approach to solve this problem, we algorithmically adaptively combine a *query probing* approach with our count-based method to save time and bandwidth while retaining performance.
- We compare our proposed method with other types of search engine evidence on three datasets of different domains. The results show that our claims consistently hold for all the three datasets.

This paper is organized as follows. In Section 2, we describe related work in the general linkage area. In Section 3, we first describe a framework that unifies various approaches that use a search engine to perform linkage. In Section 4, we propose a count-based method, and show its effectiveness compared to other linkage methods. In Section 5, we propose a query probing method, and adaptively combine it with a count-based method to reduce the number of search engine queries needed. In Section 6, we conclude the paper.

2. RELATED WORK

Most closely related to our work, [12] describe an approach that combined three kinds of data available in a search engine backend to extract short and long form pairs. In contrast, our approach does not assume access to search engine internals, and deals directly with constructing long-short form pairs using only the front end querying interface.

More generally, the linkage problem has been widely studied and is known by various names. A comprehensive survey is beyond the scope of this paper; introductions can be found in [7] and [20]. Among all record linkage works, web-based approaches are most relevant (e.g., [4, 5, 15]). Some works used conjunctive keyword queries – querying for $sf \wedge lf$ to see whether a short form sf and a long form lf are linked. As this results in quadratic time complexity, non-conjunctive keyword approaches have also been developed. These only query from one side – querying for sf to see whether the results have any evidence for lf. While efficient, this can lead to problems in accuracy and coverage. One key contribution of our work is to further improve upon this by introducing bi-directional non-conjunctive querying, resulting in higher accuracy while retaining linear complexity.

Recent work have also dealt with the finer details of using the Web for linkage evidence. [17] and [13] look at effective query expansion by using the Web. [14] extends this by identifying characteristic terms that differentiate namesakes. [6] suggested that if two strings refer to the same entity, then each string will frequently co-occur with some common information piece on the Web. Along these lines, our work again focuses on efficiency – examining how query bandwidth can be saved through the use of query probing.

Algorithm 1 Overall algorithm.

```
for each sf ∈ SF do
    for each lf ∈ LF do
        obtain information for sf and lf using search engine
        compute score_sf(lf) using obtained information
    rank the long forms in LF according to score_sf(lf)
```

3. LINKING SHORT TO LONG FORMS

In this paper, we use the following notation. Uppercase SF and LF denote the sets of short forms and long forms in the dataset, respectively. Lowercase sf and lf denote specific instances from SF and LF, respectively. $score_{sf}(lf)$ denotes a scoring function that ranks the long forms in LF that potentially match a particular short form sf.

Our intuition is that if a short form and a long form indeed refer to the same real-world entity, people would use them interchangeably on the Web. To link a set of short forms SF to a set of long forms LF, we consult a search engine for external linkage evidence, as illustrated in Algorithm 1. For a short form sf and a long form lf, we can derive information from a search engine, and use it to compute a *scoring function* $score_{sf}(lf)$ to rank the long forms in LF for a given short form sf. How to do this efficiently – in terms of query bandwidth – is the central focus of our work. Next, we investigate open details, such as the form of the queries and the computation of the scoring function.

Designing the Search Queries. As discussed, to gather linkage evidence, one can issue conjunctive keyword queries, i.e., $q = sf \wedge lf$, or issue non-conjunctive keyword queries, i.e., $q = sf$ or $q = lf$, or both. If $|SF| = m$ and $|LF| = n$, then conjunctive keyword queries require $O(mn)$ queries, which is of quadratic complexity and infeasible given long lists. On the other hand, non-conjunctive keyword queries only require $O(m)$ or $O(n)$ queries, or $O(m + n)$ queries, which is still feasible. Hence, we only consider non-conjunctive keyword queries in this paper. Note that we can add domain knowledge to the query to further filter results. For example, for DBLP publication venues, we may add the keywords "workshop" and "conference" to the query to promote web pages with publication metadata in the returned results.

Search Engine Evidence. A search engine typically returns a page of results (usually 10), and each result contains its rank, title, Keyword-In-Context (KWIC) snippet, and URL. The total number of results is also reported. We can choose how to process these information to determine linkage evidence, and specify follow-up actions as needed, e.g., download the web pages in the results. We can make repeated calls to the search engine to obtain multiple pages of results. The main focus here is on processing the search engine evidence, i.e., defining the scoring function via information obtained from a search engine.

4. COUNT-BASED LINKAGE METHODS

We propose methods for linking short forms to long forms

Dataset	Description	Short forms	Long forms	Matching pairs
DBLP	DBLP conference and workshop titles	906	920	926
	Source: http://www.informatik.uni-trier.de/~ley/db/conf/indexa.html Query: "⟨*title*⟩" conference OR conferences OR workshop OR workshops			
NASDAQ	NASDAQ Composite stock symbols	3084	3061	3084
	Source: http://www.nasdaq.com/asp/index_component.asp?symbol=IXIC Query: "⟨*title*⟩" nasdaq			
GENOMES	Human Genome Acronym List	307	307	307
	Source: http://www.ornl.gov/sci/techresources/Human_Genome/acronym.shtml Query: "⟨*title*⟩" genome OR genomes			

Table 1: The evaluation datasets.

Human Genome Project - Wikipedia, ... "Genomes: 15 Years Later A Perspective by ...
More on the sequencing of the human genome The international **Human Genome Project** (HGP) ...
More on the sequencing of the human genome ... Approximately 60% of the underlying sequence data ...

Figure 3: Snippets (simplified) from the query "HGP".

Algorithm 2 Computing $count(sf \to lf)$ by obtaining top-k search engine results for each short form.

```
for each sf ∈ SF do
    D = SearchEngineTop(sf, k)
    for each lf ∈ LF do
        count(sf → lf) = number of results in D containing lf
```

by counting the terms in the returned results. These methods use the $SearchEngineTop(q, k)$ function, which queries a search engine with q and retrieves top-k results. We define the following scoring functions:

- $count(sf \to lf)$ is the number of results (snippets or web pages) of the short form sf containing the long form lf. For a simplified illustration, suppose sf is "HGP" and lf is "Human Genome Project". We query a search engine with "HGP" and suppose we consider only the top-3 results, whose snippets are as shown in Figure 3. As two of these snippets contain "Human Genome Project", we have $count(sf \to lf) = 2$. Algorithm 2 shows this algorithm more formally.
- $count(sf \leftarrow lf)$ is the number of results of the long form lf containing the short form sf. It can be obtained by interchanging sf and lf in Algorithm 2.
- $count(sf \leftrightarrow lf) = count(sf \to lf) + count(sf \leftarrow lf)$ is a combination of the previous two.

While web-based and traditional record linkage techniques have been applied to other tasks, to our knowledge, no study has yet to examine the efficacy of these techniques on the task of short form to long form matching. To our knowledge, our less computationally expensive $O(m+n)$ $count(sf \leftrightarrow lf)$ scoring function is a new contribution that may assist in other web-based linkage tasks.

4.1 Comparison with Other Types of Evidence

We will evaluate our count-based methods against three other methods adapted to solve the same problem.

Schwartz and Hearst. This algorithm [18] is a state-of-the-art abbreviation extraction algorithm that expects full text input and does not use the Web. It scans full-text articles for fragments of the form "lf (sf)", and tries to extract the long form lf with the best alignment with the short form sf in a greedy manner. We apply this algorithm by creating a file containing all combinations of short forms and long forms in the dataset, with lines of the form "lf (sf)". This extraction algorithm returns candidate matching pairs without scoring their quality, but instead indicates which part of the long form matches (e.g., the "Key Cryptography" part of "Public Key Cryptography"). Therefore, we define the scoring function to be the fraction of characters in the long form that was matched.

Inverse host frequency. This web-based method uses URL information from search engine results. For a query q, we form a feature vector v_q of hostnames (or domain names) from the URLs of its top-k search engine results, weighted by its *inverse host frequency* (IHF) [19], i.e., $\text{IHF}(h) = \log_2 \frac{\max_h freq(h)+1}{freq(h)+1} + 1$. Here, $freq(h)$ is the number of short and long form queries whose top-k results contain the hostname h. For a short form sf and a long form lf, the cosine similarity between v_{sf} and v_{lf} is the scoring function.

Sahami and Heilman. This is a web-based information retrieval method [17] that we reimplemented. For each query q, we download the web pages at the URLs of its top-k results, $w_{q,1}, \ldots, w_{q,k}$. For each web page $w_{q,i}$, we compute its tf-idf vector $v_{q,i}$. Following Sahami and Heilman, we truncate $v_{q,i}$ to include only the 50 tokens with the highest tf-idf weights, and then normalize it. Next we compute $v_q = \frac{1}{k}\sum_{i=1}^{k} v_{q,i}$, and when normalized, it is the query expansion vector of q. For a short form sf and a long form lf, the cosine similarity between v_{sf} and v_{lf} is the scoring function.

4.2 Datasets and Search Engine

To validate our methods, we examine real-world problems of matching short and long forms. In all experiments, we used the Google search engine via its SOAP Search API. We used three datasets from different domains for our evaluation: DBLP, NASDAQ and GENOMES which have very different characteristics. Each dataset contains the SF and LF sets, as well as the solution set S of matching short and long form pairs. We applied all the methods using SF and LF as input, and used S as the gold standard. Table 1 summarizes these datasets, and lists the form of queries used. The additional keywords in the queries were selected based on the domain of the dataset, with all web-based methods using the same queries. Examples from each dataset are shown in Figure 2.

The DBLP dataset consists of computer science conferences and workshops in the DBLP digital library. This dataset is generally clean, because DBLP was manually constructed and consistently uses full words in more than 98% of its long forms. The retrieved web pages tend to be conference and workshop web sites, and publication lists on academic homepages and research groups. The web pages sometimes contain typos and spelling mistakes. The NASDAQ dataset consists of stock symbols in the NASDAQ Composite index. This large list itself is also fairly clean, but there are stock symbols that have no resemblance to the company name at all, such as "XRAY" for "DENSPLY International Inc.". The retrieved web pages come from a large variety of domains and mainly consist of financial news and stock information. Some of these web pages appear to be automatically generated from databases, hence these can be fairly clean. The GENOMICS dataset consists of abbreviations that are commonly used in the human genomics domain. It is the smallest and noisiest among the three datasets. Some long forms have abbreviated words like "Univ." and "Intl.", but their usage is inconsistent. Worse, this list includes academic institutions, academic conferences, government organizations and biological terms. Hence, the retrieved web pages have all kinds of sources and information.

For the count-based methods, as well as Sahami and Heilman method, we evaluated on both the snippets and the downloaded web pages. For all methods except Schwartz and Hearst, we evaluated with $k \in \{10, 20\}$, i.e., using top-10 or top-20 search engine results, except for web pages where we only show results for $k = 10$.

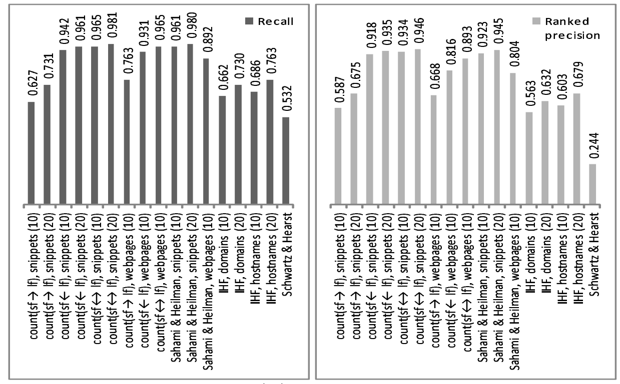
4.3 Evaluation

To evaluate the various methods, we use average recall and average ranked precision as our evaluation metrics. Suppose a short form sf corresponds to R long forms, and the top-10 candidate long forms in the ranked list for sf contains r correct long forms. Then, the recall is $\frac{r}{R}$. To account for the quality of rankings, we use *ranked precision* [10] instead of traditional precision. Let P_i be the fraction of correct long forms within the top-i candidates. Let C be the set of positions of the correct long forms within the top-10 candidates. Thus, the ranked precision is $\frac{\sum_{i \in C} P_i}{r}$. The *average recall* and *average ranked precision* are the averages of recall and ranked precision over all short forms in our dataset.

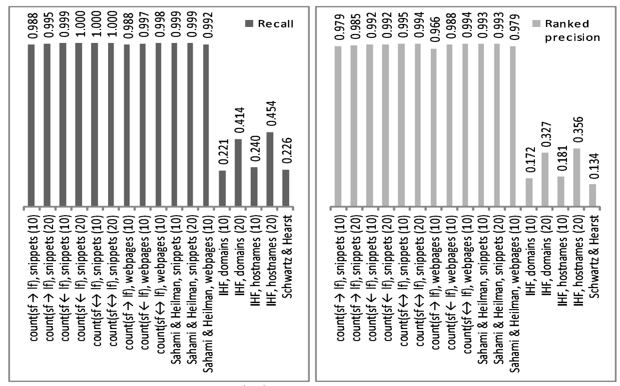
Figure 4 shows the results. Overall, we found that the count-based methods with snippets tend to produce the best performance. In particular, we found that $count(sf \leftrightarrow lf)$ consistently produces the best results. The $count(sf \leftarrow lf)$ and $count(sf \leftrightarrow lf)$ methods usually achieved average recall and average ranked precision higher than 0.9.

As noted, the count-based methods with snippets tend to produce the best performance, particularly $count(sf \leftrightarrow lf)$. Interestingly, $count(sf \leftarrow lf)$ is much better than $count(sf \rightarrow lf)$, though they differ only in direction. For $count(sf \rightarrow lf)$, many short form queries have no snippets containing the corresponding long forms, and some short forms are common words (e.g., "STEP") leading to irrelevant results, affecting both precision and recall. On the other hand, long forms make more informative multi-word queries, and often appear where short forms are defined. Finally, $count(sf \leftrightarrow lf)$ combines the best of the two unidirectional methods, and gives the best performance at the expense of doubling the number of queries. This is because $count(sf \leftarrow lf)$ is often able to obtain relevant information when $count(sf \rightarrow lf)$ misses out.

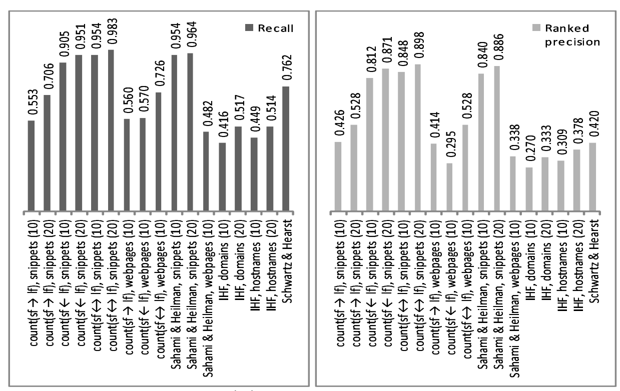
The Schwartz and Hearst algorithm generally did not perform very well, and is one of the worst performing algorithms for the DBLP and NASDAQ datasets. This is not surprising because many long form candidates share many common



(a) DBLP



(b) NASDAQ



(c) GENOMES

Figure 4: Average recall and ranked precision for the various kinds of evidence for the three datasets. Numbers in parentheses represent number of results retrieved per query.

letters as the short form. Therefore, Schwartz and Hearst is unable to distinguish between them. On the other hand, this algorithm performs better on the GENOMES dataset because this dataset is much smaller and hence fewer candidates to choose from. However, its recall suffers because it is unable to match when the short form contains a letter that is not found in the long form. Also, the greedy algorithm sometimes gives only a partial match to the correct long form (e.g., for "AAAI", only the last four words of "American Association for Artificial Intelligence" is matched). Such a weakness suggests that the given data itself lacks the required context to solve the problem, hence we need to obtain Web information to perform the matching.

The Sahami and Heilman algorithm is a strong contender for the top spot, but in all three datasets, $count(sf \leftrightarrow lf)$ always wins by at least a small margin, for both snippets as well as web pages. While both algorithms are effective in linking short forms to long forms, the computation for $count(sf \leftrightarrow lf)$ is both simpler and faster. The main difference between their performance comes from those cases where there are very few search engine results containing both the short form and its correct corresponding long form: $count(sf \leftrightarrow lf)$ tends to get it right while Sahami and Heilman finds all web pages are almost equally dissimilar.

IHF works well when the web pages of a matching short form and long form come from common sources, i.e., share common domains or hostnames. This is somewhat true for the DBLP dataset, therefore using only the URLs alone gives a fairly competitive algorithm. However, for the NASDAQ dataset, some web sites aggregate information for almost all of the stock symbols, therefore the common domains or hostnames have little discriminating power and make IHF ineffective. In all cases, using hostnames is slightly better than using domain names.

In practically all cases, obtaining 20 results gives better performance than obtaining only 10 results, which is not surprising. What is more interesting, is that using web pages is not as good as using snippets for most of the methods, particularly for the GENOMES dataset. This is likely because web pages contain a lot of noisy information, e.g., many web pages for the query "Advances in Data Base Theory" contain "AAAI" in navigation bars and external links. Hence, snippets might be better as passage retrieval is already done.

In summary, our $count(sf \leftrightarrow lf)$ method on snippets outperforms the other methods in terms of accuracy. However, the biggest drawback of any web-based method (including ours) is that they require search engine queries and/or web page downloads, both of which are expensive on running time. We remedy this drawback in the next section.

5. ADAPTIVE COMBINATION

In general, adaptive methods combine base methods to obtain the better aspect of each. It has been adopted in data integration work (e.g., [21]), duplicate detection (e.g., [1]), and determining when to search or when to crawl [11]. For our problem of linking short forms to long forms, we can adaptively combine a more accurate but slower method M_s with a weaker but faster method M_w so that we can obtain a combined method whose accuracy is closer to that of M_s and yet runs much faster than M_s. A general adaptive framework is shown in Algorithm 3, in which we allow M_w to resolve each short form to the corresponding long forms, and apply M_s only to those short forms whose candidate long forms appear to be incorrect according to some heuristic H. In this way, we reduce the execution time needed by making fewer calls to M_s.

Such an adaptive framework is very versatile and can apply to many kinds of methods. In this paper, we use a query probing method as our M_w and a count-based method as our M_s. We describe our query probing method next.

Algorithm 3 Adaptively combining a weaker method with a stronger method.

Input: a weaker method M_w, a stronger method M_s, and a heuristic H

for each $sf \in SF$ **do**
  for each $lf \in LF$ **do**
    compute $score_{sf}(lf)$ using M_w
  if H determines that scores by M_w gives a poor ranking of long forms **then**
    for each $lf \in LF$ **do**
      compute $score_{sf}(lf)$ using M_s
    return scores computed by M_s
  else
    return scores computed by M_w

Algorithm 4 Query probing.

NG = set of n-grams contained by the long forms in LF
$D = \emptyset$
for each $ng \in NG$ **do**
  if number of long forms in LF containing the n-gram $ng \geq min\_freq$ **then**
    $D = D \cup SearchEngineTop(ng, k_p)$
for each $sf \in SF$ **do**
  for each $lf \in LF$ **do**
    $count_p(sf, lf)$ = number of results in D containing both sf and lf

5.1 Query Probing

Query probing is the automatic extraction of information from a "hidden web" database by selecting suitable terms (called *query probes*) to query [8]. This approach has been used to obtain language models [2], and to estimate word frequency in different languages [9]. In our context, we can issue query probes to a search engine to derive approximate Web statistics, and use it to reduce the number of queries.

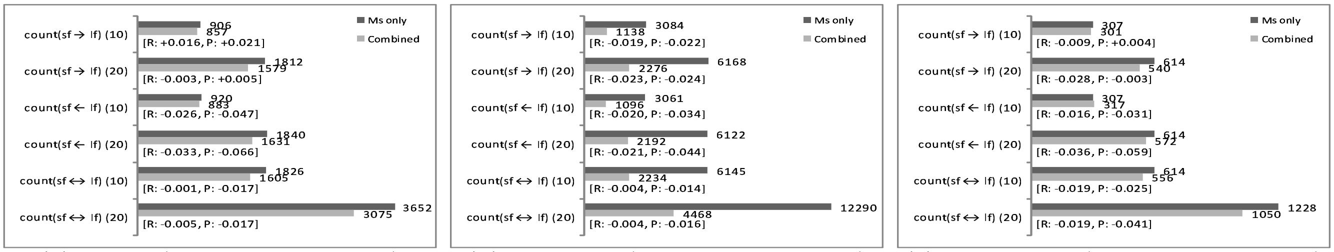
Consider three conferences "Joint Conference on Digital Libraries", "European Conference on Digital Libraries" and "Digital Libraries" and their respective short forms "JCDL", "ECDL" and "DL". Normally, we will query all three long forms to obtain $count(sf \rightarrow lf)$, query all three short forms to obtain $count(sf \leftarrow lf)$, and query all six short and long forms to obtain $count(sf \leftrightarrow lf)$. However, we observe that many long forms share common n-grams[1], such as the 2-gram "digital libraries" in our example. A single query probe "digital libraries" (together with some domain-specific keywords) yields results for all three (long form) conferences in the top-10 results, and the snippets contain all three short forms. Thus, when compared to $count(sf \rightarrow lf)$, $count(sf \leftarrow lf)$ and $count(sf \leftrightarrow lf)$, query probing can save two, two, and five queries, respectively.

With this observation, we devised a query probing algorithm that probes the search engine with n-grams that occur in at least min_freq of the long forms. Our query probing algorithm is shown in Algorithm 4. We use the top-k_p search engine results to obtain $count_p(sf, lf)$, the number of results that contain both sf and lf, and use it as a scoring function.

5.2 Adaptively Combining Query Probing with Count-based Methods

The main weakness of the query probing scoring function, $count_p(sf, lf)$, is that typically some of the short forms sf have no candidate long forms, i.e., $count_p(sf, lf) = 0$ for all long forms lf, and vice versa. However, for short forms with candidate long forms, query probing is able to save a

[1]In this paper, the term *n-gram* is used at the token level. Hence, a 3-gram refers to a three-token subsequence.



(a) DBLP ($n = 3$, $min\_freq = 3$) **(b)** NASDAQ ($n = 2$, $min\_freq = 2$) **(c)** GENOMES ($n = 3$, $min\_freq = 2$)

Figure 5: Number of search engine calls using M_s alone, and adaptively combining M_s with query probing using parameters n-grams and min_freq as shown, and $k_p = 10$, where numbers in parentheses represent the number of snippets retrieved per query. The change in average recall (R) and average ranked precision (P) are indicated in square brackets.

significant number of search engine queries.

Therefore, we propose adaptively combining query probing with a count-based method using the proposed framework of Algorithm 3. In other words, we use $count_p(sf, lf)$ as our weaker method M_w and use one of $count(sf \rightarrow lf)$, $count(sf \leftarrow lf)$, or $count(sf \leftrightarrow lf)$ as our stronger method M_s. The heuristic H we use is: *If there is no long form* lf *in* LF *with* $count_p(sf, lf) > 0$*, then we apply the stronger method* M_s*.*

5.3 Evaluation

In addition to average recall and average ranked precision, we also use the number of search engine calls to evaluate the effectiveness of our adaptive combination of query probing and count-based methods. As each search engine call returns 10 results, we will need two calls for 20 results.

We experimented with various parameter settings, and found that usually $k_p = 10$ is sufficient to provide the largest decrease in the total number of search engine calls needed. The results for selected values of n and min_freq for each dataset are shown in Figure 5. Compared to the count-based methods alone, our adaptively combined methods can reduce the reduce the number of search engine calls in practically all cases, with better savings when the stronger count-based method uses 20 snippets. The largest dataset, NASDAQ, also gave the most significant savings of 18.0% to 31.6%. The change in average recall and average ranked precision is relatively small, with decreases of up to 0.036 and 0.066, respectively; while task performance actually improved in a few cases. Other query probing parameter settings also gave similarly insignificant changes in average recall and average ranked precision. Therefore, we conclude that query probing can reduce the number of search engine calls significantly while maintaining task performance.

6. CONCLUSION

The presence of both short forms and long forms in record fields poses unique challenges to various searching and record linkage tasks. Approximate string matching does not tend to work well due to drastic differences in the application scenario. Our work formalizes the use of search engines as the two facets of query formulation and search engine evidence analysis. In formulation, we proposed the use of inverted and bidirectional non-conjunctive queries. In evidence analysis, we surveyed techniques using gathered URLs and snippets, as well as downloaded web pages. Our proposed count-based methods, particularly $count(sf \leftrightarrow lf)$, is found to be the most effective in linking short forms to long forms. As an additional contribution, we describe how we can adaptively combine two methods, using query probing as the first stage before trying (if needed) our standard count-based method. The combined method saves significantly on querying while retaining a good performance, making our algorithm scalable for large record linkage problems.

7. REFERENCES

[1] M. Bilenko, R. J. Mooney, W. W. Cohen, P. Ravikumar, and S. E. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 2003.

[2] J. P. Callan and M. E. Connell. Query-based sampling of text databases. *ACM TOIS*, 2001.

[3] J. T. Chang, H. Schütze, and R. B. Altman. Creating an online dictionary of abbreviations from MEDLINE. *JAMIA*, 2002.

[4] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *WWW*, 2004.

[5] P. Cimiano, G. Ladwig, and S. Staab. Gimme' the context: Context-driven automatic semantic annotation with C-PANKOW. In *WWW*, 2005.

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

[7] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE TKDE*, 2007.

[8] L. Gravano, P. G. Ipeirotis, and M. Sahami. QProber: A system for automatic classification of hidden-web databases. *ACM TOIS*, 2003.

[9] G. Grefenstette and J. Nioche. Estimation of English and non-English language use on the WWW. In *RIAO*, 2000.

[10] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *ACM SIGIR*, 1993.

[11] P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. To search or to crawl? Towards a query optimizer for text-centric tasks. In *ACM SIGMOD*, 2006.

[12] A. Jain, S. Cucerzan, and S. Azzam. Acronym-expansion recognition and ranking on the web. In *IEEE IRI*, 2007.

[13] K.-L. Kwok, L. Grunfelda, and P. Deng. Employing web mining and data fusion to improve weak ad hoc retrieval. *Information Processing and Management*, 2007.

[14] 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.

[15] J.-H. Oh and H. Isahara. Hypothesis selection in machine transliteration: A web mining approach. In *IJCNLP*, 2008.

[16] N. Okazaki and S. Ananiadou. Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 2006.

[17] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, 2006.

[18] A. S. Schwartz and M. A. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *PSB*, 2003.

[19] Y. F. Tan, M.-Y. Kan, and D. Lee. Search engine driven author disambiguation. In *ACM/IEEE JCDL*, 2006.

[20] W. E. Winkler. Overview of record linkage and current research directions. Technical report, U.S. Bureau of the Census, 2006.

[21] Y. Zhu, E. A. Rundensteiner, and G. T. Heineman. Dynamic plan migration for continuous queries over data streams. In *ACM SIGMOD*, 2004.