

# Mapping Web Pages to Database Records via Link Paths

Tim Weninger

Fabio Fumarola<sup>†</sup>

Jiawei Han

Donato Malerba<sup>†</sup>

University of Illinois at Urbana-Champaign

<sup>†</sup> Università degli Studi di Bari “Aldo Moro”

weninge1@illinois.edu, ffumarola@di.uniba.it, hanj@illinois.edu, malerba@di.uniba.it

## ABSTRACT

In this paper we propose a new knowledge management task which aims to map Web pages to their corresponding records in a structured database. For example, the DBLP database contains records for many computer scientists, and most of these persons have public Web pages; if we can map the database record with the appropriate Web page then the new information could be used to further describe the person’s database record. To accomplish this goal we employ *link paths* which contain anchor texts from multiple paths through the Web ending at the Web page in question. We hypothesize that the information from these link paths can be used to generate an accurate Web page to database record mapping. Experiments on two large, real world data sets, DBLP and IMDB for the structured data and computer science faculty members’ Web pages and official movie homepages for the Web page data, show that our method does provide an accurate mapping. Finally, we conclude by issuing a call for further research on this promising new task.

## Categories and Subject Descriptors

H.2.8 [Database Management]: Database applications—*data mining*; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

## General Terms

Algorithms, Experimentation

## Keywords

mapping, Web, link paths, semi-structured data

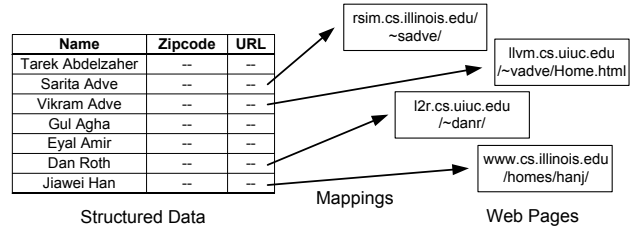
## 1. INTRODUCTION

The World Wide Web contains a wealth of information, and it is rapidly expanding in size and scope. Despite the vast complexities of the Web’s landscape, billions of people, even young children, are able to navigate the Web with relative ease. This is partly due to the usefulness of modern Web browsers, search engines and Web design techniques, and partly due to the link-based construction of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM 2010 Toronto, Canada

Copyright 2010 ACM X-XXXXXX-XX-X/XX/XX ...\$10.00.



**Figure 1: Problem example wherein two columns Zipcode and URL are added to a database schema, and the URL column is mapped to Web pages corresponding to the Name column. From this mapping the Zipcode column may be populated by extracting data from the Web page.**

the Web itself. Arguably, the aspect most fundamental and essential to the ongoing operation of the Web is the existence of hyperlinks. These page-to-page links have shown their ability to tame the Web over and over again, and has transformed an otherwise unwieldy mass of documents into information accessible to the World.

Structured databases of all types have grown alongside the World Wide Web. Recently, efforts have been made to bridge the gap between this structured data and the unstructured data from the Web, but most of these efforts have been one-way. That is, most current work focuses on extracting structured information from one or more Web pages. While this is an important task, if technology could be provided to map specific Web pages to records in a database then the structured and unstructured data could be used to mutually enhance each other in order to address many difficult problems. Therefore, mapping structured database records to Web pages is a specific challenge to the database and information retrieval community.

Consider the example shown in Figure 1 wherein a structured database holds a list of computer science faculty members with the URL and zipcode fields empty. If we could discover a mapping from each faculty members’ record to their personal Web page then we would be able to mine the Web pages for zipcodes in order to fill the missing values in the structured schema. With this information we could perform simple tasks like plotting each professor on a map, or more interesting tasks like examining the geographical density and distribution of computer scientists, etc. To accomplish this task we first require the ability to systematically mine the Web for information useful for matching records with Web pages.

In the past decade, search engines have managed to harness some of the information found in hyperlinks. Perhaps the best known example of using the power of hyperlinks is the PageRank algorithm, which fuels, in part, the ongoing success of the Google search en-

gine [1]. Similar success stories can be found within the realms of link-based social networks, communications systems, and even protein interactions. Each specific method focuses on the connectivity, link strength, centrality, etc. of the underlying graph to build models of retrieval, clustering and/or classification. Typically, the edges in these graphs are labeled with numerical edge weights corresponding to the probability of edge traversal (e.g., clicks), friendship, etc. depending on the problem domain.

On the World Wide Web, to supplement the numerical PageRank-type probabilities, edges can be assigned labels according to their associated anchor text. For example, the HTML hyperlink `<a href="www.yorku.ca/cikm10/">CIKM Conference</a>` can be annotated by the anchor text, ‘CIKM Conference’. It is a widely accepted practice for search engines to index an inbound link’s anchor text because “anchors often provide more accurate descriptions of Web pages than the pages themselves” [1]. This observation did not start with Google, in fact, the idea of indexing incoming anchor text with the page it refers to was implemented in the World Wide Web Worm in 1994 [18], and since then dozens of studies have looked at various ways to leverage anchor text information.

Metzler *et al.* use *aggregated anchor text*, which are made of anchor texts originating outside of a Web site that link to Web pages inside a Web site that, in turn, link to the Web page in question. They show that aggregated anchor texts improve retrieval effectiveness over current approaches that focus mainly on the inbound links directly adjacent to the Web page [19]. In our study, we expand upon the findings of Metzler *et al.*, and further argue that our notion of link paths are more effective at the Web page to database record mapping task than the use of links directly adjacent to a Web page.

The main contributions of this paper are as follows: (1) We formulate a new knowledge management task which aims to map Web pages to their corresponding structured database record; (2) We define link paths and show that they are able to represent the referenced Web page more effectively than existing methods; (3) We use link paths to generate mappings from Web pages to their appropriate records in a structured database; (4) We perform a case study to judge the effectiveness of our approach and demonstrate the implications of this new task.

The remainder of this paper is organized as follows. In Section 2 we survey related work with special attention paid to the various uses of anchor text. The general framework of our approach is introduced in Section 3. In Section 4 we define link paths and highlight some of their interesting properties. Next, in Section 5 we show how information from link paths can be used to map Web pages to structured data. Then we perform an experimental evaluation of this method in Section 6 and compare the results with a worthy baseline in Section 7. In Section 8 we conclude the paper and offer several avenues for future research.

## 2. RELATED WORK

The first step in mapping Web pages to database records is the retrieval of candidate Web pages. This task is commonly known as the homepage finding task, and this objective was one of the topics of the 2002 TREC Web Track [5]. The Named Page Finding Task, specifically, aimed to find Web pages which represent specific named entities. For example, a query on the computer scientist “Dan Roth” should return his homepage `l2r.cs.uiuc.edu/~danr/`. The top ranked results of the TREC competition used some combination of anchor text, hyperlinks and content filtering – methods very similar to the baseline algorithms used in our experimental section. Further research on this topic has involved learning classification models using features based on URL composition and

anchor texts [24].

Most of the recent work in bridging structured and unstructured data focuses on the information extraction task. This task rightly assumes that many Web pages are constructed by merging an HTML template with database records; the goal, therefore, is to automatically reverse that process in order to extract the structured database records from the semi-structured Web page.

Several methods have been devised to accomplish this task. Liu *et al.*’s Mining Data Records (MDR) algorithm finds patterns in the HTML code of a Web page to extract data records. This work was later extended by Zhai and Liu to extract patterns via partial tree alignments from the Web page’s DOM tree [26], and again by Miao *et al.* to extract data records by clustering HTML tags from DOM paths [20]. Unfortunately, these types of algorithms assume that each Web page contains two or more similarly structured records. Cafarella *et al.* showed that Web tables represented database records [2, 3], and Lin *et al.* used these Web tables to discover entity relationships on the Web [15]. For more information on this type of work see Hovey’s survey paper [12] or Liu’s Web Data Mining book [16]. Each of these information extraction techniques are useful for creating structured database records from Web pages. However, none of these works address the problem of mapping existing database records to their corresponding Web page.

Unlike the lack of research in the mapping of Web pages to structured database records, anchor text has been widely studied and used as an important source of relevant information for many years. McBryan was one of the first to associate anchor texts of incoming links to a Web page in the World Wide Web Worm tool [18]. Later, Brin and Page explained the importance of associating anchor texts with both the Web page it occurs on as well as the Web page it points to [1]. Harmandas *et al.* showed that anchor texts could also be used to annotate Web pages containing only images [11]. We also find that, due to their importance and descriptive ability, anchor texts are currently used in most commercial search engines.

A great deal of work has been done regarding how to best use anchor texts in the area of information retrieval. Most of this related work looks how to rank documents. Especially relevant is the popular BM25 model which was extended by Robertson *et al.* to include multiple weighted fields in documents [22, 21]. The particular focus of our work is not on ranking functions, but rather to show the usefulness of anchor texts in structured databases.

Anchor texts have shown to be beneficial in query rewriting. Jin *et al.* first demonstrated that document titles bear a close resemblance to common queries because of their succinct, descriptive nature, and because they are typically developed via a similar cognitive process [13]. Eiron and McCurley argue that anchor texts are even more beneficial than Web page titles because (1) Web pages can have only one title while several distinct anchor texts can point to each Web page, and (2) different anchor texts can help with the problem of synonymy. They further show that anchor texts also represent real-world user queries in term distribution and length [8]. Fujii *et al.* proposed a model to identify synonymous query terms in anchor texts in order to expand queries [10]. Kraft and Zien proposed a method that mines clicked anchor texts in concert with query logs in order to learn query rewriting rules to improve search engine results [14]. Shen *et al.* draw implicit links between certain Web pages by the observation that people who search the Web with the same queries often click on different, yet related documents. These implicit links are then used with explicit anchor texts to aid in document classification [23]. In sum, these works provide valuable insight into the nature of anchor texts. Specifically, these, and other studies, argue that anchor texts are similar in form and

function to common queries. We use this understanding in the next section where we formulate the details of our algorithm.

Other uses of anchor text include Craswell *et al.*'s work on site finding where all anchor texts to a Web page are treated as a separate 'anchor document.' They show that anchor texts are more useful than content words in retrieval for certain types of queries. Westerveld *et al.* used a similar method, but instead directly modified the BM25 ranking model to include anchor text. Dou *et al.* extended this work to account for site-specific hyperlinks [7]. Fujii *et al.* broke anchor texts into terms and assigned weights to those terms according to the weight of each term in the anchor text as well as the weight of each term in the referenced Web page [9]. Chakrabarti *et al.* used the link structure and anchor texts to retrieve Web pages which are authoritative for a topic in their Automatic Resource Compilation system [4]. Lu *et al.* showed that they could extract a live translation dictionary for cross-language IR by considering anchor texts referencing the same Web page as 'parallel texts' [17]. In all of these studies only the anchor text adjacent to the Web page in question is considered, and the various paths of anchor texts are not taken into consideration.

A recent paper by Metzler *et al.* proposed a method that aggregates anchor texts from the Web pages external to the Web site in question to enrich the individual document representations. However, this work aims to overcome anchor text sparsity and therefore "...only adds aggregated anchor text to documents that do not originally have anchor text lines associated with them" [19]. Our method does aggregate anchor texts, although differently, but does not use these anchor texts for information retrieval.

### 3. METHODOLOGY

The overall algorithm, shown in Algorithm 1, should serve as a general framework for the the mapping task. Each particular step is described in some detail in later sections and can be extended or changed by further research.

---

#### Algorithm 1: Mapping Framework

---

```

input : Set of records  $r \in R$ , Selected column  $c \in DB$ ,
        Reference page  $u$ , Destination pages  $v \in V'$ 
output: One to one mapping  $M$ 

foreach  $v \in V'$  do
    Find link paths  $\mathcal{P}_K$  from  $u \rightarrow v$ ; /* Section 4.1 */
    Sorted anchors  $\mathcal{A}_{u,v}$  from  $\mathcal{P}_K$ ; /* Section 4.3 */
    foreach anchor text  $a \in \mathcal{A}_{u,v}$  do
        Find record  $r$  matching  $a$  on column  $c$ ;
        /* Section 5 */
        if match found then
            add  $r - v$  to  $M$ ;
            break;
    return  $M$ ;

```

---

As input to the framework, we require a set of records (*i.e.*, a structured database), a column from that database to match against, a Web page from which link paths originate called a *reference page*, and a set of Web pages to be mapped to records in the database called *destination pages*. These destination pages most commonly refer to a single entity or item that is dually described in the records of the database. Finding these end pages is the topic of previous and ongoing research known as the homepage finding task.

The output of the framework is a one to one mapping of a record to an destination page. In some problem settings a many to one, one to many or many to many mapping may be preferred, but in this initial work we only consider singular mappings. These map-

pings can be used for any number of tasks; we describe some of the potential outcomes resulting from accurate record to Web page mappings in Section 8.

Each line in the above framework can be implemented a variety of ways. The following sections highlight the specific methods we found to be most helpful in generating accurate mappings.

### 4. LINK PATHS

Let  $G = (V, E)$  denote a given graph, where  $V = \{v_1, \dots, v_n\}$  is the set of vertices and  $E = \{e_1, \dots, e_m\} \subset V \times V$  is the set of directed edges, where each edge  $e_k$  can be represented by  $\langle v_i, v_j \rangle$ . A path  $p \in G$  is a sequence of directed edges  $p = \langle \langle v_1, v_2 \rangle, \dots, \langle v_{l-1}, v_l \rangle \rangle$ . In this paper we denote  $u = v_1$  and  $v = v_l$  as the source and destination vertices of path  $p$  respectively. Each edge is associated with a value called the edge cost denoted by  $c_e$ . For our purposes all edge costs are uniform; so for any path  $p$  in  $G$ ,  $c(p) = l$ .

On the Web, each Web page represents a vertex and each hyperlink represents a directed edge. A *link path*, therefore, is a path through the Web-graph from one Web page to another. Specifically, if a path between page  $u$  and page  $v$  contains pages  $x$ , and  $y$  then a link path from  $u$  to  $v$  is  $u \rightarrow x \rightarrow y \rightarrow v$ .

Anchor tags along the link path are extremely important. To capture this information we label each edge in the link path with the corresponding anchor text. If the link between pages  $u$  and  $x$  has the anchor text  $a$  then the link is labeled  $u \xrightarrow{a} x$ .

Because there are an infinite number of possible paths on the World Wide Web, the first step is to identify the source page  $u$  and destination pages  $v \in V'$  where  $V' \subset V$ . The source page  $u$ , known as the *reference page*, provides context to the mapping task and is therefore task dependant. For instance, if the task is to map personal Web pages at the University of Illinois to the structured university phone book then an appropriate reference page would be [www.illinois.edu](http://www.illinois.edu); if the task is to map official movie Web pages to structured IMDB data then an appropriate reference page may be <http://trailers.apple.com>. In any case, the reference pages should be identified either manually or by some heuristic.

Identifying the set of destination pages  $V'$  is similar to the homepage identification task from the 2002 TREC Conference [5] and similar to other work on homepage finding [6, 24]. Otherwise, simple heuristics can be of limited use to extract the destination pages. A separate paper describing our specific solution to finding destination pages is<sup>1</sup>.

Now that the source  $u$  and destination  $V'$  pages have been identified, we turn our attention to the various link paths between  $u$  and each destination page  $v \in V'$ .

#### 4.1 Finding link paths

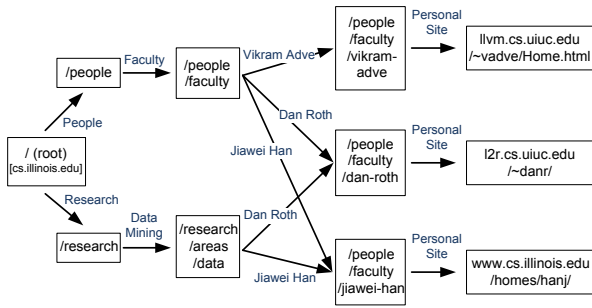
A simple way to find the link path between  $u$  and  $v$  is to perform a shortest path search on the graph. Unfortunately, a single link path may not contain all the information needed to provide an accurate mapping.

We propose two solutions to this problem: (1) find the  $K$ -shortest link paths, and (2) find the  $K$ -shortest loopless link paths. The following subsections discuss each approach.

##### 4.1.1 Repeating with $K$ -shortest paths

Arguably the most intuitive way to find several short example paths is with the  $K$ -shortest paths method. This problem is usually formulated as an optimization problem under certain constraints.

<sup>1</sup>Not cited for anonymity.



**Figure 2: Cropped Web graph from the Computer Science Department at the University of Illinois at Urbana-Champaign**

*Definition 1.* Given a positive integer  $K$ , the  $K$ -shortest paths is the set  $\mathcal{P}_K = \{p_1, \dots, p_K\} \subseteq \mathcal{P}$ , such that:

- $c(p_k) \leq c(p_{k+1})$ , for any  $k \in \{1, \dots, K-1\}$ ;
- $c(p_K) \leq c(p)$ , for any path  $p \in \mathcal{P} - \mathcal{P}_K$ ;
- $p_k$  is determined before  $p_{k+1}$ , for any  $k \in \{1, \dots, K-1\}$ .

That is, not only the shortest path is to be determined, but also the second shortest, the third shortest, and so up to the  $K^{\text{th}}$  shortest path.

Our implementation is a variant of the Bellman-Ford algorithm but instead of only storing the best path it stores the  $K$  best paths at each pass.

**Example.** From the Web graph in Figure 2 we see that by the 2-shortest paths between the reference page `cs.illinois.edu` and the destination page `l2r.cs.uiuc.edu/~danr/` we would explore two paths to Dan Roth’s page:  $p_1 = \{\text{people}, \dots, \text{faculty}, \dots, \text{dan-roth}, \text{l2r.cs.uiuc.edu/~danr}\}$  and  $p_2 = \{\text{research}, \dots, \text{areas/data}, \dots, \text{dan-roth}, \text{l2r.cs.uiuc.edu/~danr}\}$ , and all of the anchor texts would be gathered.

One potential problem with this approach is that we do not know how many short-paths to explore. If we explore too few paths then we risk not gathering enough information. If we explore too many paths then we risk redundant processing.

A second potential problem with the  $K$ -shortest path approach is that Web sites typically contain the same menu structure across member pages. These menus create frequent cycles in the Web graph which can quickly exhaust the  $K$ -shortest paths. As a result, the 1<sup>st</sup> path may contain valuable information and the other paths may only explore cycles in the menu structure and therefore not contribute additional information.

#### 4.1.2 Repeating with $K$ -shortest loopless paths

To alleviate the common problem of menu (and other) cycles interfering with our algorithm we adapt a specific class of the  $K$ -shortest path problem called the  $K$ -shortest loopless path algorithm.

A cycle or loop in  $G$  is a path from one vertex to itself ( $i = j$ ). A path is loopless if it does not have repeated vertices.

*Definition 2.*  $K$ -shortest loopless paths are defined in Definition 1 with the added constraint:

- $p_k$  is loopless, for any  $k \in \{1, \dots, K\}$ .

The  $K$ -shortest loopless paths problem was initially developed by Yen [25], and although several optimized and approximate algo-

gorithms have been developed, our implementation is based on Yen’s original work.

In the example shown in Figure 2 the  $K$ -shortest path and  $K$ -shortest loopless paths are identical. However, in real world, web-scale data the results of these different methods are quite different.

Our intuition is that loopless paths will result in a greater variety of anchor texts than the  $K$ -shortest path method because the loopless paths are not susceptible to the cycles within Web site menus.

In both the  $K$ -shortest and  $K$ -shortest loopless path finding methods we would expect to collect a set of anchor texts along various paths between the reference page  $u$  and the destination page  $v$ .

## 4.2 Anchor Text

The path  $p$  from  $u$  to  $v$  will have edges  $\{e_1, \dots, e_{l-1}\}$  annotated by the anchor text of each edge  $a_e$ . The set of anchor texts  $\{a_{e_1}, \dots, a_{e_{l-1}}\}$  for the path  $p$ , denoted  $A_p$ , typically contains a descriptive text relative to the reference page  $u$  of the destination page  $v$ .

**Example.** Continuing the above example, the anchor texts retrieved from Figure 2 with the  $K$ -shortest path algorithm are:  $A_{p_1} = \{\text{People}, \text{Faculty}, \text{Dan Roth}, \text{Personal Site}\}$  and  $A_{p_2} = \{\text{Research}, \text{Data Mining}, \text{Dan Roth}, \text{Personal Site}\}$ .

From this real world example, we see the utility of anchor texts from link paths because Dan Roth is a *person* and a *faculty* member who does *research* in *data mining*.

Next the  $K$  link paths  $\{p_1 \dots p_K\}$  are combined into a bag-of-words representation  $\{A_{p_1}, \dots, A_{p_K}\} \in \mathcal{A}_{u,v}$  for each of the  $K$  link paths between  $u$  and  $v$ . We refer to  $\mathcal{A}_{u,v}$  as a bag-of-anchors.

**Example.** The bag-of-anchors from the running example is  $\{\text{Research}:1, \text{People}:1, \text{Faculty}:1, \text{Data Mining}:1, \text{Dan Roth}:2, \text{Personal Site}:2\}$ .

## 4.3 Aggregating Link Paths

As discussed earlier, there will be many destination pages in each Web site. Therefore, each destination page  $v \in V'$  will have its own set of  $K$  link paths and its own bag of anchors  $\mathcal{A}_{u,v}$  resulting in  $|V'|$  bags  $\{\mathcal{A}_{u,v_1}, \dots, \mathcal{A}_{u,v_{|V'|}}\}$ .

**Example.** In the full example from Figure 2 there exist three destination pages, and therefore three bags:

$$\mathcal{A}_{u,v_1} = \{\text{Research}:1, \text{People}:1, \text{Faculty}:1, \text{Data Mining}:1, \text{Dan Roth}:2, \text{Personal Site}:2\}$$

$$\mathcal{A}_{u,v_2} = \{\text{Research}:1, \text{People}:1, \text{Faculty}:1, \text{Data Mining}:1, \text{Jiawei Han}:2, \text{Personal Site}:2\}$$

$$\mathcal{A}_{u,v_3} = \{\text{People}:1, \text{Faculty}:1, \text{Vikram Adve}:1, \text{Personal Site}:1\}$$

The next step is to rank the texts within each bag of anchors so that more descriptive anchor texts are given a higher ranking. To do this, we normalize the score of each word  $a$  of the  $i^{\text{th}}$  bag by

$$\frac{f(a \in \mathcal{A}_{u,v_i})}{\sum_{j=1}^{|V'|} f(a \in \mathcal{A}_{u,v_j})}$$

where  $f(a \in \mathcal{A})$  is the frequency of anchor  $a$  in the bag of anchors  $\mathcal{A}$ . Put more simply, we normalize each word by the frequency each word occurs in an bag of anchors (term frequency) over the frequency of each term in all bags (global term frequency). Finally, we sort the bag of anchors in descending order.

**Example.** Continuing the example above, the ranked bags of anchors are:

$$\mathcal{A}_{u,v_1} = \{\text{Dan Roth}:2/2=1, \text{Research}:1/2=0.5, \text{Data Mining}:1/2=0.5, \text{Personal Site}:2/5=0.4, \text{People}:1/3=0.33, \text{Faculty}:1/3=0.33\}$$

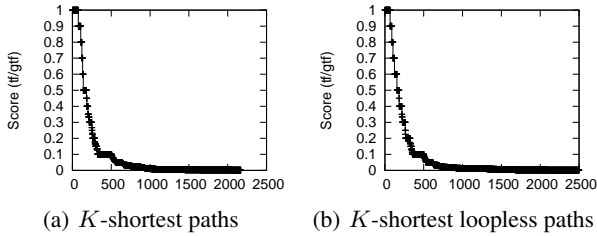


Figure 3: Distribution of scores for all final link paths

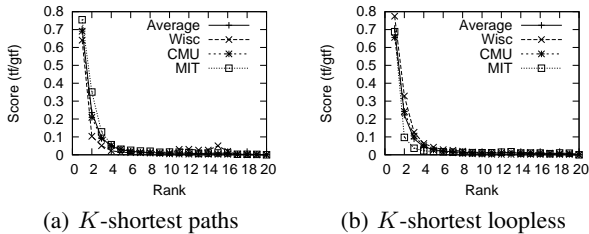


Figure 4: Mean scores per rank

$\mathcal{A}_{u,v_2} = \{ \text{Jiawei Han}: 2/2=1, \text{Research}: 1/2=0.5, \text{Data Mining}: 1/2=0.5, \text{Personal Site}: 2/5=0.4, \text{People}: 1/3=0.33, \text{Faculty}: 1/3=0.33 \}$   
 $\mathcal{A}_{u,v_2} = \{ \text{Vikram Adve}: 1/1=1, \text{People}: 1/3=0.33, \text{Faculty}: 1/3=0.33, \text{Personal Site}: 1/5=0.2 \}$

Thus, we find that the most descriptive terms for each destination page are ranked highest in each list. We especially notice that the anchor text nearest the destination page is not necessarily the most descriptive.

The small example from Figure 2 does not necessarily show the real world distribution of the scores. Figure 3 shows the actual distribution of scores from three randomly chosen computer science departments’ Web sites (CMU, Wisconsin and MIT). These graphs show that, overall, a relative few anchor texts are highly descriptive. From this data we also find that the mean and median score for the  $K$ -shortest path method is .103 and .007 respectively, and the mean and median score for the  $K$ -shortest loopless path method is .084 and .010 respectively. The  $K$ -shortest path method found 2,168 total hyperlinks and the  $K$ -shortest loopless path method found 2,596 total hyperlinks.

These statistics confirm our intuition that the  $K$ -shortest loopless path method found a greater variety of anchor texts than the  $K$ -shortest path method. But as a result of this variety the average descriptive value of each text is lower than the  $K$ -shortest path method.

Figure 4 shows the mean scores for each rank, that is, the average score for anchor texts ranked first, the average score for anchor texts ranked second, etc. Although the distribution is not guaranteed to be monotonically decreasing, the central principle here is that the descriptive value of each rank trails off quickly, after about 4 or 5 in this dataset. We use this information in the next section to perform the mapping.

## 5. MAPPING WEB PAGES TO DATABASE RECORDS

The ranked link path information described in the previous section can be used for many purposes including Web search and information retrieval, and while the IR task may be beneficial it is not

the goal of this work. Instead, our particular goal is to use the texts encoded in the link path to map the destination Web page ( $v \in V'$ ) to its corresponding record in a structured database  $r \in R$ .

To rephrase, given a set of structured database records, we wish to add a new column to the schema labeled URL and populate the new cells of the record with URLs of the corresponding Web pages. This task was described previously in the example from Figure 1 wherein a list of names from, say, DBLP or a phonebook is extended with URL information. The example continues to show the potential benefits of this mapping by showing that extra information from the Web page can be extracted and added to the schema of the database.

If we consider the homepage URL and link paths to be a structured schema itself then we could employ any number of information integration methods which attempt to reconcile two or more schemas. However, these information integration approaches are typically complicated and require a lot of computational effort. Fortunately, link paths allow for a more straightforward approach. This is because links paths typically contain canonical texts (as shown in related work [13, 8]) that can be matched with text from the database. All that is needed ahead of time is some indication from the user as to which database columns to match against. For example, if the task is to match faculty Web pages to the DBLP database then the user should indicate the Name column because professors’ names are likely to be present on the link paths. Of course, it is not absolutely necessary to pick a specific column but it does help to narrow the search because otherwise each column of each record would need to be searched.

Very frequently the text on a link path is not exactly the same as corresponding text in the database. Names, especially, can be represented in several different ways. For example, a persons name can be represented with or without the middle name, with the middle name abbreviated, last name first, and so on. Therefore, an exact byte-by-byte query would rarely return any results. To mitigate this problem, before a query is actually performed, the anchor text is sanitized, that is, all punctuation and extra spaces are removed and all letters are lowercased.

The actual retrieval function should collect records which match terms from the query string, otherwise the ordering of terms would matter (*e.g.*, ‘Dan Roth’ would not match ‘Roth, Dan’). Most database systems have an indexing or search function to handle these types of queries; we use MySQL and its `match` against function to retrieve records as described in the next subsection.

### 5.1 Search function

Searching the selected columns for matches is a straightforward task. For each sorted bag of anchors the database is queried with the top ranked anchor text as the search term. The major difference between byte-by-byte search and our search function is that a full text search on the name ‘Roth, Dan’ would return ‘Dan Roth’s record as well as anyone containing the name ‘Dan’ or ‘Roth’. This will likely return at least one result. Unfortunately, there exist 135 Roth’s in DBLP, and ‘Personal Site’ also retrieves a record for ‘Luigi Delle Site’.

Obviously, in approximate matching it is necessary to pick a record which most closely resembles the original query string. For this task we use a word alignment algorithm shown in Algorithm 2, which is similar to, but not to be confused with, the character alignment or edit distance algorithm, which attempts to find the string with the fewest *word* differences. Like traditional edit distance algorithms, lower scores denote a closer distance; therefore, lower scores are better.

---

**Algorithm 2:** Search Function

---

**input** : Sorted Bag of Anchors  $\mathcal{A}_{u,v}$ , Set of records  $R$   
**output**: Best Match  $best$

```
foreach anchor text  $a \in \mathcal{A}_{u,v}$  do
   $R' \leftarrow$  fulltext query on database  $R$  with search term  $a$ ;
  foreach record  $r \in R'$  do
     $s \leftarrow r_{text}$ ; /* text from selected column */
     $a \leftarrow$  remove punctuation and lowercase  $a$ ;
     $t \leftarrow a_{len}$ ;
    if  $s_{len} > a_{len}$  then
      Swap( $s, a$ );
       $t \leftarrow a_{len}$ ; /*  $a$  must be longer than  $s$  */
     $S \leftarrow$  split  $s$  on whitespace;
    for  $i \leftarrow 1$  to  $|S|$  do
      if  $a$  contains  $S_i$  then
        remove  $S_i$  from  $a$ ; /* remove match */
     $res \leftarrow a_{len}/t$ ; /* percentage not matched */
     $best \leftarrow \text{Min}(best, res)$ ; /* fewest diffs */
    if  $best = 0.0$  then
      return  $best$ ; /* perfect match */
return  $best$ ;
```

---

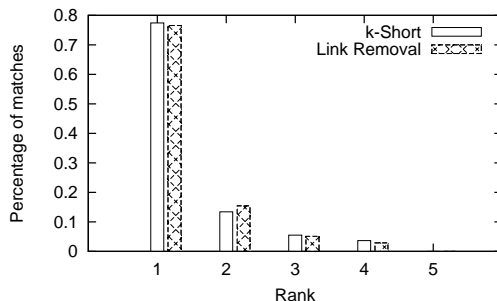
To illustrate the execution of the approximate matching algorithm let there be a sorted link path  $\{\textit{Personal Site}, \textit{Roth Dan}, \textit{Faculty}\}$  – capitalization is maintained for clarity. We first query the database with ‘Personal Site’ and retrieve the record of ‘Luigi Delle Site’. To begin the matching algorithm, we split the shorter string ‘Personal Site’ into two words ‘Personal’ and ‘Site’. The word ‘Personal’ is removed from the string ‘Luigi Delle Site’ resulting in no change; then the word ‘Site’ is removed from the string ‘Luigi Delle Site’ resulting in ‘Luigi Delle’. The score of this matching is  $11/16=.69$ .

Next, we query the database with ‘Roth Dan’ and retrieve two records (in this short example), ‘Dan Andresen’ and ‘Dan Roth’. Starting with ‘Dan Andresen’, we split the shorter string ‘Roth Dan’ into two words ‘Roth’ and ‘Dan’. The word ‘Roth’ is removed from the string ‘Dan Andresen’ resulting in no change, and the word ‘Dan’ is removed from the string ‘Dan Andresen’ resulting in ‘Andresen’. The score of this matching is  $8/12=.66$ , and the best matching so far is ‘Dan Andresen’. The second record ‘Dan Roth’ is split into ‘Dan’ and ‘Roth’. The word ‘Dan’ is removed from ‘Roth Dan’ resulting in only ‘Roth’ remaining, and the word ‘Roth’ is removed from the string ‘Roth’ resulting in an empty string. The score of this matching is  $0/8=0.0$ , a perfect match so the algorithm stops.

In this way, we can find perfect matches even though the strings are not identical. Furthermore, approximate matches are given scores based on how closely matched they are so that scores above some threshold do not result in a mapping. If, after iterating through the entire link path, an exact match is not found then the best match is used for the mapping.

Searching through the link path in descending order is particularly important because the most descriptive anchor texts will appear at the top of the list, and once a match is found there is no need to continue searching. In fact, we find that it is very likely that the top ranked anchor text will contain a match to the database.

Figure 5 shows the percentage of matches for each rank in the link path. We see that that the vast majority of matches are with top-ranked texts. Note that this figure does not represent the precision or recall of our method since the figure only shows the rank that a positive match comes from, not the ratio of matches to misses.



**Figure 5: Percentage of database matches for each rank in link paths from Wisconsin, CMU, MIT’s computer science departmental Web sites**

Section 6 contains an evaluation of the efficacy of our method.

Another important product of the ranked link paths is its use in limiting false positives. In the likely event that the link path contains two or more exact database matches the lower-ranked match will be ignored because the algorithm stops after the first exact match. We observed that in some of our testing domains one or two names frequently appeared on the link paths in addition to the actual, correct name. Fortunately, by searching in a sorted order we reduce the occurrence of false positives because the correct match is likely ranked higher than an incorrect match. In the unlikely event that the correct match is ranked lower than an incorrect match then our method would produce an incorrect mapping.

One potential pitfall of this method is the irreconcilability of misspellings. We realize that misspellings happen, but they will not harm the overall algorithm unless the names are all misspelled in the exact same way. This is because misspellings are not likely to be highly ranked in the sorted link path. Therefore we only need to assume that the number of correctly spelled occurrences outnumber the most frequent misspelling.

## 5.2 Disambiguation

In some domains there will be two or more matches with the same score. In these instances, a tiebreaker is needed to choose between the ambiguous matches otherwise a one-to-many mapping is needed. In our specific problem setting we only consider a single column so disambiguation is difficult because there is no tiebreaker information to consider. If we possessed, for example, not only the list of names from DBLP, but also the abstracts for each author, then some simple extensions to basic approach can be tried. A topic model, for instance, based on the abstracts could be used in concert with the link paths’ texts as a tiebreaker. In our experiments, we only allow one-to-one mappings and therefore do not address this problem because, for example, only 0.07% of records in DBLP share the same author name.

## 5.3 Achieving Strict and Approximate Matching with a Threshold

The search function lends itself to multiple variants of the same algorithm based on the adaptation of a threshold on the word distance. This threshold  $\lambda$  does not allow a mapping to occur when the word distance is above the threshold. Because the word distance threshold is guaranteed to be between 0 and 1 inclusive,  $0 \leq \lambda \leq 1$ . For our purposes we examine the two possible extremes of  $\lambda$ : strict matching ( $\lambda = 0$ ), and approximate matching ( $\lambda = 1$ ).

In strict matching we map a Web page to a database record if and only if an exact match is found, that is, when the word distance is

0. We hypothesize that the strict matching will result in a high precision and relatively low recall. The recall should be low because this is a strict condition, and there may not be many of these types of matches; however, the precision should be high because the few mappings which do happen should be accurate.

In approximate matching we map a Web page to the database record with the closest word distance. We hypothesize that the approximate matching will result in a slightly lower precision and perfect recall. The recall should be perfect because the closest word will always be mapped to the database record; however, the precision should be lower because more matches provides a larger room for error.

The next section shows how the precision and recall are specifically affected by the threshold.

## 6. EXPERIMENTS

We evaluated the effectiveness of our algorithm using two data sets, and we compare the performance of our methods against two worthy baseline systems. Despite the large number of studies related to this work, to the best of our knowledge there do not exist other methods for mapping Web pages to structured records. Therefore, we cannot directly compare our algorithm to other published methods. Nevertheless, this section contains results of tests to determine the effectiveness of the algorithm described in this paper.

### 6.1 Data sets

The first data set is a crawl of the departmental Web sites of the top 25 American computer science graduate schools<sup>2</sup>. The crawl for this dataset was a breadth first crawl from each department’s homepage (e.g., <http://cs.illinois.edu>) depth limited to a distance of 4 pages from the homepage. This crawl resulted in 1,970,691 distinct hyperlinks across 123,157 distinct Web pages. From these Web pages, the personal homepages of 1,137 computer science faculty members were found via a homepage finding algorithm, described in Sections 2, and manually verified.

The task for this first data set is to map each of the 1,137 faculty homepages to their corresponding record in DBLP using only information from their link paths. Assuming that all computer science faculty members are listed in DBLP is a faulty assumption. Unlisted cases are true negatives that add to the difficulty of the problem and can be represented in the results; for example, returning “not found” for an unlisted faculty member is a positive result.

The second data set is a crawl of movie Web pages starting from <http://trailers.apple.com> and <http://www.allmovie.com>. The crawl for this dataset was a breadth first crawl which terminated when the domains were exhausted. We expect that many official movie Web pages would be included in this crawl. For example, <http://trailers.apple.com> points to the Robin Hood trailer which has a link to the official Robin Hood Web page <http://robinhoodthemovie.com/>, as well as many other potential link paths. This crawl resulted in 110,666 distinct hyperlinks across 8,862 distinct Web pages from Apple and 123,596 distinct hyperlinks across 32,258 distinct Web pages from AllMovies.com. From these Web pages 345 official movie Web pages were found with the homepage finding algorithm and manually verified.

The task for the second data set is to map each of the movie homepages to their corresponding records in the Internet Movie Database (IMDB). In this case, assuming that each of these movies will be listed in IMDB is a valid assumption because of their relative popularity and because of the wide coverage of IMDB.

### 6.2 Baselines

One of the goals of this section is to compare the expressive power of link paths to the current method which uses only adjacent links. Therefore, for the first baseline we query Yahoo’s Site Explorer Service – which returns inbound links to the query page – with each destination Web page  $v \in V'$ , and we retrieve the first 10 results and extract the corresponding anchor text from the referencing Web page. The resulting 10 anchor texts are ordered by their frequency and mapped to the structured database with the approximate matching method. The results of this baseline will allow for a comparison between adjacent links and link paths.

The second baseline maps the result of a search engine query to the corresponding record in the database. Popular search engines like Google, Yahoo and Bing also use incoming anchor text in their retrieval algorithms. However, like most of the related work, these search engines only consider the anchor texts which are directly adjacent to each Web page, but they also include title text, content, hyperlinks, etc. for retrieval. Google was chosen because its results were easiest to parse; ancillary experiments showed that Yahoo and Bing typically retrieved the same top result as Google.

The homepage finding task from the 2002 TREC Web Track competition would serve as a good comparison. Unfortunately, the typical description of TREC competitors is not detailed enough for re-implementation and we were unable to find any existing implementations of their approaches. The top ranked results of the TREC competition received a mean reciprocal rank (MRR) of about 70% which is comparable to our chosen baseline.

### 6.3 Setup

For the computer science data set we set  $K = 10$ . 10 was chosen empirically; with the  $K$ -shortest loopless path algorithm. The link paths were found between the department’s homepage and each faculty members’ personal Web pages using the  $K$ -shortest path and  $K$ -shortest loopless path methods. The paths were combined and ranked as described in Section 4 and mapped to DBLP’s author column using the strict and approximate matching methods from Section 5.

For the movie data set we also set  $k = 10$ . The link paths were found between <http://trailers.apple.com> and <http://www.allmovie.com> and each of the 345 movie Web pages using the  $K$ -shortest path and  $K$ -shortest loopless path methods. The paths were combined, ranked and mapped to the IMDB title column using the strict and approximate matching methods.

#### 6.3.1 Metrics

We judge the effectiveness of each method by standard precision and recall methods. Note that the recall metric is only important in the strict matching experiments because mappings are not made where there is not an exact match. In the approximate matching experiments recall is always 100% because the algorithm will always make an attempt resulting in 0 null mappings. Mean reciprocal rank (MRR) scores for the search engine results are shown so that these results can be implicitly compared to the TREC results. In all results the final mean scores are based on total sums and counts, not an average of averages.

## 7. RESULTS

The Web page to DBLP mapping task can be broken down into the individual departments in order to investigate the efficacy of our method in detail. Although we tested the algorithm on 25 computer science departments, in Tables 1 and 2 only the worst performing departmental Web sites are specifically shown, the remainder are

<sup>2</sup>Rankings from US News 2010

**Table 1: Web page to DBLP baseline mapping results sorted and grouped to match Table 2**

	Baseline		
	Google MRR	Google Prec.@1	Adjacent Prec.
Top 7	.7017	67.40	72.55
Middle 15	.7733	73.24	74.62
UCLA	.7726	73.21	78.57
UMass	.7721	75.51	81.63
UPenn	.7757	73.68	65.78
Mean	.7516	71.58	74.19
Median	.7757	74.60	77.35

**Table 2: Web page to DBLP mapping results sorted and grouped by  $\approx$  Mapping**

	Strict Matching				$\approx$ Matching Loopless Prec.
	$K$ -short		Loopless		
	Prec.	Recall	Prec.	Recall	
Top 7	100	63.64	100	93.95	100
Middle 15	97.54	51.33	98.04	53.51	94.42
UCLA	100	28.57	100	50.00	82.14
UMass	100	28.41	100	28.57	81.63
UPenn	100	30.61	85.71	47.37	65.78
Mean	96.34	52.86	98.63	54.49	94.32
Median	100	56.52	100	57.89	97.83

averaged for brevity. The mean and median of the scores are shown at the bottom of both tables.

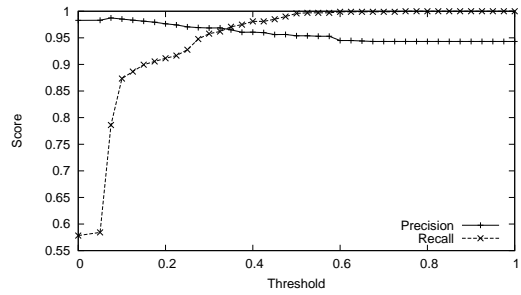
Table 1 shows the results of the baseline algorithms. As we alluded to earlier, we find that the MRR results from the Google baseline are comparable to (and in some instances better than) the results from the TREC results. Based on this observation we believe that the precision at 1 is a good baseline for the mapping task.

Table 2 shows the results for the variations of our algorithm. The first four columns show the results for the strict matching method on both the  $K$ -shortest path and  $K$ -shortest loopless path algorithms. These results show that, even under the strict matching conditions, the mappings are very precise but have low recall. Moreover, the overall accuracy under strict conditions confirms our assertion that anchor texts often represent succinct, canonical representations of the referenced Web page.

Table 2 is sorted and grouped by the approximate mapping results in descending order, and the top seven results were grouped together because they all achieved 100% precision. Of the results from the strict matching we find that the  $K$ -shortest loopless paths approach performed the better than the  $K$ -shortest path algorithm, and because manually verifying each matching takes a considerable amount of time we decided to only test the approximate matching algorithm with the  $K$ -shortest loopless paths method. cursory experiments show that the  $K$ -shortest path method achieves similar, but lower, performance.

Comparing the baseline results from Table 1 to our results from Table 2, we see a 36.03% increase in precision over the Google baseline and a 31.24% increase in precision over the adjacent only approximate matching baseline.

In terms of recall, our algorithm will always return a result when there is no threshold to limit the string distance. For tasks which require a higher level of precision a lower threshold ( $\lambda$ ) may be appropriate. Figure 6 shows how the precision and recall of the DBLP data set fluctuate as  $\lambda$  varies between 0 and 1.



**Figure 6: Precision and Recall tradeoff for DBLP data with  $K$ -shortest loopless paths as  $\lambda$  varies from 0 to 1.**

**Table 3: Web page to IMDB baseline mapping results**

	Baseline		
	Google MRR	Google Prec@1	Adjacent Prec.
Apple	.4082	20.61	38.17
AllMovie	.5053	41.18	43.14
Mean	.4240	23.96	38.97

The graph in Figure 6 confirms our assertion that lower thresholds will result in high precision and lower recall. We also see that at its worst the recall is relatively low, but the lowest precision score is not too low. For our purposes we prefer  $\lambda = 1$  because the small drop in precision is worth the large gain in recall.

We specifically call attention to the lowest three results in the spirit of full disclosure and in order to address weaknesses of this approach. UCLA and UMass received low scores mainly because sometimes faculty members on the link path were matched ahead of the correct faculty member. This is usually avoided because the correct name should appear more frequently than other names, and therefore be ranked higher, but we see in these results that this assumption does not always hold.

The UPenn result ranked the lowest for a different reason. In the UPenn computer science departmental Web site the faculty names are typically not on the link path; instead, “web page” is the anchor text which links to the personal homepage of a faculty member. Fortunately, many of the link paths traverse through news stories and other Web pages that contain enough appropriate anchor text to make some mappings possible.

The IMDB data set performed similarly to the DBLP data, and these complimentary results offer reinforcement to the validity of our approach. Table 3 shows the baseline results of mapping the structured IMDB database to an official movie homepage. Baseline results show that this is a much more difficult task than the DBLP data set. The results from Google were low because gossip, biography, etc. Web pages were frequently ranked higher than the official movie Web page. Using only adjacent anchor texts also performed poorly because, according to our observation, most of the links to the movie Web page were images, which do not contain anchor text; a finding similar to that of Metzler *et al.* [19].

The results for the IMDB task shown in Figure 4 show that the strict matching also resulted in a low precision and a high recall. For Apple specifically, the recall was particularly low because many movie titles in this data set were shortened to fit on the Web page making strict matching difficult. Data from AllMovie.com has a higher recall under strict conditions because it does not shorten movie titles.

Of the results from the strict matching we find that the  $K$ -shortest

**Table 4: Web page to IMDB mapping results with Exact Matching method**

	Strict Matching				$\approx$ Matching Loopless Prec.
	$K$ -short		Loopless		
	Prec.	Recall	Prec.	Recall	
Apple	100	35.16	100	37.41	70.99
AllMovie	98.03	50.98	98.03	64.71	76.47
Mean/Med	99.69	37.70	99.69	41.85	71.86

loopless paths method approach again performed better than the  $K$ -shortest path algorithm, and therefore we only test the approximate matching algorithm with the  $K$ -shortest loopless paths method.

When the matching requirements are relaxed we see from Table 4 that the performance improves in both Apple and AllMovie data sets. Comparing the baseline results from Table 3 to our results from Table 4, we see that our approach triples the precision of the adjacent only baseline and results in an 84.40% increase in precision over the Google baseline.

The precision and recall tradeoff is not shown for the IMDB data, but we believe that the curves would maintain a shape similar to the DBLP data from Figure 6.

## 7.1 Discussion

Overall, the tables above show promising results for the difficult task of mapping Web pages to their appropriate record in a structured database. We remind the reader that our algorithm uses *only* data from anchor texts and does not rely on the titles or content of the referenced Web page. Undoubtedly, other information sources could be used to further enhance this task, but we leave that for future research. It is also important to note that the Web pages used in these experiments come from many different sources with a variety of styles and templates. So the traditional wrapper generation approach would not be effective at this task.

Generally, the results show that the task of mapping faculty members to their DBLP record is more precise than the movie data set. This is because the ambiguity among names of computer scientists is (perhaps purposefully) quite low, and a link path to one faculty member typically does not contain the name of another faculty member. On the other hand, the ambiguity among movie titles is quite large and the link paths frequently contained more than one movie title.

Although we focused on the bad results in order to highlight the shortcomings of our approach, the overall performance is actually quite promising. Having achieved 94% and 72% average precision with 100% recall, we believe this is a good first step towards the ultimate goal of automatically mapping Web pages to database records.

## 8. CONCLUSIONS

In conclusion, this paper proposes a new information management task: the automatic mapping of Web pages to their corresponding records in a structured database. We show that sorted link paths can be used to achieve this mapping, and we perform experiments on two large real-world data sets that show the effectiveness of our algorithm. Finally, we demonstrate that our approach is able to generate an accurate mapping across two large and varied data sets. Furthermore, we believe that our method does generalize to other data sets, and we encourage the community to explore this approach on more data sets.

Linking unstructured data with the records in structured database has become a popular task because researchers have recognized that



**Figure 7: Automatically generated map of 25 computer science departments**

there is a greater use for information when it is available in a structured or otherwise organized form. Yet most of the current work has revolved around extracting records from a Web page or ranking documents for retrieval. While these are important tasks, we argue that there is a greater need for a framework which employs structured data to enhance unstructured data, and conversely, employs unstructured data to improve the expressiveness of structured data. We believe that the mapping algorithm proposed in this paper is a promising step towards the development such a framework.

## 8.1 Motivation Revisited

To finally show the overall purpose for mapping Web pages to structured database records we return to the motivating example from earlier where we implied that with an acceptable mapping it would be possible to extend a database schema to include the zipcodes of faculty members in order to place the persons on a map. By extracting all 5 digit numbers from the Web pages and treating the 25 most frequent numbers as zipcodes we are able to visualize the geographical distribution of the faculty members as shown in Figure 7. Note that the specific colors are byproducts of the map API and are not significant.

Of course, extracting zipcodes is an easy task and not much can be gained from the zipcode task alone. However, this example shows the potential of this line of research. A more difficult, and more interesting, task would be the mapping, clustering and ranking of persons (or other entities) according to information from within structured and unstructured sources.

## 8.2 Future Work

As mentioned throughout this paper, the task of mapping Web pages to database records is not an end in itself, but rather an intermediary step to a number of other tasks. For instance, unstructured data (from a Web page) can be used to aid in the retrieval of semantically related database records, or to extend the database schema, etc. Alternatively, structured data can be used in unstructured tasks in order to categorize search results or improve ranking algorithms, etc.

In terms of the specific link path method, we do not argue that our approach is optimal. There may indeed exist extensions to our method that improve the mapping performance. For example, by modifying the link path weighting algorithm to apply a higher weight to anchor texts closer to the destination page may be more effective than our ranking approach.

Otherwise, the use of title text, URL text, and page content information may be used to improve the overall accuracy of the mapping algorithm. We intend to continue this line of research, and expect to

develop a mutually enhancing Web-database system based on this framework in the near future.

## 9. ACKNOWLEDGMENTS

We would like to thank Name Redacted for her help on this project. This work is made possible by an NDSEG Fellowship to the first author, and the second author is supported by both the project “Knowledge Discovery in Relation Domains” funded by the University of Name Redacted and the Strategic Project DIPIS (Distributed Production as Innovative Systems) funded by Name Redacted.

## 10. REFERENCES

- [1] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
- [2] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, 2008.
- [3] M. J. Cafarella, J. Madhavan, and A. Halevy. Web-scale extraction of structured data. *SIGMOD Rec.*, 37(4):55–61, 2008.
- [4] S. Chakrabarti, B. Dom, P. Raghavan, S. R. D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *WWW7: Proceedings of the seventh international conference on World Wide Web 7*, pages 65–74, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.
- [5] N. Craswell and D. Hawking. Overview of the trec-2002 web track. In *TREC '02: In Proceedings of the eleventh text retrieval conference TREC-2002*, pages 86–95. NIST, 2003.
- [6] N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 250–257, New York, NY, USA, 2001. ACM.
- [7] Z. Dou, R. Song, J.-Y. Nie, and J.-R. Wen. Using anchor texts with their hyperlink structure for web search. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 227–234, New York, NY, USA, 2009. ACM.
- [8] N. Eiron and K. S. McCurley. Analysis of anchor text for web search. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 459–460, New York, NY, USA, 2003. ACM.
- [9] A. Fujii. Modeling anchor text and classifying queries to enhance web document retrieval. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 337–346, New York, NY, USA, 2008. ACM.
- [10] A. Fujii, K. Itou, T. Akiba, and T. Ishikawa. Exploiting anchor text for the navigationalweb retrieval at ntcir-5. In *NTCIR-5 Workshop Meeting*, 2005.
- [11] V. Harmandas, M. Sanderson, and M. D. Dunlop. Image retrieval by hypertext links. *SIGIR Forum*, 31(SI):296–303, 1997.
- [12] E. H. Hovy. *Natural Language Processing and Information Systems*, chapter 1, pages 1–7. Springer Berlin / Heidelberg, 2010.
- [13] R. Jin, A. G. Hauptmann, and C. X. Zhai. Title language model for information retrieval. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–48, New York, NY, USA, 2002. ACM.
- [14] R. Kraft and J. Zien. Mining anchor text for query refinement. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 666–674, New York, NY, USA, 2004. ACM.
- [15] C. X. Lin, B. Zhao, T. Weninger, J. Han, and B. Liu. Entity relation discovery from webtables and links. In *WWW*. ACM, April 2010.
- [16] B. Liu. *Web Data Mining – Exploring Hyperlinks, Contents and Usage Data*. Springer, 2006.
- [17] W.-H. Lu, L.-F. Chien, and H.-J. Lee. Anchor text mining for translation of web queries: A transitive translation approach. *ACM Trans. Inf. Syst.*, 22(2):242–269, 2004.
- [18] O. A. McBryan. Genvl and www: tools for taming the web. In *WWW1: Proceedings of the 15th international conference on World Wide Web*, 1994.
- [19] D. Metzler, J. Novak, H. Cui, and S. Reddy. Building enriched document representations using aggregated anchor text. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 219–226, New York, NY, USA, 2009. ACM.
- [20] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser. Extracting data records from the web using tag path clustering. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 981–990, New York, NY, USA, 2009. ACM.
- [21] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.
- [22] S. E. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA, 2004. ACM.
- [23] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. A comparison of implicit and explicit links for web page classification. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 643–650, New York, NY, USA, 2006. ACM.
- [24] W. Xi, E. A. Fox, R. P. Tan, and J. Shu. Machine learning approach for homepage finding task. In *SPIRE 2002: Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, pages 145–159, London, UK, 2002. Springer-Verlag.
- [25] Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(1):712–716, 1971.
- [26] Y. Zhai and B. Liu. Structured data extraction from the web based on partial tree alignment. *IEEE Trans. on Knowl. and Data Eng.*, 18(12):1614–1628, 2006.