Collective Extraction from Heterogeneous Web Lists

Ashwin Machanavajjhala* Arun Iyer† Philip Bohannon* Srujana Merugu*
*Yahoo! Research, Santa Clara, USA
†Yahoo! Research, Bangalore, India
{mvnak,aruniyer,plb,srujana}@yahoo-inc.com

ABSTRACT

Automatic extraction of structured records from inconsistently formatted lists on the web is challenging: different lists present disparate sets of attributes with variations in the ordering of attributes; many lists contain additional attributes and noise that can confuse the extraction process; and formatting within a list may be inconsistent due to missing attributes or manual formatting on some sites.

We present a novel solution to this extraction problem that is based on *i*) collective extraction from multiple lists simultaneously and *ii*) careful exploitation of a small database of seed entities. Our approach addresses the layout homogeneity within the individual lists, content redundancy across some snippets from different sources, and the noisy attribute rendering process. We experimentally evaluate variants of this algorithm on real world data sets and show that our approach is a promising direction for extraction from noisy lists, requiring mild and thus inexpensive supervision suitable for extraction from the tail of the web.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Data Mining

General Terms

Algorithms

1. INTRODUCTION

Content portals powered by databases of extracted data are now in wide use for a number of domains [10, 18]. As such portals compete for audience engagement, each will naturally seek to enhance its database with data from lists found on a variety of smaller, structurally diverse sites. This is partially because the smaller sites are often more authoritative, but also because larger sites are often competitors. In this paper we focus on the extraction challenges faced on such smaller sites; in particular, those that contain lists of entity information; for example, lists of businesses, books, movies, etc. While some of these lists are script-generated, some are not, and traditional extraction from these lists via site-specific supervision is cost-prohibitive [5, 13].

To simplify the problem, we assume that lists have been correctly segmented into snippets by an unsupervised technique (see, e.g., [2]) and that each snippet contains *at least some* attribute data from a given entity schema. In this context, we informally define the *Snippet Extraction Problem* as: *Segment each snippet, and label the resulting segments with the correct attribute label, or* `Other` *if none applies.* Given the assumption that snippets contain exactly one record, correct attribute labeling yields entity extraction. We now illustrate the snippet extraction problem with an example:

Example. *Figure 1 shows fragments of three web pages with information about schools in California. Across the pages, information about a variety of attributes is available, including* `officialURL`, `name`, `city`, `state`, `phone`, `team` *and* `address`. *Each of these three web-pages have been segmented to yield snippets (separated by horizontal lines), each associated with a single school.*

An example of snippet $t_{2,1}$, correctly segmented and labeled, is shown between sources 2 and 3 in Figure 1. Note that portions of the snippet not corresponding to the schema have the distinguished label "`Other`*".*

Given the snippet extraction problem and the requirement of little or no supervision, two classes of techniques might generally be applied. First, an *unsupervised wrapper* (e.g. [1, 3, 9, 11]) technique might be used to further segment each snippet into likely attribute segments, and to *align* these segments across snippets. These techniques are particularly effective when strong HTML or punctuation signal is present on the site. Recently, these techniques have been generalized to collectively segment multiple sites [7]. Note that by segmenting and aligning snippets, these techniques fall short of full Snippet Extraction since *labeling is not done.* Second, *database supervision* [5, 12, 13, 15, 17] might be used to label some snippets on the site using a seed entity database, and a sequential model learned from these examples. This model can then be applied to segment and label, usually skipping the alignment step. Unlike unsupervised wrappers, databases supervision may also leverage *sequential signal*, for example, that certain attribute orderings are used on a particular site.

Challenges Two challenges arise when state-of-the-art techniques are applied to diverse and noisy web data:

1. Attribute boundaries from HTML or punctuation are often inconsistent or missing, leading to poor segmentation (and thus alignment and labeling errors).

WSDM'11, February 9–12, 2011, Hong Kong, China.

Source 1		Entity
$t_{1,1}$	• 42nd Street Elementary School– www.42StSchool.com – Los Angeles	e_3
$t_{1,2}$	• Collins Elementary–www.collinselementary.com– Cupertino	e_2
$t_{1,3}$	• **Z** Zamorano Elementary School – www.zamorano.edu– San Diego	e_4

Attributes
officialURL
name
addr
city
state
zip
phone
team

Source 2		Entity
$t_{2,1}$	† www.lincolnHS.edu, Lincoln HS – **Pirates** – (650) 343-2321 – (next game Wednesday)	e_5
$t_{2,2}$	† www.smmm.com, Memphis MS – **Grizzlies** – (408) 330-1247 – San Mateo	e_6
$t_{2,3}$	† www.montavista.edu, Monta Vista HS – **Matadors** – (408) 366-7777 – Santa Clara	e_1

Source 3		Entity
$t_{3,1}$	⋆ **Collins Cougars** – L. P. Collins Elementary – www.collinselementary.com – Cupertino, CA · 408 366-5555	e_2
$t_{3,2}$	⋆ **Monta Vista Matadors** – Monta Vista High – 21840 McClellan Rd · Cupertina, CA · 408 366-7777	e_1
$t_{3,3}$	⋆ Cupertino Union School District – www.CupertinoSD.edu – 10301 Vista Drive · Cupertino, CA	e_7

$t_{2,1}$, correctly segmented and labeled								
†	www.lincolnHS.edu	,	Lincoln HS	–	**Pirates**	–	(650) 343-2321	– (next game on Wednesday)
	officialURL		name		team		phone	Other

Figure 1: Example sources and snippets.

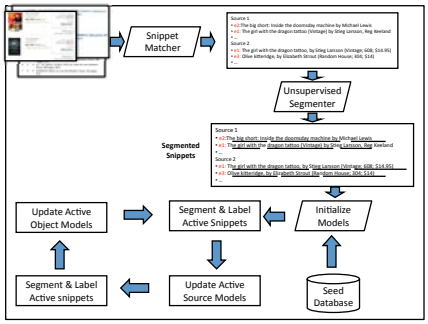


Figure 2: Collective Extraction Algorithm

2. *Confusing* attributes or *junk segments* cause a variety of problems. For example, consider *team name* and *name*, as shown in Source 3 of Figure 1, if *team name* is not in the extractor's schema then it can cause false matches to *name*. This can cause trouble for both attribute model-based techniques [7] and automatic labeling [12, 13].

Obviously these challenges interact, and in conjunction can lead to extremely difficult extraction scenarios.

Collie

In this paper, we introduce COLLIE, a novel system developed at Yahoo! for snippet extraction. The goals of COLLIE are: *i*) To work flexibly with a wide variety of HTML and non-HTML data sets, even with poor signal for attribute separation, *ii*) To improve quality in the presence of confusing junk segments, and *iii*) To work with no explicit supervision, just a small database of seed entities.

Following [7], COLLIE can *collectively* extract from multiple sites, and introduces two key improvements over prior work:

1. We provide a *collective extraction* technique that leverages *sequential models* as in [13], but we are significantly more resilient to noisy data.

2. We make explicit use of (approximate) *entity match* information throughout the process, rather than using it only initially to perform noisy labeling as in [12, 13]. For example, in Figure 1, snippets $t_{1,2}$ and $t_{3,1}$ both correspond to entity e_2, and it may be possible to identify this match prior to extracting information from $t_{1,2}$ and $t_{3,1}$. In particular, the match information is leveraged in our adaptation of Viterbi decoding for sequential labeling based on a hidden Markov model.

3. Rather than proceeding source by source as in [11, 12, 13] or using all sources at once as in [7], we carefully maintain *active sets* of sources, entities and attributes. This helps us cope with poor initial knowledge of a particular site or entity, especially when sites are noisy.

Figure 2 outlines our extraction algorithm. The three parallelograms are key initialization steps of the algorithm: first a matching of snippets is found, using an algorithm like [8]. Second, an unsupervised segmentation algorithm is applied ([16] and [7]). Third, our various models for entities, attributes and sources are initialized based on the seed entities and the unsupervised segmentation. Once initialization is complete, we iteratively perform four steps: *1*) re-segment and relabel snippets, *2*) update models of sources, *3*) re-segment and relabel again, then *4*) update entity models. After each iteration through these steps, our set of active entities, snippets and sources is revised. This is critical to avoiding garbage labelings in noisy sources.

We experimentally evaluate COLLIE against two baseline algorithms, and the recent WWT [13] algorithm on a number of challenging, real data sets taken from "Bestseller" book lists on the web. Our experimental evaluation highlights the importance of leveraging a variety of cues and collective extraction from multiple sources in order to extract from noisy data with low supervision. COLLIE achieves good attribute segmentation and labeling significantly better (achieving at least 50% error reduction in labeling on average) than the state of the art. We show that COLLIE is able to tolerate attribute confusion by leveraging sequential models, and confusing non-attribute junk segments by estimating the complete set of attributes presented on each source. Finally, while COLLIE assumes perfect knowledge of which snippets are from the same entity, we show that COLLIE can also lever-

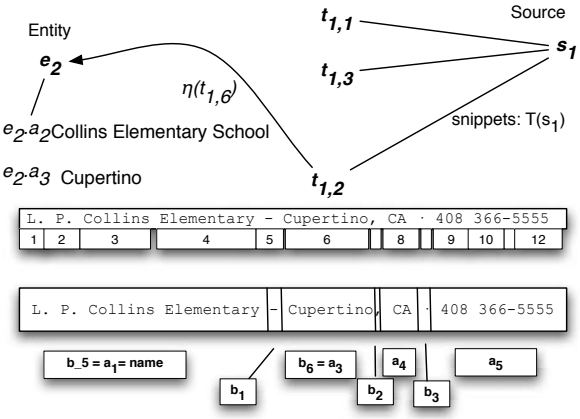


Figure 4: Source-snippet-entity associations.

age imperfect matching. In fact, we show that approximate matching schemes that match snippets with very similar attribute values can actually boost COLLIE's performance *over perfect entity matching*.

In summary, COLLIE is a promising new direction for collective extraction of structured entity records from a large number of small web sites, with zero per-site supervision, and even limited supervision in the form of seed entities.

Outline: In Section 2, we formally state the multi-site attribute extraction problem. In Section 3 we introduce the component attribute and sequential models needed in COLLIE. In Section 4, we present our detailed algorithm. In Section 5, we present our experimental results. We discuss related work in Section 6, and conclude in Section 7.

2. PROBLEM DEFINITION

In this section, we introduce sufficient notation to formally define the *collective record extraction* problem. Sets are represented using upper-case calligraphic letters (e.g., $\mathcal{X}$) with the corresponding lower case letter (e.g., x) denoting an element of $\mathcal{X}$ and $N_{\mathcal{X}}$ the cardinality.

Schema. Let $\mathcal{A}$ denote a fixed set of attributes that should be recovered by extraction. In the running example, each entity has seven such attributes – `officialURL`, `name`, `city`, `state`, `phone`, and `address`. Each attribute $a \in \mathcal{A}$ takes values from a domain denoted by $\mathsf{dom}(a)$. For example, the attribute $a_1 =$ `officialURL` takes values only among valid URLs while $a_7 =$ `state` takes string values that refer to one among the fifty US states.

Seed Entities. Let $\mathcal{E}$ denote the set of entities. Each *entity* $e \in \mathcal{E}$ is associated with a unique attribute value $e.a$ for each attribute $a \in A$ from $\mathsf{dom}(a)$ or a distinguished value null (which indicates that the attribute does not exist for that entity). Let $\mathcal{E}_{seed} \subset \mathcal{E}$ denote a set of *seed entities* whose attribute values are already known prior to the extraction. Figure 3 shows two seed entities for the scenario described in the running example.

Snippets. Let $\mathcal{T}$ be a set of text or HTML *snippets* each containing information on an entity of interest along with some potentially irrelevant information. Each snippet $t \in \mathcal{T}$ is modeled as a sequence of $|t|$ tokens. We use $t[x]$ to refer to the x^{th} token in t, and $t[x_1 : x_2]$, $x_2 > x_1$ to refer to a subsequence of tokens in t, $t[x_1] + t[x_1 + 1] + \ldots + t[x_2 - 1]$. Example snippets are shown in Figure 1.

Sources. Each snippet $t \in \mathcal{T}$ is derived from a *source* $s = S(t)$. Abusing notation, we use $S(\mathcal{T})$ or simply S to denote the set of all sources from which snippets $\mathcal{T}$ are drawn, and $\mathcal{T}(s)$ to denote the snippets from source s, i.e., $\{t|S(t) = s\}$. Figure 1 shows a set of snippets derived from three sources. In this example, each snippet embeds information about a school in the United States.

Segmentation. A *segmentation* function χ maps snippets to a sequence of increasing integers, $\chi(t) = (x_1, x_2, \ldots, x_{|\chi(t)|})$. χ breaks t into $|\chi(t)|-1$ subsequences, $t[x_r : x_{r+1}]$, $[r]_1^{|\chi(t)|-1}$ where $x_1 = 1$ and $x_{|\chi|} = |t|$. We overload [] to define $t[\chi(t)[r]]$ as $t[x_r : x_{r+1}]$. Figure 4 shows snippet $t_{1,2}$ ="L. P. Collins Elementary - Cupertino, CA ů 408 366-5555" with tokens being partitioned into 4 segments corresponding to $\chi(t_{1,2}) = \{1, 5, 6, 7, 8, 9, 13\}$, and $\chi(t_{1,2})[4] = (7 : 8)$.

Labels. Let $\mathcal{B}$ be a set of labels that includes the attributes in $\mathcal{A}$, and other auxiliary labels helpful for extraction (e.g., attribute separators and unknown attributes). In particular, let $B_i \subset \mathcal{B}$ be a set of labels that may be assigned to segments on site s_i, a non-empty subset of which are from $\mathcal{A}$. The remaining labels in B_i are assigned to punctuation and other separators, junk segments, unknown attributes, etc. In addition B_i always contains Other.

Labeling. A *labeling*, $l(t)$, maps a snippet t to a function that maps segments to labels. Usually, $l(t)(\cdot)$ is defined only on the sequences defined by a segmentation function χ, in which case $l(t)$ may be subscripted by $l_\chi(t)$. If t is from source s_i, then the range of $l(t)$ is B_i. For example, in Figure 4, the first two segments of snippet $t_{1,2}$ are labeled as $b_5 = a_1 =$ name, $b_1 = other_1$ $b_6 = a_3 =$ city respectively, so we would write $l_\chi(t)[\chi(t)[1]] = a_1$. Since this notation is cumbersome, the (t) may be omitted following l or χ when clear from context.

The **Collective Record Extraction Problem** Given a set of snippets $\mathcal{T}$, over sources $S(\mathcal{T})$ along with $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{E}_{seed}$ as defined above, to produce a solution (χ, l_χ) that agrees with the correct segmentation and labeling, up to a given equivalence (such as white space).

3. COMPONENTS

In this section, we introduce our base algorithms for segmentation, labeling and snippet matching.

3.1 Attribute and Field Models

Content models are widely used [1, 4, 5, 7, 11] to estimate the two relationships between segments ($t[x : y]$) and labels, b_i. First, we may wish to estimate $P(l \mid t[x : y])$, the probability that a label applies given the contents of the segment (*classification*), and second, we may wish to estimate $P(t[x : y] \mid b_i)$, the probability that the value $t[x : y]$ appears in a segment correctly labeled b_i (*emission*).

Attribute Models For each attribute that appears in $\mathcal{E}_{seed}$, we create a model $\{\alpha_m : a_m \in \mathcal{A}\}$. This model is not updated. We use $P_\alpha(t[x : y] \mid a_m)$ and $P_\alpha(a_m \mid t[x : y])$ to indicate the emission and classification probabilities estimated from α_m, respectively.

Source Field Models For each label $b_{m,i} \in B_i$, we define $\beta_{m,i}$ as a content model of $b_{m,i}$. The $\beta_{m,i}$ models always come from a segmentation and alignment of active snippets on a source s_i. We describe what these models correspond to in more detail in the next section. We use $P_\beta(t[x : y] \mid b_i)$

Seed Entities						
eid	a_1 : officialURL	a_2 : name	a_3 : city	a_4 : state	a_5 : phone	a_6 : addr
e_1	www.montavista.edu	Monta Vista High School	Cupertino	CA	408366777	21840 McClellan Rd
e_2	www.collinselementary.com	Collins Elementary School	Cupertino	CA	4083665555	null

Figure 3: A set of *seed entities*

or $P_\beta(b_i \mid t[x:y])$ to indicate the emission and classification probabilities estimated from β_m, respectively.

Implementation A variety of techniques have been used for content models including bag-of-words [8], ngram language models [5, 11], statistical feature based models [7], and HMM based models [1, 4]. We implement our content models by combining (a) a bag of words model P_1, (b) Poisson models for character and token lengths P_2, and (c) a few multinomial models capturing different binary features, such as presence/absence of alphabets, digits, punctuation, and html tags, P_3. The models are smoothed appropriately to account for unseen values. Finally, the emission probability $P(t[x:y] \mid b_i)$ is computed as $\prod_i P_i^{w_i}$, with manually tuned weights (we used a uniform assignment). The implementation of $P(b_i \mid t[x:y])$ is discussed in the next section.

3.2 Entity Models

Another way to estimate the emission probability arises when an *entity*, e_j, is known for the snippet, t. In this case, we use the token-level Jaccard similarity between $e_j.a_m$ and a segment $t[x:y]$ as an estimate of $P(t[x:y]|a_m)$. In particular, if $jac(x_1, x_2)$ represents the ratio of intersecting tokens between strings x_1 and x_2 to total tokens in both, then we estimate $P_e(t[x:y]|a_m) \propto e^{jac(x_1,x_2)}$.

3.3 Labeling with a Sequential Model

With each source s_i, we associate a first order Markov model that models the distribution over the *segment* labellings of the snippets in $\mathcal{T}(s_i)$. This model is parameterized by (π_i, λ_i) as follows:

- $\pi_i[m], [m]_1^{N_{B,i}}$ denote the probability that the label of the first segment for any snippet in $\mathcal{T}(s_i)$ is $b_{m,i}$.
- $\lambda[m_1][m_2]$, $[m_1]_1^{N_{B,i}}$ $[m_2]_1^{N_{B,i}+1}$ denotes the conditional probability of label $b_{m_2,i}$ following label $b_{m_1,i}$ in a snippet labeling. Note that $b_{N_{B,i}+1,i}$ is interpreted as a dummy terminal state.

Given a segmentation $\chi(t)$ for a snippet t, the probability of observing the snippet t can be written as

$$P(t|\chi, M_i, \pi_i, \lambda_i) = \sum_{l(t)} (P(t|l(t), M_i, \chi) \cdot P(l(t)|\pi_i, \lambda_i)) \quad (1)$$

$$P(t|l(t), M_i, \chi) = \prod_{r=1}^{|\chi(t)|} P(t[\chi(t)[r]] \mid l(r)) \quad (2)$$

$$P(l(t)|\pi_i, \lambda_i) = \pi_i[l(t)[1]] \cdot \prod_{r=2}^{|\chi(t)|} \lambda_i[l(t)[r-1]][l(t)[r]] \quad (3)$$

where, $l(t)[|\chi(t)|] =$ terminal.

We use the standard Viterbi [21] algorithm in order to find the labeling l_χ that maximizes $P(t|\chi, M_i, \pi_i, \lambda_i)$, the probability of generating the snippet t given segmentation χ. As input, Viterbi takes a given a segmentation $\chi(t)$, (π_i, λ_i) and an *emission probability function* that estimates $P(t[\chi(t)[r]] \mid b_{m,i})$, with $b_{m,i}$ drawn from a family of labels B_i. The way the emission probability is computed in COLLIE is one of our contributions, and it carefully depends on the currently active snippets, entities, where in most prior work only the attribute models, α_m are used, not source-specific "junk" models β_m, nor entity-specific models. The former is critical for dealing with noisy sites, and the latter is critical for capitalizing on small $\mathcal{E}_{seed}$.

Algorithm 1 COLLECTIVE EXTRACTION ALGORITHM

Input: Schema $\mathcal{A}$, set of text snippets $\mathcal{T}$ over sources $S(\mathcal{T})$, seed entity set $\mathcal{E}_{sup}$

Output: Segmentation and labeling (χ, l_χ) of snippets in $\mathcal{T}$, Entity attributes $\{e_k.a_m : a_m \in \mathcal{A}, e_k \in \mathcal{E}\}$

```
initEntitiesAndModels
initSnippetToEntityMapping
segmentSnippets
while repeat till convergence do
    relabelActiveSnippets
    updateSourceTemplates
    relabelActiveSnippets
    updateEntityModels
end while
```

3.4 Active Sets

Given a key goal of working with very low supervision, we assume that *no* snippets have been labeled, and that in general $\mathcal{E}_{seed}$ is small. Given these constraints, very few snippets will correspond to entities in $\mathcal{E}_{seed}$, and there may be some sources for which no snippets are associated with an entity in $\mathcal{E}_{seed}$. To avoid estimating field models, transition probabilities, or entity values with insufficient evidence, we limit inference to *active* entities, and carefully grow this set as the algorithm progresses. We maintain sets of *active* snippets, $\mathcal{T}_{active}$, sources S_{active} and entities $\mathcal{E}_{active}$.

4. EXTRACTION ALGORITHM

In this section, we present the COLLIE algorithm for the snippet extraction problem. The algorithm takes as input the set of snippets $\mathcal{T}$ and the seed set of entities $\mathcal{E}_{seed}$, and outputs a pair (χ, l_χ) that map each snippet $t \in \mathcal{T}$ to a segmentation, $\chi(t)$ and a labeling $_\chi(t)$. Algorithm 1 provides an overview of the key steps (also illustrated in Fig 2), and the steps are described in detail in this section.

4.1 Initialization

The algorithm begins by initializing entities, models, snippet to entity mappings, and by performing an unsupervised segmentation of each source.

4.1.1 *initEntitiesAndModels*

In this step, a variety of variables and model parameters are initialized, as follows.

Active Sets $\mathcal{E}_{active}$ is initialized to $\mathcal{E}_{seed}$, and $\mathcal{T}_{active}$ is initialized to $\{t|\eta(t) \in \mathcal{E}_{active}\}$ (snippets mentioning active objects), and S_{active} to $\emptyset$.
Global Models First, the entity attribute values, $e_i.a_m$, are initialized using the attribute values in $\mathcal{E}_{seed}$, and the global models α_m are initialized with the a_m values for each e_i.
Sequential Models The parameters (h_π, h_λ) that determine the Dirichlet priors for the source-specific initial label and transition distributions are initialized using domain knowledge (e.g., name is the first attribute, state follows city) where available, and are otherwise uniformly initialized.

Algorithm 2 RELABELACTIVESNIPPETS

```
1: T_active ← {t : η(t) ∈ E_active ⋁ S(t) ∈ S_active}
2: for t ∈ T_active do
3:    if S(t) ∈ S_active then
4:       (l_t, χ_t) ← VITERBINEIGHBORHOOD(t)
5:    else
6:       (l_t, χ_t) ← DIRECTLABEL(t, η(t))
7:    end if
8:    l(t) ← l_t(t); χ(t) ← χ_t(t)
9: end for
```

Algorithm 3 DIRECTLABEL(t, e)

```
1: l_t, ← new labeling ;
2: for r ∈ χ_i^0(t), S(t) = s_i  do
3:    m* ← argmax_m P_e(t[χ(t)[r]]|m) // object model
4:    l_t(t)[r] ← m*;
5: end for
6: // Remove multiple labels
7: for r ∈ χ_i^0(t) do
8:    m ← l(t)[r]
9:    if m exists as label of any other segment r' ∈ χ_i^0(t)
      with higher score then
10:      l_t(t)[r] ← OTHER
11:   end if
12: end for
13: return (l_t, χ_i^0); // doesn't change χ_i^0
```

4.1.2 *initSnippetToEntityMapping*

An important benefit of our collective extraction approach relative to previous work on synchronized extraction [7] is the ability to effectively exploit overlap in *entities* across sources. To do so, we need to first infer the mapping, η, from snippets to entities. This is done in two steps, given a similarity function $\sim$ that is defined on both (a) pairs of snippets ($t_1 \sim t_2$) (b) pairs consisting of one snippet and one entity ($t_1 \sim e$) (see [8] for simple but effective snippet matching techniques). For a snippet t, let e_i be the entity that maximizes $t \sim e_i$. If this match is over a threshold, then we set $\eta(t) \leftarrow e_i$. However, due to the assumption that η_{sup} is small, the bulk of entities will not be assigned, and for the remaining snippets, they are clustered using $\sim$ with the constraint that two snippets from the same site are not in the same cluster. A new "dummy entity" e_c is created, and for each cluster c, and for each t in c, $\eta(t) \leftarrow e_c$.

4.1.3 *segmentSnippets*

We first perform collective unsupervised segmentation of snippets within each source, and the resulting segmentation is then refined and labeled using our probabilistic model. To be specific, for each source s_i, we obtain a segmentation $\chi_i^0(\cdot)$ (allowing for empty segments) of the snippets in $\mathcal{T}(s_i)$ such that $|\chi_i^0(t)|$ is invariant across all snippets in s_i and the aligned segments (i.e., r^{th} segments of all snippets) are maximally similar. Currently, there exist many such techniques [7, 16, 22] and we use both [7] and [16] in our experiments.

4.2 Iterative Inference

The main portion of the algorithm alternates between updating the source template model (the B_i labels and (π_i, λ_i) transition probabilities) and the entity models (the current estimates for $e_i.a_m$ for entities not in $\mathcal{E}_{seed}$). After each such update, either to templates or entity models, RELABELACTIVESNIPPETS is called. This is actually required, since both the entity models and source templates are updated from the active snippets and their current estimated segmentation and labeling. The whole process is repeated until convergence; i.e., when the total change to the source templates and entity models is below a low threshold.[1]

Algorithm 4 VITERBINEIGHBORHOOD (t)

```
X_neigh = {χ(t)} // initialize neighborhood
for r ∈ χ(t)  do
   χ'(t) ← χ(t) with segment r and r + 1 merged
   X_neigh ← X_neigh ∪ {χ'(t)}
   for p ∈ χ(t)[r] ... χ(t)[r + 1] each token in χ(t) do
      χ'(t) ← χ(t) with segment r split after token p
      X_neigh ← X_neigh ∪ {χ'(t)}
      // find best segmentation amongst X_neigh
      return argmax_χ(t) VITERBILABEL(χ(t),COLLIEEMISSION)
   end for
end for
```

Algorithm 5 COLLIEEMISSION $(t[x : y], b_{m,i})$

Input: segment $t[x : y]$ of a snippet, and candidate label b_i
Output: $P(t[x : y]|b_{m,i})$

```
if b_i is a non-attribute model then
   return P_β(t[x : y]|b_{m,i}) using β_{m,i}
else
   let a_m be the attribute corresponding to b_{m,i}
   p ← P_α(t[x : y]|a_m)^.75 · P_β(t[x : y]|b_{m,i})^.25;
   if η(t) ∈ E_active then
      p ← p · P_e(t[x : y]|a_m)
   end if
   return p
end if
```

relabelActiveSnippets (Algorithm 2) As a first step, the active set of snippets is set to snippets that are either a) part of active sources or b) associated by η with an active entity. Next, in order to relabel active snippets with the latest entity and source-template information, we iterate through each snippet as shown in Algorithm 2. For sites that are inactive, we call DIRECTLABEL, which uses entity models P_e to label columns based on the initial aligned segmentation χ_i^0.

[1]With reasonably connectivity, we find that our algorithm quickly converges to a solution in about 5-10 iterations.

Algorithm 6 updateLabelPool

Input: source s_i, snippets $T = \mathcal{T}(s_i)$
Output: new current label pool, B_i, for s_i, training label function l_{tmp}

$B \leftarrow \emptyset$
$n \leftarrow 0$ // *junk labels*
$L \leftarrow l(t) \forall t \in T$
$\chi_i \leftarrow$ ALIGN-NW(L)
for each column r in alignment of χ_i **do**
  if r is an attribute a_m **then**
    add a_m to B
    $\forall_{t \in T} l_{tmp}(t)[r] = a_m$
  else
    add new state b_n to B
    $\forall_{t \in T} l_{tmp}(t)[r] = b_n$
    $n++$
  end if
end for
return B_i, l_{tmp}

For active sites, the re-labeling is more sophisticated, and is accomplished by viterbiNeighborhood, shown in Algorithm 4. The idea of this routine, following [7], is to perform a limited search around the current estimated segmentation of t, $\chi(t)$ (but [7] does not use a sequential model at all). This involves all new segmentations that can be formed by merging a single adjacent pair of segments, or splitting a single existing segment. Among the resulting candidate segmentations, the best labeling is chosen using the standard Viterbi decoding algorithm [21], but with using a novel emission probability shown in Algorithm 5. The idea is to carefully trade off between the use of attribute models from the initial seed set of entities $\mathcal{E}_{seed}$, unsupervised label models, and information from known entity correspondences.

updateSourceTemplates (Algorithm 7) This routine updates two aspects of the template for a source, s_i. First, the set of source-specific labels, B_i, is re-estimated. Second, the Markov parameters (π_i, λ_i) are updated. The label set B_i is computed by the subroutine updateLabelPool, shown in Algorithm 6. This routine actually embodies the key subtleties of the template update logic. First, the set of *label sequences* from all active snippets on the source are aligned by Needleman-Wunsch [16], with the constraint that *segments having two different attribute labels cannot be in the same column*. The result stored in $\mathcal{X}_i$, and due to the alignment, the r'th entry in each segment is expected to receive the same label – columns that contain at least one attribute label ℓ_a are given that label; other columns are given a distinct source specific label. This generates a temporary labeling, l_{tmp} of all the active snippets of s_i. Once updateLabelPool finishes, l_{tmp} is used to train an HMM, and the parameters of that HMM are stored as the new values of (π_i, λ_i). The use of l_{tmp} *only for training* is in fact critical – the assumptions made by the alignment routine are too strong to actually use for labeling. By using it for training, the next relabelActiveSnippets will be influenced more softly, also allowing the entity models will also be taken into account. Finally the source is added to S_{active}.

Updating Entity Parameters As in the case of sources, in each iteration, any entity that is associated with at least one active snippet is added to $\mathcal{E}_{active}$. For each entity e_k and attribute $a_m \in \mathcal{A}$, we first collect the snippet fragments with label a_m and fairly high normalized marginal likelihood values, and pick the fragment which has the highest average Jaccard distance to all other choices to be $e_k.a_m$.

Algorithm 7 updateSourceTemplates

$(B_i, l_{tmp}) \leftarrow$ updateLabelPool $(s_i, \mathcal{T}(s_i))$
if first call to updateSourceTemplates for s_i **then**
  ignore l_{tmp}, set (π_i, λ_i) to uniform probabilities over B_i
else
  $HMM \leftarrow$ hidden Markov model with states B_i
  train HMM with l_{tmp}
  $(pi_i, \lambda_i) \leftarrow$ model from HMM
  add s_j to S_{active}
end if

4.3 Algorithm Efficiency

We now briefly discuss the computational complexity of our algorithm. Let N_B denote the maximum number of labels across the sources and L_{max} the maximum snippet length. The initialization costs are linear in the size of the dataset and the data structures required for model, i.e., $O((N_S + 1)(N_B + 1)L_{max} + N_S \cdot N_B^2 + N_E \cdot N_B)$. Snippet labeling using the Viterbi algorithm as well as the marginal probability computation has a complexity that is $O(|t|^2 \cdot N_B^2)$ for a snippet t. Updating the source attribute ordering profiles requires at most $O(N_B^2)$ operations for each source. The estimation of the entity attribute values, on the other hand, is linear in the number of the candidate snippet fragments (due to the simplification via the mean model), which is usually fairly small (< 10). Therefore, the total time complexity of our algorithm for a single iteration is given by $O(|\mathcal{T}_{active}| \cdot L_{max}^2 \cdot N_B^2 + N_T \cdot L_{max} + |S_{active}| \cdot N_B^2 + |\mathcal{E}_{active}| \cdot N_A)$. In the beginning, the number of active snippets,entities and sources are fairly small and eventually cover the entire connected portion of the entity-snippet-source graph. With reasonably connectivity, we find that our algorithm converges quickly to a solution (in about 5-10 iterations).

5. EXPERIMENTAL EVALUATION

In this section, we report on comparing collie to state-of-the art techniques on a collective extraction task involving real web datasets. After describing our experimental set up, we show the following results:

- On a number of real list pages, our algorithm, collie, achieves good attribute segmentation and labeling even with a small seed set of examples entities in $\mathcal{E}_{seed}$.
- Baselines that use unsupervised segmentation fail to accurately label the segments in the presence of confusing attributes. Sequential models (like wwt) fail to correctly segment snippets in the presence of extra junk attributes. Our algorithms are able to overcome both these challenges as evidenced by collie's superior performance over unsupervised segmentation baselines as well as wwt.
- Our collective inference algorithm shows gain in performance compared to a variant of our algorithm that segments and labels each source independently.

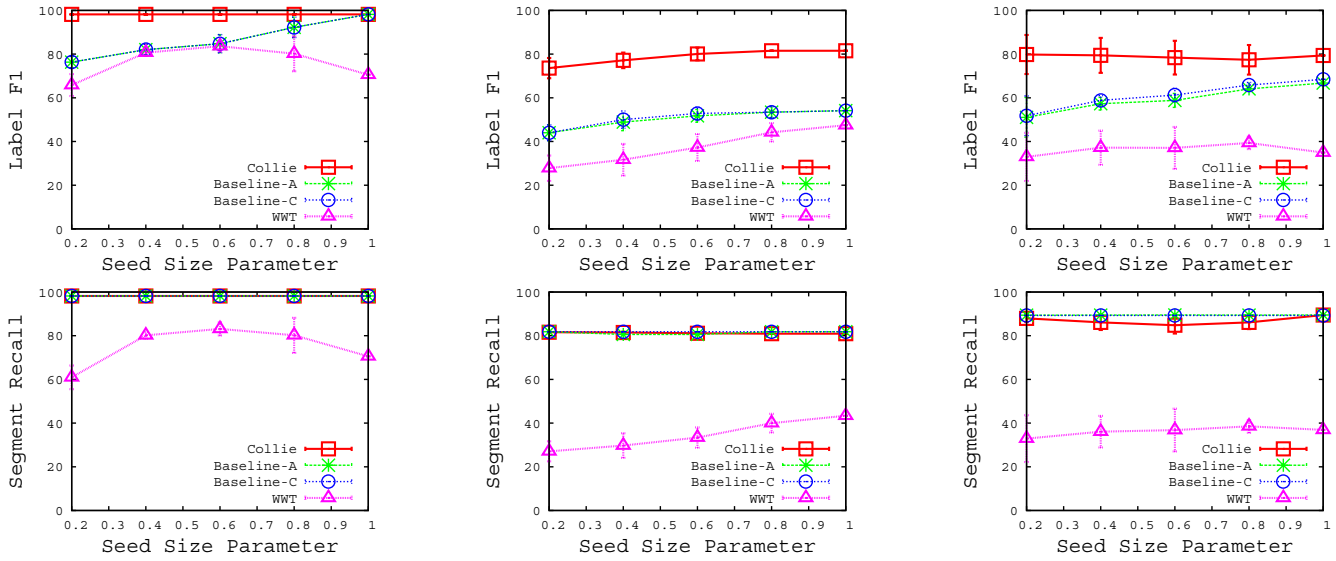


Figure 5: Performance of COLLIE and competing algorithms on HTML-1-b (left), HTML-S2 (middle) and HTML-S3 (right), with increasing seed size parameter.

- Finally, we study the impact of approximate entity match information utilized by COLLIE. We experimentally show that approximate matching schemes that ensure that matching snippets have very similar attribute values boost COLLIE's performance.

5.1 Experimental Setup

Dataset. We use challenging datasets consisting of real HTML web pages for our experiments. Each of these pages contains a list of bestselling books from different sites – (a) `amazon.com`, (bn) `barnesandnobles.com`, (p) `powells.com`, (b) `bookweb.org`[2], (pw) `publishersweekly.com`, (u) `usatoday.com`. Text snippets were extracted from these HTML pages using an unsupervised list segmentation algorithm [2]. Snippets from each source contained a subset of attributes from bookname, author_list, list_price, publisher, isbn, excerpt, type in addition to other site-specific fields. Moreover, snippets from a few of the sources (e.g., `amazon.com`) had strong HTML structure, a few had some inline HTML formatting (e.g., a tag around the book title), while others were plain textual snippets with little or no delimiters between attributes. Finally, every source had some internal variance in the number of attributes on snippets.

Since all the pages describe bestselling books, some books appear on multiple sources. We associated objectIDs with snippets (thus "matching" snippets across sources) in three ways – perfect, bookname and author. Snippets share the same perfect ID if they contain information about exactly the same entity. They share the same bookname (author) ID if they have the same bookname (author, respectively).

Our extraction schema is $\mathcal{A}$ = {bookname, author_list, list_price, publisher}. $\mathcal{E}_{seed}$ is generated by manually extracting attributes from an s% of the records on one of the sources, which we call the *database source*. This is reasonable, since one may wish to find overlapping sources based on database content, or to seed extraction with a few records from one source to supervise further extraction. We call s as *seed size parameter*.

We will use 3 variants of this dataset. **HTML-1-x** consists of a single source (x ∈ {a, b, bn, p, pw, u}), and seed entities are picked from the same source. **HTML-S2** consists of 111 snippets from 4 sources (b, p, pw, u) and mentions 94 unique books. $\mathcal{E}_{seed}$ is chosen by picking s% of the 16 entities from b. **HTML-S3** consists of 220 snippets from 5 sources (a, bn, p, pw, u) mentioning a total of 159 books; here $\mathcal{E}_{seed}$ is chosen as a s% fraction of 30 entities listed on u. Out of the 94 books mentioned **HTML-S2**, 14 appear on more than one source. Similarly for **HTML-S3**, 38 out of the 159 books appear on multiple sources.

Evaluation. All snippets were semi-automatically segmented and labeled, then manually reviewed to generate ground truth. We evaluate the output of our algorithms using precision and recall metrics defined as follows. For each snippet t, let S_t denote the set of segments which have been labeled by an algorithm with some $a \in \mathcal{A}$. Similarly, let G_t denote the set of ground truth segments labeled with some $a \in \mathcal{A}$. A segment in $x \in S_t$ is considered a *true positive* if its label and string value matches the ground truth segment x_g with the same attribute label (we ignore any leading and trailing white space and html tags); such segments are given a score $score(x) = 1$. Segments that intersect the ground truth segment, but miss or add some tokens are scored as

$$score(x) = \frac{\text{number of tokens appear in } x \text{ and } x_g}{\text{number of tokens appear in } x \text{ or } x_g} \quad (4)$$

We now define precision and recall as follows.

$$\text{precision} = \Big(\sum_{t\in\mathcal{T}}\sum_{x\in S_t} score(x)\Big) / \Big(\sum_{t\in\mathcal{T}} |S_t|\Big) \quad (5)$$

$$\text{recall} = \Big(\sum_{t\in\mathcal{T}}\sum_{x\in S_t} score(x)\Big) / \Big(\sum_{t\in\mathcal{T}} |G_t|\Big) \quad (6)$$

$$\text{F-1} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (7)$$

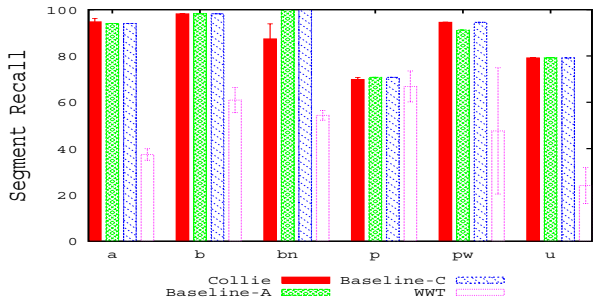
[2]American Booksellers Association

We analogously define recall for segmentation (number of ground truth segments in the predicted segmentation ignoring labels). To measure labeling performance, we use F-1 score since both precision and recall matter. For segmentation we measure only recall, as over-segmentation of "junk" segments does not affect final quality.

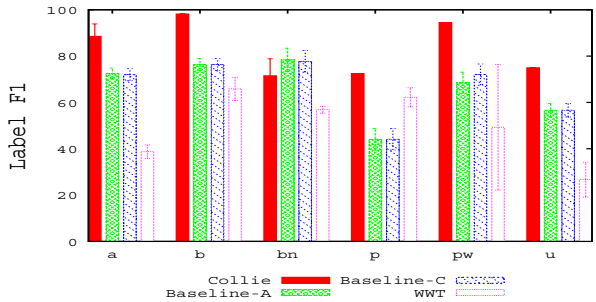
Algorithms. We refer to our algorithm as COLLIE. We perform the initial unsupervised segmentation in COLLIE using a variant of a well known alignment algorithm [16]. The same algorithm is used to align label sequences while updating source parameters. We compare our algorithms to WWT, BASELINE-C and BASELINE-A.

The BASELINE algorithms have the following structure. First, snippets are segmented in an unsupervised manner; we use alignment [16] in BASELINE-A, and Chuang et al.'s context aware wrapping algorithm [7] based on initial alignments of sources in BASELINE-C. Attribute models α_m are built for each attribute $a_m \in \mathcal{A}$, and each segment is labeled by the attribute whose model gives the segment the highest probability. Since this might result in multiple segments being labeled by the same attribute, we clean up labels by retaining only the highest scoring label for each attribute. The rest of the segments are labeled Other. WWT is the attribute extraction component from Gupta et al's [13] implementation of a list augmentation framework[3] (described in detail in Section 6).

5.2 Results



(a) Segmentation Recall on HTML-1-x with 20% seed size



(b) Labeling F-1 scores on HTML-1-x with 20% seed size

Figure 6:

Figures 6(a) and 6(b) compare the segmentation recall and labeling accuracy of the HTML-1-x datasets (x =a, b, bn, p, pw, u) each seeded with entities extracted from 20% of the snippets on those sources (5 random runs). Note that most of these sources have about 15-30 snippets, hence 20% is about 3-6 seed entities. Even with this small amount of supervision, COLLIE achieves higher accuracy both the BASELINE algorithms as well as WWT. In the rest of the section, we present more detailed results on only one of the single source datasets, HTML-1-b, and the multisource datasets HTML-S2, HTML-S3.

[3]We thank Gupta et al. for sharing their source code, and Chuang et al. for clarifying implementation details.

5.2.1 *Effect of database seed size*

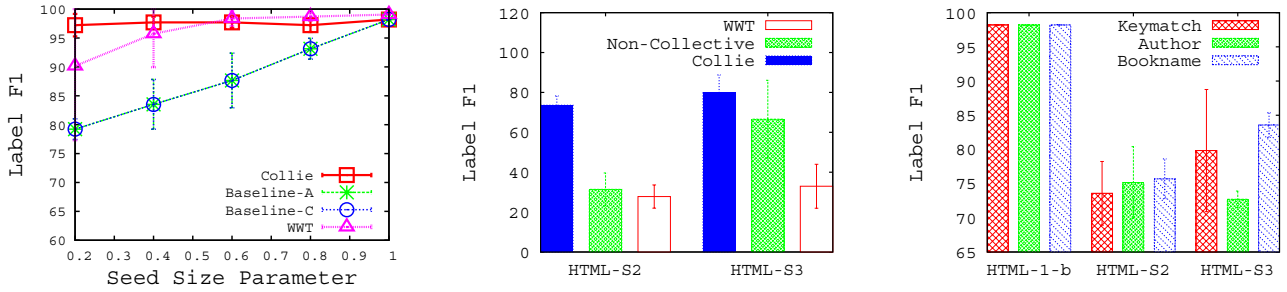
In Figure 5, we compare the accuracy of COLLIE to that of the competing algorithms for attribute extraction on HTML-1-b, HTML-S2 and HTML-S3. In each of the datasets, we varied the seed size parameter (s% of the entities picked at random), and compared labeling F-1 score and segmentation recall (averaged over 5 random runs). On all three datasets, we see that COLLIE has superior labeling F-1 over all the competing algorithms, and that COLLIE's segmentation is comparable to the unsupervised segmentation techniques, and has superior recall to to WWT's segmentation. Even a few examples are enough for COLLIE to get to its best performance; BASELINE algorithms show a mild increase in labeling F-1 scores with increasing seed size. We explain the dip in WWT's performance at higher supervision next.

5.2.2 *Effect of attribute confusion*

As mentioned in the introduction, WWT and the BASELINE algorithms have poor labeling F-1 scores due to two reasons: confusing attribute values, and confusing non attribute *junk* segments. Figure 5 illustrates that for small $\mathcal{E}_{seed}$, attribute models alone are insufficient to distinguish between confusing attributes, like e.g., *author* and *publisher*. To illustrate the effect of junk segments, in Figure 7(a), we modified the snippets HTML-1-b to only retain segments corresponding to attribute values (we removed serial number, ISBN number, description, and rank). COLLIE still has close to perfect labeling and segmentation. The BASELINE algorithms still have trouble for small $\mathcal{E}_{seed}$ due to attribute confusion. However, we see that WWT performs remarkably well. This was because in the original dataset, some author and book names also appeared in the description field. WWT initializes the segmentation, by finding string matches between attribute values in supervision and the snippets – this step wrongly matches some of the attribute values to mentions in junk segments (like description). We avoid this problem, since we leverage the unsupervised segmentation to identify both the known and unknown attribute segments on the page, and then use object models to label whole segments.

5.2.3 *Impact of Collective Extraction*

Figure 7(b) illustrates the impact of collective simultaneous extraction from multiple sites with seed size parameter set to 20%. We created a modified NON-COLLECTIVE variant of our algorithm wherein extraction proceeds on each source independent of the others. Consequently, NON-COLLECTIVE cannot leverage the values extracted on other sites, and solely depends on the amount of overlap between the snippets on the source and the initial seed set supervision. We compare the labeling accuracy of this algorithm to that of our collective algorithm and WWT. WWT is not a fully collective algorithm – it performs extraction one source at a time, but incorporates extracted values from each source to supervise the next source.



(a) Labeling F-1 scores on HTML-1-b without non attribute segments

(b) Labeling accuracy of COLLIE, WWT and a non-collective variant on HTML-S2 and HTML-S3 with 20% seed size

(c) COLLIE accuracy under different matching keys on HTML-1-b, HTML-S2, HTML-S3 datasets with 20% seed size

Figure 7:

As expected on both HTML-S2 and HTML-S3 COLLIE has higher accuracy than NON-COLLECTIVE. While the only a few seed entities may match each source, collective extraction helps improve extraction by sharing object models across sources. Surprisingly, NON-COLLECTIVE is significantly more accurate than WWT on the HTML-S3 dataset.

5.2.4 *Impact of key quality*

An important cue leveraged by COLLIE is that the same entity is mentioned on multiple sources, and that we know $\eta(t)$, the true entity associated with snippet t. While this perfect matching is seldom available, approximate matches between snippets can be easily found. To study how COLLIE behaves with imperfect keys, we tested the accuracy of extraction when using bookname ID or author ID instead of the perfect ID. As can be seen from Figure 7(c), even with imperfect keys, extraction accuracy is quite good. Such imperfect keys are quite prevalent, e.g., URLs and phone numbers indicating the wide applicability of our approach.

In some cases, like bookname ID, we get better labeling accuracy than with perfect keys. Segmenting a snippet t is helped the most if an algorithm knows that the attribute value extracted from a different snippet t' matches that attribute on t. Knowing the perfect key does not guarantee this; e.g. the title of the book "Titan's curse" is represented as "The Titan's Curse (Percy Jackson and the Olympians #03)" in one source and as "Percy Jackson and the Olympians, Book 3:The Titan's Curse" in another source. Combined with a small seed set, this variation can lead to a drop in accuracy. In fact, we computed the average pairwise Jaccard distances between attribute values for snippets sharing the same key, which we will call *keyPurity*. For instance, in HTML-S3, we found that $keyPurity(\text{perfect}) = 0.79$ while the $keyPurity(\text{bookname}) = 0.88$; this might explain the better accuracy.

5.3 Discussion

The study on the books data sets supports four key points about our approach to collective extraction: First, by leveraging a variety of cues, COLLIE, achieves good attribute segmentation and labeling even with small seed sets significantly better than state of the art (Section 5.2.1). Second, our algorithm is able to label accurately despite attribute confusion, and confusing non attribute segments (Section 5.2.2). Third, our collective extraction algorithm outperforms non-collective baselines (Section 5.2.3). Finally, while COLLIE assumes perfect knowledge of $\eta(t)$, the true entity associated with the snippet, COLLIE can also use imperfect keys. We show that approximate matching schemes which ensures that matching snippets have very similar attribute values boost COLLIE's performance (Section 5.2.4).

6. RELATED WORK

Recently, there has been considerable interest in large scale extraction of multi-attribute entity records from list pages on the web. Such extraction entails discovery of list page sources, segmentation of lists into snippets that correspond to entity descriptions, and finally, extraction of entities themselves. Our current work focuses on the last step, i.e., entity extraction given multiple segmented lists from different sources. Most existing techniques for addressing this problem follow three main paradigms discussed below.

The first class of approaches make use of source-specific supervision (i.e., a small number of segmented and labeled records on each list page) to learn attribute-ordering and layout patterns via wrapper induction [6, 14] or conditional random fields [20, 23], which are then used to label the rest of the snippets. In contrast, we use supervision strictly from an external database. There exist variants [5] that make use of annotations generated from a large seed database to learn source-specific models even without explicit source-specific supervision. In either case, the labeled snippets have to be separately deduped [19] to finally identify the unique entities. This approach is, however, prohibitively expensive to scale due to the excessive supervision requirements and does not take any advantage of the content overlap between sources. Recent techniques [15] add information from the database and perform deduping, but still require source-specific supervision, perhaps due to the use of a discriminative technique (CRFs).

The second class of approaches [7, 11] perform attribute extraction using both within-source structural regularity (e.g., HTML, delimiter cues, lexical features), as well as global attribute distributional properties. Most of these techniques involve two core steps where the first step collectively segments all the snippets in a source into aligned fields in an unsupervised fashion and the second step involves matching (or labeling given a reference database) the fields across different sources using the global distributional properties. There also exists an alternative approach [23] that makes use of the same predictive signals, but in a coupled fashion via

a conditional random field with features chosen carefully to capture the local structural similarity and global commonalities. This CRF-based technique, however, requires a fair bit of supervision, typically 1/3 of the sources in the test set in experiments,[4] and thus cannot be used if site-specific supervision is not cost-effective. Though all these approaches account for global distributional commonalities, these techniques do not involve any entity deduping and cannot make use of any fine-grained dependencies arising from multiple snippets describing the same entity. These methods also tend to be not as effective in the presence of confusing attributes, e.g., phone and fax numbers, and moderate structural variations within a source.

Gupta and Sarawagi [13] recently proposed a third approach called World Wide Tables (WWT) that is closest in spirit to our current work. Like COLLIE and unlike most of the related work, WWT can handle lists with little or no HTML structure. WWT is an end-to-end system for expanding a query table using unstructured lists on the web that addresses source discovery, extraction, and ranking. Since the initial query is only a few example tuples, the size of supervision required is modest. The WWT approach labels records on a new site based on the current database, and adds to that database after extraction.

Unlike our collective extraction approach, each inference step is performed independently without adequately accounting for the uncertainty involved in the preceding steps and hence, there is a considerable potential for errors to propagate. The choice of discriminative model in WWT allows for inclusion of a wide variety of features, but places the onus on the system designer to specify these features. It also requires extremely high quality initial annotations since it operates in a supervised fashion. On the other hand, our sequential generative model models the snippet labeling as a latent variable allowing us to capture the uncertainty in the annotations in a principled fashion. Our model also allows discovery of unknown attributes, and can capture the layout patterns fairly well even in the presence of such latent fields, while the semi-Markov CRF has to be provided additional long-range features to obtain a comparable performance. Furthermore, our *active set* based incremental inference is significantly more powerful than WWT's source prioritization (based on the number of annotations).

7. CONCLUSIONS

In this paper, we consider the problem of multi-attribute record extraction from unstructured lists on the web. This is a challenging task due to the need to extract from a variety of small sites with different formatting templates. We present a novel approach to jointly extract attribute values for these records based on collective extraction with a sequential model, use of unsupervised segmentation and local search to handle additional attributes and junk segments, and careful active set management to better cope with noisy sources and confusing attributes. We evaluate our algorithms on real, noisy data sets and obtain superior performance relative to some of the existing state-of-art techniques indicating that our methodology is able to effectively exploit multiple types of predictive cues, and adapt to difficult, noisy extraction cases where site-specific supervision is cost-prohibitive.

[4]We thank the authors for this clarification via private communication.

8. REFERENCES

[1] E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *KDD*, pages 20–29, 2004.

[2] M. Alvarez, A. Pan, J. Raposo, F. Bellas, and F. Cacheda. Extracting lists of data records from semi-structured web pages. *Data Knowl. Engg.*, 2008.

[3] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD, 2003*. ACM, 2003.

[4] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. *SIGMOD Rec.*, 30(2), 2001.

[5] S. Canisius and C. Sporleder. Bootstrapping information extraction from field books. In *EMNLP*, pages 827–836, 2007.

[6] C. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. Knowl. Data Eng.*, 2006.

[7] S.-L. Chuang, K. C.-C. Chang, and C. Zhai. Context-aware wrapping: Synchronized data extraction. In *VLDB*, 2007.

[8] W. W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Trans. Inf. Syst.*, 18(3), 2000.

[9] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, 2001.

[10] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured web community portals: A top-down, compositional, and incremental approach. In *VLDB*, pages 399–410, 2007.

[11] H. Elmeleegy, J. Madhavan, and A. Halevy. Harvesting relational tables from lists on the web. In *Proceedings of the VLDB Endowment (PVLDB)*, pages 1078–1089, 2009.

[12] P. Gulhane, R. Rastogi, S. Sengamedu, and A. Tengli. Exploiting content redundancy for web information extraction. In *VLDB*, 2010.

[13] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. In *VLDB*, 2009.

[14] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *IJCAI*, 1997.

[15] I. R. Mansuri and S. Sarawagi. Integrating unstructured data into relational databases. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 29, Washington, DC, USA, 2006.

[16] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol. Bio.*, 1970.

[17] P. Papotti, V. Crescenzi, P. Merialdo, M. Bronzi, and L. Blanco. Redundancy-driven web data extraction and integration. In *WebDB*, 2010.

[18] A. Rajaraman. Kosmix: Exploring the deep web using taxonomies and categorization. *IEEE Data Eng. Bull.*, 32(2):12–19, 2009.

[19] P. Ravikumar and W. Cohen. A hierarchical graphical model for record linkage. In *UAI '04: Proceedings of the 20th conference on Uncertainty in Artificial Intelligence*, pages 454–461, 2004.

[20] C. Sutton and A. Mccallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*, chapter 4. MIT Press, 2007.

[21] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2), 1967.

[22] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *WWW*. ACM, 2005.

[23] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. Simultaneous record detection and attribute labeling in web data extraction. In *KDD*, 2006.