Exploiting Content Redundancy for Web Information Extraction

Pankaj Gulhane, Rajeev Rastogi, Srinivasan H Sengamedu, and Ashwin Tengli
Yahoo Labs, Bangalore, India
{pankajeg, rrastogi, shs, ashwint}@yahoo-inc.com

ABSTRACT

We propose a novel extraction approach that exploits *content redundancy* on the web to extract structured data from *template-based* web sites. We start by populating a seed database with records extracted from a few initial sites. We then identify values within the pages of each new site that match attribute values contained in the seed set of records. To filter out noisy attribute value matches, we exploit the fact that attribute values occur at fixed positions within template-based sites. We develop an efficient Apriori-style algorithm to systematically enumerate attribute position configurations with sufficient matching values across pages. Finally, we conduct an extensive experimental study with real-life web data to demonstrate the effectiveness of our extraction approach.

Categories and Subject Descriptors: I.7.m [Computing Methodologies]: Document and Text Processing - Miscellaneous

General Terms: Algorithms, Design

Keywords: Content redundancy, information extraction

1. INTRODUCTION

A significant fraction of pages on the web are dynamically generated by populating *fixed page templates* with content from a backend DBMS. In this paper, we address the following extraction problem: *Given a web site W and a set $A = \{a_1, \ldots, a_q\}$ of entity attributes, extract from each page of W, a single record containing the attribute values for the corresponding entity.*

In recent years, a number of research papers have studied the problem of extracting structured data from web pages. Early web information extraction methods were based on *wrapper induction*. Since human editors are required to annotate pages from each new site, wrapper induction can be labor-intensive and expensive. More recently, there has been a growing interest in using machine learning models for web information extraction. Learning good models can be quite challenging because of page structure and content format variations across sites and because of noise. In this paper, we present a technique for high-precision extraction that addresses the above issues and also leverages content redundancy across the web to reduce training data requirements.

2. EXPLOITING CONTENT REDUNDANCY

We restrict our attention to extracting "string-valued" attributes from "single-entity" pages. Our extraction solution leverages the following two key properties of template-based sites: (1) Multiple sites contain pages for the same entity. Further, the values of an attribute across the various pages for an entity are "textually" similar. (2) Pages within a web site have a similar structure conforming to a common template.

Our extraction procedure starts by populating a seed database R of records from a few initial sites. These records are extracted from the sites by having human editors annotate attribute values in a few sample pages from each site, and learning wrappers for the sites. Note that each seed record in R contains attribute values for an entity from a single entity page. Now, a new site W will typically contain pages for entities that already have records in the seed database. Hence, we scan the pages in W to find values that match attribute values in the seed records. These matching attribute values within the pages of overlapping entities are used to learn wrappers that are subsequently used to extract records from the remaining pages of the new site.

Note that our extraction approach is largely *unsupervised*. Except for a few initial sites where human annotations are needed to generate the seed database, it eliminates the need for manually annotating attribute values in pages of web sites. Moreover, by continuously expanding the initial seed database with extracted records for new entities, it ensures that there is sufficient overlap between the seed database and new sites. Finally, since our approach matches the actual content of attribute values, it can handle variations in web page structure and attribute content formats (e.g., zip codes across countries) better than the machine learning models.

Our extraction approach faces two major challenges. First, it relies heavily on the fact that attribute values across pages for the same real-world entity are textually similar. However, in practice, entity attribute values can vary between sites due to data entry errors, abbreviations, heterogeneous representations, etc. This can lead to attribute value matches going undetected during extraction. Second, web pages are inherently noisy and contain extraneous values that can lead to spurious attribute value matches (e.g., between a restaurant name in the seed database and the same name in the "Nearby Restaurants" section of a web page). The next section discusses the use of approximate string similarity and template structure of web pages for filtering out noisy matches.

3. EXTRACTION ALGORITHM

A basic primitive for our extraction task is to be able to find, for values in a web page, matching attribute values in the the seed database. For our web extraction scenario, we require a similarity metric that is robust to typographical errors, word re-orderings, abbreviations, etc. We use a variant of the "Cosine similarity over q-grams" similarity function [1]. We also associate a weight with each q-gram based on the importance of the word that it originates

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

Algorithm 1 FINDATTRPOS

```
Input: Seed relation $R$, Web site $W$;
Output: Maximal set of (attribute, position) pairs with support $\geq \beta$;
$WS(a, x) = \emptyset$;
for each node $n$ (at position $x$) in each page $p$ of $W$ do
    for each seed record, attribute pair $(r, a)$ such that $sim_a(r[a], p[x]) \geq T_w$ do
        $WS(a, x) = WS(a, x) \cup (r, p)$;
    end for
end for
$C = \{\{(a, x)\} : WS(a, x)$ contains at least $\beta$ distinct pages $p\}$;
$C_1 = C$; $k = 1$; $S_{max} = \emptyset$;
while $C_k \neq \emptyset$ do
    for each set $S \in C_k$ do
        $sup(S)$ = number of distinct pages $p$ in $\cap_{(a,x) \in S} WS(a, x)$;
        Prune $S$ from $C_k$ if $sup(S) < \beta$;
    end for
    Set $S_{max}$ to set in $C_k$ with maximum support;
    $C_{k+1} = \emptyset$;
    For each pair of sets $S, S'$ in $C_k$ with $k - 1$ elements in common and $k^{th}$ element with distinct attributes, add $S \cup S'$ to $C_{k+1}$;
    $k = k + 1$;
end while
return $S_{max}$;
```

from. Here, we adopt the popular *inverse document frequency* (IDF) weight to capture the importance of each word w that appears in attribute a of a seed record. We assign a weight to each q-gram equal to the sum of the IDF weights of all the words that contain it. The similarity, $sim_a(u, v)$, between values u and v is defined as the Cosine similarity metric between their corresponding vectors in q-gram space.

We now define some notation used in Algorithm 1. Consider the DOM tree representation of a web page p. For a node n in the page, let x be the unique path from the root to n in the DOM tree. We treat the path x as the position of node n in page p. Further, we use $p[x]$ to denote the value of the node n at position x in page p. For a leaf node, the value is essentially the text string contained in it. If n is an internal node, then its value is the concatenated sequence of text from the leaves of the subtree rooted at n (in the order in which the nodes appear in the DOM tree). For a seed record r in R, we denote the value of attribute $a_i \in A$ in r by $r[a_i]$.

Procedure FINDATTRPOS scans the pages of a new web site W to find node values that match attribute values in the seed database R, and uses this to infer the node position for each attribute within the pages of W. It then extracts entity records from pages of the web site by extracting the attribute values at the identified positions. We use multiple attribute matches and the template structure of pages to filter out spurious matches. Consider a subset of attributes $A' = \{a_1, \ldots, a_t\}$. Let $S = \{(a_1, x_1), \ldots, (a_t, x_t)\}$ be a configuration of attribute positions where x_i is the position of attribute a_i. We define the support $sup(S)$ of S to be the number of pages p in site W such that values at positions x_i in page p match values of attributes a_i in some seed record r.

Procedure FINDATTRPOS computes the maximal set S of (attribute, position) pairs with $sup(S) \geq \beta$ for a new web site W. The procedure first stores in $WS(a, x)$ the record, page pairs (r, p) such that $sim_a(r[a], p[x])$ exceeds a threshold (T_w). Within each page of W, there may be multiple attribute position configurations for which values in the page match (portions of) a record in the seed database R. Hence $WS(a, x)$ may contain several spurious attribute value matches. Across all the pages of W, the number of these attribute position configurations could become really large, even though most of them do not have the required support. Computing support for each of these configurations to determine the maximal configuration with support $\geq \beta$ can turn out to be very inefficient. Instead, we devise an efficient Apriori-style algorithm based on the following observation: *for a pair of (attribute, position) pairs sets S, S', if $S \subseteq S'$, then $sup(S') \leq sup(S)$*. Thus, we can use an iterative Apriori-style algorithm that generates candidate (attribute, position) pairs sets of size k in the k^{th} iteration and stores these in C_k. Candidate sets whose support is less than β are pruned from C_k since any superset of these cannot have support $\geq \beta$. The remaining sets in C_k (after pruning) are used to generate supersets of size $k + 1$ which become candidates for the next iteration.

Once we have identified the maximal set S of (attribute, position) pairs with support $\geq \beta$, we extract entity records by extracting attribute values at the specified positions in S from the pages of web site W.

Restaurant			Bibliography		
Attribute	Prec	Coverage	Attribute	Prec	Coverage
Name	74.91	90.22	Title	95.75	81.51
Address	99.74	90.22	Author	98.13	81.51
Phone	100.00	62.70	Source	100.00	58.50
Payment	100.00	11.76			
Cuisine	100.00	62.57			

Table 1: Precision and coverage of extractions for all attributes.

4. EXPERIMENTAL EVALUATION

We use two real-life datasets covering two verticals: *restaurant* and *bibliography*. Each dataset consists of a set of seed records and crawled pages from a set of test sites. We use the seed records to extract from the single-entity pages belonging to each of the test sites, and report the precision and coverage of the extractions.

The seed data for restaurants is obtained from `chefmoz.com`. Extractions are performed on 17 sites. The seed dataset consists of 40000 records randomly selected from chefmoz data. For the bibliography dataset, we use data from DBLP as seed records. The seed dataset consists of 40000 records randomly selected from this dump. 7 sites are used as test sites.

We choose a random set of 1000 pages from each dataset, and generate the ground truth for this set. Precision metrics are reported on this random set. We define the *coverage* for a dataset as the fraction of pages in the dataset from which we are able to extract core attributes.

We use the FINDATTRPOS procedure described in Section 3 to compute the attribute positions (from which values are extracted) for each test site. We vary the similarity threshold (T_w) from 0.5 to 0.9 in our experiments, and use $\beta = 10$ to prune attribute position configurations with inadequate support. Table 1 lists the precision and coverage for all attributes at the best similarity threshold of 0.7.

5. REFERENCES

[1] L. Gravano, P. Ipeirotis, N. Koudas, and D. Srivastava. Text joins in an RDBMS for web data integration. In WWW, 2003.