Joint Training for Open-domain Extraction on the Web: Exploiting Overlap when Supervision is Limited

Rahul Gupta
IIT Bombay
grahul@cse.iitb.ac.in

Sunita Sarawagi
IIT Bombay
sunita@cse.iitb.ac.in

ABSTRACT

We consider the problem of jointly training structured models for extraction from multiple web sources whose records enjoy *partial content overlap*. This has important applications in open-domain extraction, e.g. a user materializing a table of interest from multiple relevant unstructured sources; or a site like Freebase augmenting an incomplete relation by extracting more rows from web sources. Such applications require extraction over arbitrary domains, so one cannot use a pre-trained extractor or demand a huge labeled dataset. We propose to overcome this lack of supervision by using content overlap across the related web sources. Existing methods of exploiting overlap have been developed under settings that do not generalize easily to the scale and diversity of overlap seen on Web sources.

We present an agreement-based learning framework that jointly trains the models by biasing them to agree on the *agreement regions*, i.e. shared text segments. We present alternatives within our framework to trade-off tractability, robustness to noise, and extent of agreement enforced; and propose a scheme of partitioning agreement regions that leads to efficient training while maximizing overall accuracy. Further, we present a principled scheme to discover low-noise agreement regions in unlabeled data across multiple sources.

Through extensive experiments over 58 different extraction domains, we establish that our framework provides significant boosts over uncoupled training, and scores over alternatives such as collective inference, staged training, and multi-view learning.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Parameter learning; I.2.7 [**Artificial Intelligence**]: Text Analysis

General Terms

Algorithms, Experimentation

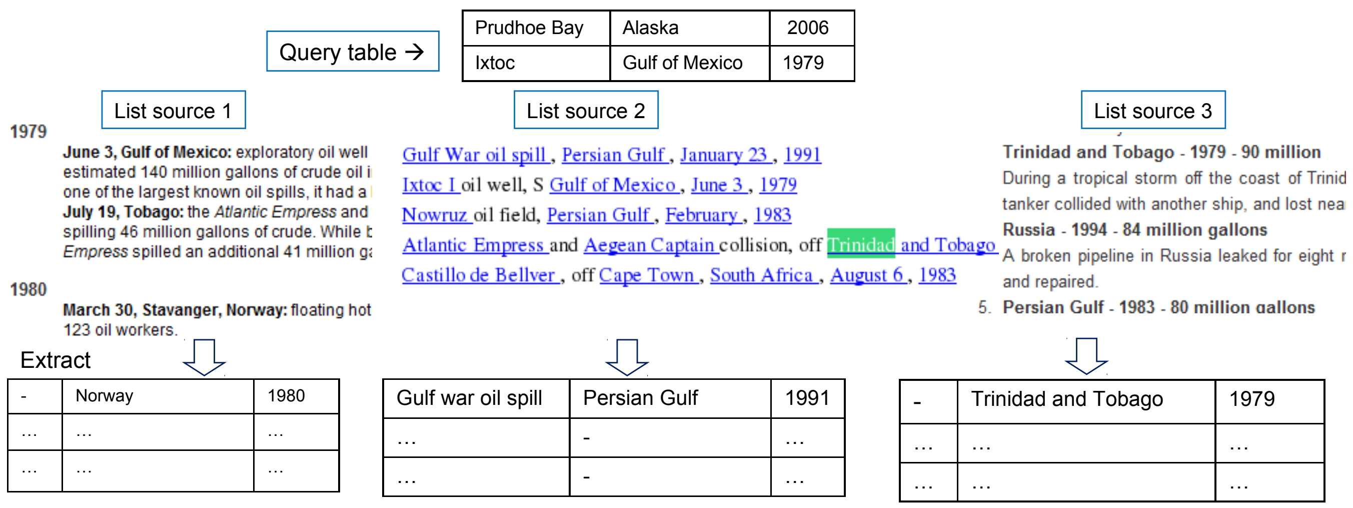
Keywords

Collective training, graphical models, information extraction

1. INTRODUCTION

There continues to be increasing research and commercial interest in harvesting and querying for structured data on the Web. This calls for automated techniques for converting organically created unstructured web sources to structured views that align with a user's needs. Since the space of possible domains that a user could query is open-ended, it is necessary to design information extraction methods that do not depend on either domain-specific extraction rules or a large supervised labeled set as in classical tasks such as extracting named entities, citations, or personnel records. On the web, the lack of training data is often compensated by redundancy of representation, wherein the same information is expressed in multiple formats and styles. In this paper we show how to exploit such redundancy for improving the accuracy of extractors that convert HTML lists into structured tables. HTML lists are a much better source of structured data than totally unformatted free-format documents, and are a valuable component of the recently popularized table materialization tasks on the Web [13, 7, 10, 1].

In table materialization, a user wishes to compile a table of structured records, say a table of oil spills with fields like TankerName, SpillLocation, and SpillDate. The user starts with a small seed set of rows, and based on a keyword match filters several HTML list sources that provide more records, as shown in Figure 1. However, each HTML source is unstructured and in order to obtain clean consolidated results from the union of these sources, it is necessary to first convert each list record to a structured row containing the fields of interest. The lists are not necessarily machine generated, therefore pattern-based extractors (as in Web wrappers) are ruled out. We resort to statistical models such as Conditional Random Fields (CRFs) to convert each of the S lists into a set of structured records. As in Figure 1, each list source usually has a distinct style and arbitrary feature set, so it is necessary to train a *separate CRF for each source*. Using the seed query rows we can obtain only a very limited amount of labeled data in each list source [13]. However, many text segments are repeated across the various sources. For example, in the snippet of the three list sources in Figure 1 the text segment "Atlantic Empress" appears in sources 1 and 2, "Trinidad and Tobago" appears in source 2 and 3, and "Tobago" appears in all three sources. The question we address in this paper is how to train the S extraction models, where each model has only a limited amount of labeled



Prudhoe Bay	Alaska	2006
Ixtoc	Gulf of Mexico	1979

1979

June 3, Gulf of Mexico: exploratory oil well
estimated 140 million gallons of crude oil i
one of the largest known oil spills, it had a
July 19, Tobago: the *Atlantic Empress* and
spilling 46 million gallons of crude. While t
Empress spilled an additional 41 million g

1980

March 30, Stavanger, Norway: floating hot
123 oil workers.

Gulf War oil spill , Persian Gulf , January 23 , 1991
Ixtoc I oil well, S Gulf of Mexico , June 3 , 1979
Nowruz oil field, Persian Gulf , February , 1983
Atlantic Empress and Aegean Captain collision, off Trinidad and Tobago
Castillo de Bellver , off Cape Town , South Africa , August 6 , 1983

Trinidad and Tobago - 1979 - 90 million
During a tropical storm off the coast of Trinid
tanker collided with another ship, and lost nea
Russia - 1994 - 84 million gallons
A broken pipeline in Russia leaked for eight r
and repaired.
5. **Persian Gulf - 1983 - 80 million gallons**

-	Norway	1980
…	…	…
…	…	…

Gulf war oil spill	Persian Gulf	1991
…	-	…
…	-	…

-	Trinidad and Tobago	1979
…	…	…
…	…	…

Figure 1: Example structured query to retrieve structured oil spill records from unstructured list sources. Lists differ a lot in style and features, thus need dedicated extraction models.

data but where among the unlabeled records of each source there are many overlapping text segments.

We propose a training objective to jointly train all S models so as to maximize the likelihood of the labeled data on the one hand, and the likelihood of the models agreeing on the entity labels of the overlapping text on the other. Typically this, and many other related training objectives lead to an intractable inference problem during training, usually tackled with approximate inference [15]. However, we show that an alternative strategy based on partitioning the set of overlapping segments into subsets of *tractable* segments enables more efficient training, while providing similar accuracy improvements. We present an algorithm for finding such a partitioning that clusters adjacent segments together to preserve correlation and hence accuracy, while ensuring tractability of each cluster. Another challenge of exploiting overlap is that of choosing an *agreement set* viz. the set of overlapping segments on which we desire agreement. We show that naïve approaches based on enforcing agreement among all occurrences of a repeated word introduce a lot of noise in the agreement set. We present an alternative strategy that leads to a significantly cleaner and more compact agreement set, with vast gains in training time and accuracy. We present an extensive evaluation on 58 real-life collective extraction tasks, each enforcing agreement between 2 to 20 sources, and covering a rich spectrum of data characteristics. We show that the partitioned objective along with our partitioning algorithm provides the best trade-offs of accuracy and training time compared to the alternatives.

2. RELATED APPROACHES

Statistical learning and NLP communities offer several relevant techniques for this problem. An established method that exploits content repetition is *collective inference* [22, 6, 17, 11]. When applying it to our problem, the S models are trained independently ignoring the overlap. During deployment, inference is done jointly on records of all sources to encourage agreement in the entity labels of repeated content.

In contrast approaches that exploit content overlap during training are expected to be more effective than collective inference, whose benefit is limited to repeated text. One such approach of "growing the seed set" i.e. *bootstrapping* has been applied in various information harvesting and extraction settings [2, 8, 13, 21]. In [13] we developed one such strategy called *staged training* where we train a more confident source first, label the records in that source using the trained model, add the top-few most confident labeled records to the seed labeled set, use this augmented seed set to generate labeled records for the next source, and so on. This snow-balling of the seed set is well-known to occasionally lead to error cascades and consequent performance degradation.

Another form of label-transfer is used in methods like Co-Boosting [9], Co-Training [4], and two-view Perceptrons [5]. These methods alternately refine two models in tandem by each model providing labeled data for the other. The success of these methods crucially depends on the presence of two or more views of the data that make independent errors, which is rarely the case in practice.

Instead of hard label-transfer, a more robust and elegant approach is to define a single training objective over all models. One such method is *posterior regularization* (PR) [12] that trains the various models so as to minimize the distance between the posteriors over the labels of the unlabeled data. PR was shown to achieve significant accuracy gains over the label transfer approaches [12]. However PR is not directly applicable to our setting as it is developed in the context of *multi-view learning* where multiple models are trained on different views of a *single source*. In contrast we have *multiple sources* with arbitrary overlapping content. Further, multi-view learning enjoys noise-free agreement sets where the records across views are *known duplicates* instead of *assumed duplicates* like in our case. Still, our method is similar as it also defines a joint training objective over all the models. In Section 6.2 we will show how we can adapt the PR multi-view objective to our problem and contrast our proposed objective with it both empirically and analytically.

Our problem also differs from multi-task learning that trains multiple models for different tasks on a *single data source* using shared feature representations. Our setup is closest to the agreement-based learning framework [18, 19] which trains multiple models so as to maximize the likelihood of agreement on shared variables. However, [18, 19] assume that all models agree on the same set of variables — this trivially holds for two sources where these methods have been applied. In our case any subset of models can agree on any subset of variables, and the number of sources is often as large as 20. As the number of sources increases, there is a bewildering number of ways in which they can overlap. This makes it challenging to devise objectives that maximally exploit overlap, but also account for noisy shared segments, and intractability of training. We are aware of no study where such issues are addressed in the context of jointly training more than two sources with partial overlap.

Last, there are scoped-learning methods [3, 23] that assume features to be a mix of global and source-specific functions. In our problem due to lack of labeled data we do not have any word features, so our sources end up not sharing any global features in many cases.

3. COLLECTIVE TRAINING

Our goal is to train S Conditional Random Fields (CRFs) corresponding to the S source lists from which we wish to extract the columns of a query table Q with a small seed set of rows. We again stress the need for separate models as the lists vary a lot in their features, thus ruling out the possibility of learning a common model for all of them.

The CRF for a list source $s \in \{1, \ldots, S\}$ defines a distribution $\mathrm{P}_s(\mathbf{y}|\mathbf{x})$ over a vector of output labels $\mathbf{y} = y_1, \ldots, y_n$ given tokens $\mathbf{x} = x_1, \ldots, x_n$ of a single record $\mathbf{x}$ in s. Each $y_p \in \mathbf{y}$ takes one of a discrete set $\mathcal{L}$ of labels denoting the columns of the query[1] and a special label "None of the above". The distribution is constructed from a feature set $\mathbf{f}_s(\mathbf{x}, \mathbf{y})$ that captures various properties of the context in which data is laid out in the list s, including the presence of delimiters, HTML tags, frequent words, order of labels, and so on. Then the CRF defines $\mathrm{P}_s(\mathbf{y}|\mathbf{x}, \mathbf{w}_s)$ in terms of these features and parameters $\mathbf{w}_s$ as:

$$\mathrm{P}_s(\mathbf{y}|\mathbf{x}, \mathbf{w}_s) = \frac{1}{Z(\mathbf{x}, \mathbf{w}_s)} \exp(\mathbf{w}_s \cdot \mathbf{f}_s(\mathbf{x}, \mathbf{y})) \tag{1}$$

where $Z(\mathbf{x}, \mathbf{w}_s) = \sum_{\mathbf{y}'} \exp(\mathbf{w}_s \cdot \mathbf{f}_s(\mathbf{x}, \mathbf{y}'))$ is the normalizing constant, known as the *partition function*. As in standard CRFs, the features decompose over individual positions (called node features) and pairs of adjacent positions (called edge features) as follows: $\mathbf{f}_s(\mathbf{x}, \mathbf{y}) = \sum_p \mathbf{f}_s(\mathbf{x}, y_p, p) + \mathbf{f}_s(\mathbf{x}, y_{p-1}, y_p, p)$. Thus $Z(\mathbf{x}, \mathbf{w}_s)$ can be efficiently calculated using the forward-backward algorithm. Each list s comes with a small labeled set $L_s = \{(\mathbf{X}_{s1}, \tilde{\mathbf{Y}}_{s1}), \ldots, (\mathbf{X}_{sm_s}, \tilde{\mathbf{Y}}_{sm_s})\}$ obtained by matching s to the structured rows in query Q; and many unlabeled records $U_s = \{\mathbf{X}_{sm_s+1}, \ldots, \mathbf{X}_{sn_s}\}$ where m_s, n_s denote the number of labeled and total records respectively in source s.

The traditional goal of training is to find a $\mathbf{w}_s$ that maximizes the regularized likelihood of the labeled data L_s:

$$\mathrm{LL}_s(L_s, \mathbf{w}_s) = \sum_{(\mathbf{X}_{si}, \tilde{\mathbf{Y}}_{si}) \in L_s} \log \mathrm{P}_s(\tilde{\mathbf{Y}}_{si}|\mathbf{X}_{si}, \mathbf{w}_s) - \gamma ||\mathbf{w}_s||^2 \tag{2}$$

where γ is set via cross-validation to restrict over-fitting. This objective is concave in $\mathbf{w}_s$ and is maximized easily using gradient ascent. The gradient can be computed using exact sum-inference since the features decompose over adjacent positions. When L_s is small, the parameters $\mathbf{w}_s$ are not trained adequately and are often prone to over-fitting.

[1]In practice, the label space is encoded in a finer grained fashion denoting the begin, continuation, end of labels as in standard information extraction [21]

Our goal is to improve upon traditional training by exploiting text overlap in the unlabeled data of the various sources. We next describe how the overlap is represented.

Representing overlap.

We formally represent overlap as an *agreement set* $\mathcal{A}$ comprising of a set of *shared segments*. A shared segment is a span of contiguous text that occurs in the records of two or more sources, e.g. Trinidad and Tobago in Figure 1. Thus each shared segment $C \in \mathcal{A}$ is associated with a span of tokens $\mathbf{t}$ and a list of triples (s, i, r) indicating the source s, the record $i \in U_s$, and the contiguous range of tokens $r \in i$ that matches $\mathbf{t}$. We do not assume any mutual exclusion between shared segments, and each shared segment can repeat across an arbitrary number of sources and have an arbitrary length. Also, a shared segment can repeat in records from the same source, but we prohibit a shared segment from spanning two different spans of the same record to avoid intractability. Figure 2(a) presents an example of an agreement set comprising of four shared segments. Assume that the three sentences are first records of three different list sources. The first shared segment corresponding to $\mathbf{t}$ = "Matt Groening , The Simpsons" includes two member triples $(2, 1, [4 \ldots 8])$ and $(3, 1, [3 \ldots 7])$ since it appears in the first record of sources 2 and 3 at token spans $[4 \ldots 8]$ and $[3 \ldots 7]$ respectively. For now we assume that $\mathcal{A}$ has already been computed. In Section 5 we present our strategy for principled computation of the agreement set.

We now use the agreement set $\mathcal{A}$ to augment the baseline objective of Equation 2. To the base term, we add the log-likelihood of the S models agreeing on the labeling of shared segments in $\mathcal{A}$. We define this likelihood next.

First, let $\mathbf{Y}$ be a random variable vector that denotes the concatenation of all the record labelings $\mathbf{Y}_{si}$. Since the individual $\mathbf{Y}_{si}$ are independent of each other, we have

$$\mathrm{P}(\mathbf{Y}|\mathbf{X}, \mathbf{w}_1, \ldots, \mathbf{w}_S) \triangleq \prod_{(s,i)} \mathrm{P}_s(\mathbf{Y}_{si}|\mathbf{X}_{si}, \mathbf{w}_s) \tag{3}$$

where (s, i) denotes record i in source s as before, and $\mathbf{X}$ represents all the records $\mathbf{X}_{si}$.

A given $\mathbf{Y}$ is consistent w.r.t. $\mathcal{A}$ iff for each segment in $\mathcal{A}$, $\mathbf{Y}$ assigns the same labeling to all its occurrences. Therefore we define the set of consistent labelings as:

$$\mathcal{Y}_{\mathcal{A}} \triangleq \{\mathbf{Y} : \forall C \in \mathcal{A}, (s, i, r), (s', i', r') \in C : \mathbf{Y}_{sir} = \mathbf{Y}_{s'i'r'}\}$$

We define the log-likelihood of agreement w.r.t. $\mathcal{A}$ as:

$$\begin{aligned} \mathrm{LL}(\mathcal{Y}_{\mathcal{A}}, \mathbf{W}) &\triangleq \log \Pr(\mathcal{Y}_{\mathcal{A}}|\mathbf{W}) \\ &= \log \sum_{\mathbf{Y} \in \mathcal{Y}_{\mathcal{A}}} \prod_{(s,i)} \mathrm{P}_s(\mathbf{Y}_{si}|\mathbf{X}_{si}, \mathbf{w}_s) \end{aligned} \tag{4}$$

where $\mathbf{W}$ is shorthand for $(\mathbf{w}_1, \ldots, \mathbf{w}_S)$. Our goal now is to jointly train $\mathbf{w}_1, \ldots, \mathbf{w}_S$ so as to maximize a weighted combination of the traditional likelihood of the labeled data and that of agreement over $\mathcal{A}$:

$$\max_{\mathbf{w}_1, \ldots, \mathbf{w}_S} \sum_s \mathrm{LL}(L_s, \mathbf{w}_s) + \lambda \mathrm{LL}(\mathcal{Y}_{\mathcal{A}}, \mathbf{W}) \tag{5}$$

where λ is a balancing parameter set using cross-validation.

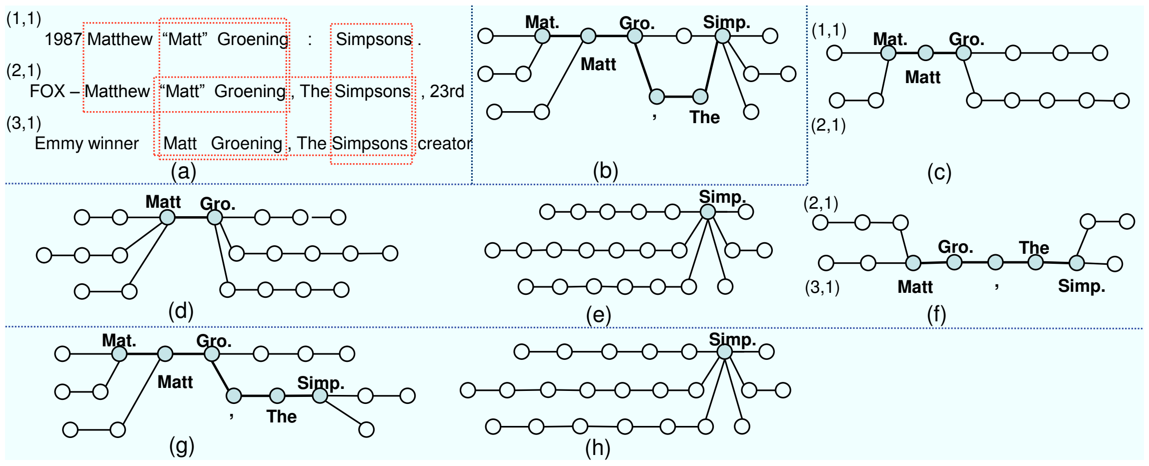


Figure 2: (a) Three samples sentences from different sources with $\mathcal{A}$={(Matthew Matt Groening), (Matt Groening), (Matt Groening , The Simpsons), (Simpsons)}. (b) The fused graph. (c)-(f) Per-segment partitioning that defines one fused model per segment. (g)-(h) Tree partitioning that partitions $\mathcal{A}$ and forms a tractable fused model for each part. Parentheses denote (source-id, record-id) for the chains.

3.1 Computing $\text{LL}(\mathcal{Y}_\mathcal{A}, \mathbf{W})$

It is clear that for agreement sets containing arbitrarily overlapping shared segments spanning many sources, the computation of $\text{LL}(\mathcal{Y}_\mathcal{A}, \mathbf{W})$ explicitly via Equation 4 is not tractable because the space of labelings $\mathcal{Y}_\mathcal{A}$ could be exponentially large. In this section we show that $\text{LL}(\mathcal{Y}_\mathcal{A}, \mathbf{W})$ is equivalent to the log of the partition function in a suitably defined graphical model. This connection enables us to apply well-known algorithms from the extensive graphical model inference literature to computing $\text{LL}(\mathcal{Y}_\mathcal{A}, \mathbf{W})$.

Substituting Equation 1 in Equation 4, we get

$$\text{LL}(\mathcal{Y}_\mathcal{A}, \mathbf{W}) = \log \sum_{\mathbf{Y}\in\mathcal{Y}_\mathcal{A}} \exp(\sum_{(s,i)} \mathbf{w}_s \cdot \mathbf{f}_s(\mathbf{X}_{si}, \mathbf{Y}_{si})) - \sum_{(s,i)} \log Z(\mathbf{X}_{si}, \mathbf{w}_s) \quad (6)$$

The second part in this equation is the sum of the log-partition function over individual records which also appears during normal CRF training and can be computed efficiently. The first part is equal to the log-partition function of a *fused graphical model* $G_\mathcal{A}$ constructed as follows.

Fused Model for $\mathcal{A}$: Initially, each record (s,i) uses its distribution $P_s(.)$ to create its base chain model G_{si}, which has one node per token in $\mathbf{X}_{si}$. Next, for each shared segment $C \in \mathcal{A}$, and for each pair of occurrences $(s,i,r),(s',i',r') \in C$, we collapse the nodes of the span r in G_{si} with the corresponding nodes of the span r' in $G_{s'i'}$, along with the corresponding edges. Figure 2(d) shows the fused graph after collapsing on the shared segment "Matt Groening". We do this successively for each segment in $\mathcal{A}$ to get the final graph $G_\mathcal{A}$. Figure 2(b) shows the final graph after collapsing on all the four shared segments. Since the graph fuses shared segments together, we call it a fused model.

We now define the log-potentials for $G_\mathcal{A}$. Let K be the number of nodes in $G_\mathcal{A}$ and $z_1, \ldots, z_K$ denote random variables for the node labels. Every node j in an initial graph G_{si} is now mapped to some final node $k \in 1, \cdots, K$, and let this mapping be denoted by $\pi(s,i,j)$. The log-potential for a node k in $G_\mathcal{A}$ is simply an aggregate of the log-potentials of the chain nodes that collapsed onto it.

$$\theta_k(z) \triangleq \sum_{(s,i,j):\pi(s,i,j)=k} \mathbf{w}_s \mathbf{f}_s(\mathbf{X}_{si}, z, j) \quad (7)$$

and the log-potential for an edge (k,ℓ) is the aggregate of the log-potentials of the edges that collapsed onto it:

$$\theta_{k,\ell}(z', z) \triangleq \sum_{(s,i,j):\pi(s,i,j-1)=k,\pi(s,i,j)=\ell} \mathbf{w}_s \mathbf{f}_s(\mathbf{X}_{si}, z', z, j) \quad (8)$$

The above θ parameters now define a distribution over the fused variables $z_1, \ldots, z_K$ as follows:

$$\text{P}_\mathcal{A}(\mathbf{z}|\theta) = \frac{1}{Z_\mathcal{A}(\theta)} \exp(\sum_{k=1}^{K} \theta_k(z_k) + \sum_{(k,\ell)\in G_\mathcal{A}} \theta_{k,\ell}(z_k, z_\ell))$$

where $Z_\mathcal{A}(\theta)$ is the partition function of the distribution:

$$Z_\mathcal{A}(\theta) = \sum_{\mathbf{z}'} \exp(\sum_k \theta_k(z'_k) + \sum_{(k,\ell)\in G_\mathcal{A}} \theta_{k,\ell}(z'_k, z'_\ell))$$

Observe that fusing the nodes trivially leads to agreement on their labels, and that $\log Z_\mathcal{A}(\theta)$ is the same as the first term of Equation 6. If the set of shared segments in $\mathcal{A}$ is such that the fused graph $G_\mathcal{A}$ has a small tree-width, we can compute $\log Z_\mathcal{A}(\theta)$ efficiently. Since this is rarely the case, we need to approximate the term in various ways. We discuss several such approximations in Section 4.

To summarize, we equated $\text{LL}(\mathcal{Y}_\mathcal{A}, \mathbf{W})$ to the better understood log partition function of a graphical model [15]. With this reduction, we can characterize the complexity of the problem as $O(K|\mathcal{L}|^T)$ where T is the size of the largest clique in the fused graph. For large T as the problem gets intractable, many well-known approximations can be used as discussed in Section 4. Also, we will see how this understanding will guide us to develop a faster and more accurate partitioning-based approximation (Sections 4.1, 4.2).

3.2 Training Algorithm

The overall training objective of Equation 5 is not necessarily concave in $\mathbf{w}_s$ because of the agreement term with sums within a log. One possibility is to use the EM algorithm over a variational approximation of the objective with extra variables [18, 12]. EM will give a local optima if the marginals of $\mathrm{P}_{\mathcal{A}}$ can be computed exactly. Since this cannot be guaranteed for general fused graphs, we also explore the simpler approach of gradient ascent. In Section 6 we show that gradient ascent achieves better accuracy than EM, while also being much faster.

The gradient of $\mathrm{LL}(\mathcal{Y}_{\mathcal{A}}, \mathbf{W})$ with respect to the parameter w_{st} of a node feature f_{st} in source s, is:

$$\frac{\partial \mathrm{LL}(\mathcal{Y}_{\mathcal{A}}, \mathbf{W})}{\partial w_{st}} =$$

$$\sum_{i \in U_s} \sum_{j=1}^{|\mathbf{X}_{si}|} \sum_{y} (\mu_{\mathcal{A},\pi(s,i,j)}(y|\mathbf{X}) - \mu_{s,j}(y|\mathbf{X}_{si})) f_{st}(\mathbf{X}_{si}, y, j)$$

where j varies over the tokens in $\mathbf{X}_{si}$, and $\mu_{s,j}, \mu_{\mathcal{A},j'}$ denote the marginal probability at variable j of P_s, and j' of $\mathrm{P}_{\mathcal{A}}$ respectively. The derivative with respect to an edge feature can be computed similarly. Thus if the feature f_{st} has the same expectation under the agreement model $\mathrm{P}_{\mathcal{A}}$ and the chain model P_s, its gradient is zero. Note that the E-step of EM requires the computation of the same kind of marginal variables. These marginals are computed using the same inference algorithms that compute the log-partition $Z_{\mathcal{A}}(\theta)$, and we discuss the various options next.

4. APPROXIMATIONS

In formulations where computing the objective and gradient leads to intractable inference, the standard approach in machine learning is to use approximate inference [15]. In general, any available sum-product inference algorithm like Belief Propagation (BP) or its convergent tree-reweighted versions (TRW) [20, 16] can be used to approximate $Z_{\mathcal{A}}(\theta)$ and the marginals required by the gradient. However, these typically need multiple iterations and can sometimes be slow to converge. [18] proposes an efficient one-step approximation that reduces to a single step of TRW [16] where the roles of trees are played by individual chains. As in all TRW algorithms, this method returns an upper bound of the log-partition value. We will show in Section 6 that accuracy-wise, this approximation is quite poor compared to BP. However with approximations such as BP, the convergence of the outer gradient ascent loop is quite slow. Another downside of these approaches is that there is no guarantee that the approximation leads to a valid probability distribution. For example, we often observed that the approximate value of $Z_{\mathcal{A}}(\theta)$ was greater than $\sum_{(s,i)} \log Z(\mathbf{X}_{si}, \mathbf{w}_s)$ causing $\mathrm{P}_{\mathcal{A}}$ to be greater than 1.

We therefore explored a second form of the objective where instead of enforcing *joint* agreement over all shared segments in $\mathcal{A}$, we partition $\mathcal{A}$ into smaller subsets $\mathcal{A}_1, \ldots, \mathcal{A}_R$ such that each $\Pr(\mathcal{Y}_{\mathcal{A}_k})$ is easy to compute, and $\cap_k \mathcal{Y}_{\mathcal{A}_k} = \mathcal{Y}_{\mathcal{A}}$. We then replace $\Pr(\mathcal{Y}_{\mathcal{A}})$ by $\prod_k \Pr(\mathcal{Y}_{\mathcal{A}_k})$, thus replacing the corresponding log-likelihood term by $\sum_k \mathrm{LL}(\mathcal{A}_k, \mathbf{W})$. Before we present our general strategy in Section 4.2 to partition $\mathcal{A}$, for simplicity we describe its very special case.

4.1 Per-segment Partitioning

This scheme partitions $\mathcal{A}$ by assigning each shared segment $C \in \mathcal{A}$ to its own partition. $G_{\mathcal{A}}$ is now replaced by several simpler graphs, each of which has its nodes fused only at one shared segment. Figures 2(c)-(f) illustrate this partitioning for the example of Figure 2(a). The probability $\Pr(\mathcal{Y}_{\{C\}})$ of agreement on occurrences of a single shared segment C simplifies to

$$\Pr(\mathcal{Y}_{\{C\}}) = \sum_{\mathbf{y} \in \mathbf{Y}_C} \prod_{(s,i,r) \in C} \mathrm{P}_s(\mathbf{Y}_{sir} = \mathbf{y} | \mathbf{X}_{si}, \mathbf{w}_s) \qquad (9)$$

where $\mathbf{Y}_C$ is set of all possible labelings for any occurrence of C, and $\mathrm{P}_s(\mathbf{Y}_{sir} = \mathbf{y}|.)$ is the marginal probability of the span r in record (s, i) taking the labeling $\mathbf{y}$ under P_s.

This partitioning is useful for two reasons. First, since C spans contiguous tokens, the fused graph of $\Pr(\mathcal{Y}_{\{C\}})$ is always a tree, e.g. the ones in Figures 2(c)-(f). Second, since for trees we can use sum-product to compute $\Pr(\mathcal{Y}_{\{C\}})$ instead of Equation 9, we can use arbitrarily long shared segments instead of choosing unigram shared segments, which is usually the norm in extraction applications.

We found that empirically this partitioning was more accurate than doing approximate inference on the globally fused graph $G_{\mathcal{A}}$, while being significantly faster. However the absolute runtime of this approach is still high since it creates as many repetitions of a chain as the number of shared segments in which it appears (e.g. Figures 2(c)-(f)). We next present a smarter partitioning strategy that directly minimizes total runtime, while ensuring that the fused graph of each partition remains a tree where inference is tractable.

4.2 Tree-based Partitioning

Our goal[2] is to partition $\mathcal{A}$ into disjoint sets $\mathcal{A}_1, \ldots, \mathcal{A}_R$ (R unknown) and compute $\prod_{i=1}^{R} \Pr(\mathcal{Y}_{\mathcal{A}i})$ instead of $\Pr(\mathcal{Y}_{\mathcal{A}})$. The key desiderata for a good partitioning are: (a) For every partition $\mathcal{A}_i$, its fused graph $G_{\mathcal{A}_i}$ is a tree (for tractability) (b) Total number of nodes across all $G_{\mathcal{A}_i}$ should be small, as the runtime is linear in it, and (c) Two or more shared segments that overlap or are adjacent in many chains should ideally be in the same partition to preserve correlation of agreement during training.

The per-segment partitioning of Section 4.1 passes only the first criteria and is the worst with regards to the other two. In contrast an unpartitioned $\mathcal{A}$ fulfills the last two criteria trivially but not the first. So we need a partitioning somewhere between these two extremes.

We observe that the last two criteria can be satisfied with just one metric — minimizing the number of nodes in every $G_{\mathcal{A}_i}$. This is because when two (or more) shared segments that occur in many common chains go in the same partition, they reuse a lot of nodes in the fused graph. Thus minimizing the number of nodes forces us to club together such segments in the same partition. Keeping this in mind, we write our partitioning goal as:

$$\min_{R, \mathcal{A}_1, \ldots, \mathcal{A}_R} \sum_{i=1}^{R} \#\mathrm{nodes}(G_{\mathcal{A}_i}) \qquad (10)$$
$$\text{s.t. Each } G_{\mathcal{A}_i} \text{ is a tree } \ \forall i = 1, \ldots, R$$

We now show that this objective is NP-hard.

Theorem 4.1. *The objective in Equation 10 is NP-hard.*

[2]Tree-based partitioning should not be confused with Tree re-weighted message passing (TRW) [16] — the former partitions $\mathcal{A}$ to approximate the objective with a product of tractable probabilities, whereas TRW approximates the partition function that appears in the numerator of the original objective, as discussed in the beginning of Section 4.

PROOF. We reduce the following NP-hard problem to our objective: Given a graph H, partition its vertices so that each part induces a forest and the number of edges with both endpoints in the same part is maximized.

The reduction is as follows. Each edge $e = (u, v) \in H$ defines a three-node chain $u - e - v$. Each vertex u defines a shared segment of length 1, and node e is not in any shared segment. Thus a vertex set in H corresponds to a set of segments, and it will induce a forest iff the segment set's fused graph is a forest.

Also, for a vertex set V in H, the number of nodes in the fused graph of the corresponding segment set is just three times the number of chains, which in turn is (sum of degrees of vertices in V - number of edges in the forest induced by V). Summing up, the total number of nodes is just 6 * number of edges in H - 3 * total number of edges with both endpoints in the same set. Thus minimizing the number of nodes is the same as maximizing the number of edges of H with endpoints in the same part. □

Algorithm 1 PartitionAgreementSet($\mathcal{A}$)

```
input $\mathcal{A}$
output a partition of $\mathcal{A}$
  $\mathcal{P} \leftarrow$ Each segment in its own set
  Initialize $G_{\{C\}}$ for each $\{C\} \in \mathcal{P}$
  {Phase One}
  repeat
    Find $\mathcal{A}_i, \{C\} \in \mathcal{P}$ such that MergeValid($\mathcal{A}_i, C$) and which maximize node reduction
    if $\mathcal{A}_i, \{C\}$ found then
      $\mathcal{P} \leftarrow \mathcal{P} \setminus \{C\}$
      $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{C\}$
      Update the data-structures in $G_{\mathcal{A}_i}$
    end if
  until change
  {Phase Two}
  repeat
    Find $\mathcal{A}_j, \mathcal{A}_k \in \mathcal{P}$ such that OneChainCommon($\mathcal{A}_j, \mathcal{A}_k$) and which maximize node reduction
    if $\mathcal{A}_j, \mathcal{A}_k$ found then
      $\mathcal{A}_j \leftarrow \mathcal{A}_j \cup \mathcal{A}_k$
      $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{A}_k$
    end if
  until change
return $\mathcal{P}$
```

We now present a greedy partitioning algorithm (Algorithm 1) to solve the objective in Equation 10. The algorithm works in two phases. Phase one begins with the per-segment partitioning of Section 4.1 and merges adjacent or overlapping segments as follows. In each round the algorithm picks a set $\mathcal{A}_i$ in the partition, a shared segment $C \notin \mathcal{A}_i$, and adds C to $\mathcal{A}_i$ if $G_{\mathcal{A}_i \cup \{C\}}$ is a tree and C is adjacent to at least one segment in $\mathcal{A}_i$. $\mathcal{A}_i$ and C are greedily chosen to minimize #nodes($G_{\mathcal{A}_i \cup \{C\}}$) $-$ #nodes($G_{\mathcal{A}_i}$) $-$ #nodes($G_{\{C\}}$). For example, consider Figure 2(a), and let $\mathcal{A}_i$ = {Matt Groening}. We can add the segment "Matt Groening , The Simpsons" to it because it does not create a cycle. But, we cannot add "Simpsons" to $\mathcal{A}_i$. It is easy to see that the partitioning after phase one will be {{"Matt Groening", "Matthew Matt Groening", "Matt Groening , Simpsons"}, {"Simpsons"}} whose fused models are trees, as shown in Figures 2(g) and 2(h).

When we cannot do any merges, the algorithm moves to phase two. In a round in phase two, the algorithm picks two sets $\mathcal{A}_j, \mathcal{A}_k$ and merges them if the segments in $\mathcal{A}_j$ and $\mathcal{A}_k$ have exactly one common chain, i.e. $|\{(s, i) \mid \exists C_1 \in \mathcal{A}_j, C_2 \in \mathcal{A}_k : (s, i, .) \in C_1 \wedge (s, i, .) \in C_2\}| = 1$. Merging two fused trees with only one common chain cannot introduce a cycle so $G_{\mathcal{A}_j \cup \mathcal{A}_k}$ will be a tree. Again the set pair to be merged is chosen using the greedy criteria.

The key step in the algorithm is checking for adjacency and cycle creation in phase one. Algorithm 2 describes this check for adding a shared segment C to partition $\mathcal{A}_i$. It maintains the fused graph $G_{\mathcal{A}_i}$ (which is a tree since it is acyclic) for $\mathcal{A}_i$. Given C, it locates its occurrences in $G_{\mathcal{A}_i}$ and sees if they can be fused without introducing a cycle. This is true if every pair of occurrences has the same common ancestor in $G_{\mathcal{A}_i}$, and either this ancestor is adjacent to all the occurrences, or is inside all the occurrences. Algorithm 2 describes this check, which in practice is done using tree branch information in $G_{\mathcal{A}_i}$. To illustrate, consider the fused graph in Figure 2(e). In this graph, the three occurrences of "Matt Groening" are at distances 1,2,2 from the common ancestor (the fused Simpsons node) so the check fails. In contrast, both the occurrences of "Matt Groening , The Simpsons", contain their common ancestor (again, the Simpsons node), so the check succeeds.

The main merit of the two-phase algorithm is that, by delaying the merging of non-adjacent segments to phase two, the bottom-up algorithm is able to assign higher priority to the merging of adjacent segments.

Algorithm 2 MergeValid($\mathcal{A}_i, C$)

```
input Set of shared segments $\mathcal{A}_i$, shared segment $C$
output Boolean check for mergeability of $C$ and $\mathcal{A}_i$.
  $r_j \leftarrow j^{th}$ occurrence span of $C$ in $G_{\mathcal{A}_i}$, $\forall j$
  $u \leftarrow$ Unique lowest ancestor node in $G_{\mathcal{A}_i}$ of every pair $(r_j, r_{j'})$
  {u may not exist or u may be inside a span(s)}
  if $u$ exists and ($u$ is adjacent to every $r_j$ or $u$ is inside every $r_j$) then
    return true {adjacent and mergeable}
  end if
return false
```

5. GENERATING THE AGREEMENT SET

We now discuss principled generation of a clean agreement set $\mathcal{A}$. Generating a clean $\mathcal{A}$ is highly important; we shall see in Section 6 that even the best collective training schemes can get derailed by noisy agreement sets. This is analogous to the impact of a good neighborhood graph on the accuracy of semi-supervised learning [14].

Traditional collective extraction methods have not focused on the process of finding quality agreement sets. These methods simply announce arbitrary repetitions of a unigram as a shared segment [22, 11, 17]. This is inadequate because of two reasons. First, unigram segments cannot transfer any strong first order dependencies across records and sources. Longer shared segments are essential for that. Second, blindly using repetitions of a token/n-gram as a shared segment can inject a lot of noise in the agreement set. For example, enforcing agreement on two random occurrences of "America" is not advisable as their true labels might be different (Organization as in Bank of America, vs Place as in United States of America).

Instead we present a more principled strategy for generating agreement sets. We make the working assumption that significant content overlap across sources is caused by approximate duplication of records, modulo any differences in style. This is predominantly true in organically created HTML lists on the web. Our approach works in two stages (pseudo code as Algorithm 3).

The first stage clusters records into groups of approximate duplicates. In the second stage, for each group, we find maximally long segments that repeat among records inside that group, and output these as shared segments. By limiting ourselves to generating shared segments among similar records, we increase the chance that segment occurrences refer to the same field(s) and should be labeled similarly.

Our algorithm for computing the groups in the first stage is also different from conventional clustering and is designed to exploit our observed pattern of duplicates within and across sources. First, we group together any almost-duplicate records within a source. Next, we merge record groups across sources using multi-partite matching. Since this is NP-hard, we employ the following phased scheme: First, we order the sources using a natural criteria such as maximizing pairwise similarity with adjacent sources. Each record group in the first source forms a cluster. In phase s, we find a bipartite matching between source $s+1$ and the clusters formed by the first s sources. The edge weight between clusters g and g' is the best similarity score between any record of g and any record of g'. Similarity is measured by a user-specified function like Jaccard or TF-IDF. A record group in source $s+1$ is assigned to the cluster to which it is matched. Unmatched record groups form new clusters.

We will empirically show that this two-stage method finds more accurate shared segments than the traditional unigram generation methods mentioned earlier.

Algorithm 3 Generating the agreement set $\mathcal{A}$

```
input $\{\mathbf{X}_{si} : \forall s,\ s = \text{source}, i \in U_s\}$
output $\mathcal{A}$ = set of shared segments
  Set of clusters $\mathcal{D}$ = empty, $\mathcal{A}$ = empty.
  {First stage}
  for $s = 1 \ldots S$ do
    $\mathcal{D}_s$ = Cluster records of source $s$ to collapse duplicates.
    Find bipartite match between record groups in $\mathcal{D}$ & $\mathcal{D}_s$.
    Update $\mathcal{D}$ by merging matched groups and creating new clusters for unmatched groups in $\mathcal{D}_s$.
  end for
  {Second stage}
  for record group $g \in \mathcal{D}$ do
    Add to $\mathcal{A}$ all maximal segments that repeat across records in $g$
  end for
```

6. EXPERIMENTAL EVALUATION

We present extensive experiments over several extraction domains covering a rich diversity of data characteristics.

Task: We evaluate our framework on the table materialization task of [13]. The task takes a user-provided query table containing only a few rows (e.g. top table in Figure 1), and up to 20 potentially relevant HTML lists, and converts every list record into a structured row of interest to the user. The 20 lists are chosen by using the cells of the query table to probe an index of 16M lists obtained from a 500M webpage crawl. The query rows are also used to generate labeled records for each list[3]. We use *dataset* to denote a query table and its list sources.

Table 1: Properties of the datasets

Dataset Group	#	#Srcs	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{A}\rvert$	Records	Base error	$\lvert\mathcal{A}\rvert$ Noise
50F	2	9	4.0	23	75	44.8	0.10
50M	3	11	4.3	202	223	45.4	0.11
40F	4	6	3.5	147	409	33.1	0.07
40M	4	14	4.5	235	344	32.7	0.12
30F	3	9	5.3	146	346	26.5	0.35
30M	14	10	4.4	413	336	23.9	0.21
20F	9	14	4.0	172	575	14.4	0.04
20M	7	13	4.0	959	831	13.4	0.11
10F	6	10	4.0	154	440	5.7	0.04
10M	6	15	4.0	436	493	3.9	0.13
All	58	11	4.2	348	451	16.7	0.15
Std	0	5.8	1.1	500	432	12.24	0.14

Datasets: We use a set of 58 datasets from [13] [4]. These cover various tables of interest, e.g. Oil spills, University mottos, Caldecott medal winners, Supreme court cases etc. Figure 1 shows three lists from the Oil Spill dataset.

Our ground truth consists of every list token manually labeled with a relevant dataset-specific label. We measure *extraction error* as 100-F1 accuracy of the extracted entities. We measure *noise* in an agreement set $\mathcal{A}$ as the percentage of shared segments which have any one member disagreeing with the rest in their ground truth label.

Settings: We experiment with two query sizes – 3 and 7 seed rows, simulating limited supervision in our task. Our numbers are averaged over five random selections of the seed rows. We set λ in Equation 5 using a validation set. Our base model is a CRF (`http://crf.sourceforge.net`), one per list as stressed before. We use standard context features over the neighborhood of a word including HTML tags and punctuations, along with class prior and edge features.

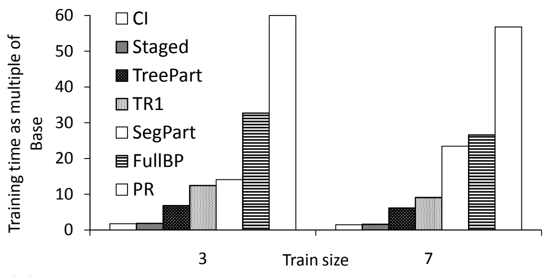
Our corpus of 58 real datasets forms an interesting benchmark, as it spans more than 600 lists, and exhibits a wide spectrum in terms of error rate of the base models, number of HTML lists, number of shared segments per record, and noise in the agreement set. For ease of presentation, we cluster these 58 datasets into ten groups using a paired criteria — error of the base models and relative size of the agreement sets. We create five bins for errors: 40–50%, 30–40%, and so on, and two bins for agreement set: "M" (many) when there are more than 0.5 shared segments per record and "F" (few) otherwise. Table 1 lists for each of the ten groups: the number of datasets (#), average number of sources (#Srcs), number of labels ($|\mathcal{L}|$), number of shared segments ($|\mathcal{A}|$), number of records, error of base models, and percentage of noisy segments in $\mathcal{A}$. The last row in the table that lists the standard deviation of these values over all 58 sources illustrates the diversity of the datasets.

Methods: We compare the following methods on extraction error and training time.

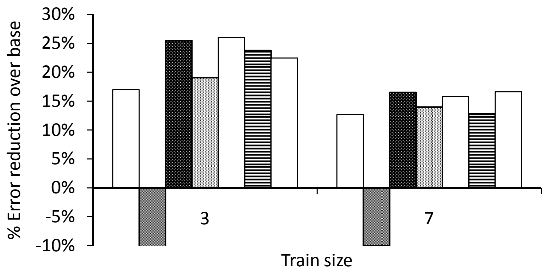
1. Base: Baseline that independently trains each CRF.
2. Four prior methods discussed in Section 2:

[3]Labeled data generation is non-trivial [13] but not relevant to this paper so we omit the details.

[4]The original task had 65 datasets. We dropped those whose ground truth we could not independently re-verify.



(a) Ratio of running time with Base which takes 11 and 17 seconds respectively for 3 and 7 training records.



(b) % reduction in error over Base which has error 16.7% and 12.7% respectively for 3 and 7 training records

Figure 3: Comparing different methods of exploiting overlap on running time and accuracy. The legends are shared across the two bar charts and appear in the left to right order.

   (a) CI: Collective inference at deployment after independently training each model.
   (b) Staged: Sequentially transferring labels from trained to untrained sources via shared segments.
   (c) PR: The EM-based posterior regularization method for multiview learning [12].
   (d) TR1: The agreement maximizer of [18].

3. Our partitioning-based approach with two options:
   (a) SegPart: Per-segment partitioning of Section 4.1.
   (b) TreePart: Tree-based partitioning of Section 4.2.
4. FullBP: Belief Propagation based approximate inference on the global fused graph $G_{\mathcal{A}}$ (Section 4).

For TR1, TreePart, SegPart, and FullBP, we also perform collective inference after collective training.

6.1 Benefit of Collective Training

In Figures 3(a) and 3(b) we compare all the methods on their total runtime and relative error reduction over Base. Table 2 breaks down these numbers on individual dataset groups. We make various observations from these graphs:

First, observe that many schemes give significant error reduction over Base, asserting that overlap in unlabeled data is useful when labeled data is limited. Starting with three training records, Base needs four more records to reduce the error by 25% (16.7% to 12.7%). In contrast, our collective training methods (TreePart, SegPart and FullBP) achieve 26% error reduction by exploiting overlap in unlabeled data. This shows that proper overlap exploitation can compensate for limited supervision in open-domain extraction tasks.

Collective inference (CI) reduces error by 17% and 13% overall for three and seven training records respectively. In contrast, Staged overall performs worse than Base and shows large swings in errors across datasets. It is highly sensitive to the ordering of sources, and the hard label-transfer often cascades errors to all downstream sources.

SegPart provides the largest error reduction over Base of 27% and 19% respectively for three and seven training records, and TreePart is a close second, followed by FullBP. These methods are superior to CI mainly because CI cannot benefit records with no shared segments. This superiority persists even if we do not do collective inference after collective training. For example, SegPart without CI (denoted "SegPart (No CI)" in Table 2) still provides an error reduction of 23.5% for three training records. This shows that collective inference is best used in tandem with collective training and not alone. We also observe that the gains of collective training are prominent for datasets with large agreement sets, and base errors in the 15-35% range.

The TR1 method (Table 2) achieves a training time close to TreePart and SegPart. However, its error reduction over Base is only 19% as compared to >25% for SegPart and TreePart. This is expected as TR1 performs only one round of message passing for the sake of speed. However the error reduction of TR1 without collective inference (denoted "TR1 (No CI)" in Table 2) is quite small at 9% and 4% for three and seven training records respectively. This shows that TR1 does not train sufficiently good joint models and most of its limitations are masked by collective inference. In contrast SegPart and TreePart provide significant benefits even without collective inference.

Figure 4(a) shows the spread of relative error reduction of TreePart vs the absolute error of Base for all datasets using three training records. This plot shows that there are many extraction tasks where TreePart achieves a big reduction in error (> 40%) even for small values of base error. Only four of the 58 cases see > 10% error increase after collective training, which can be explained by noise in the agreement sets. Figure 4(b) shows that the benefits of TreePart over Base decrease as agreement sets become noisier.

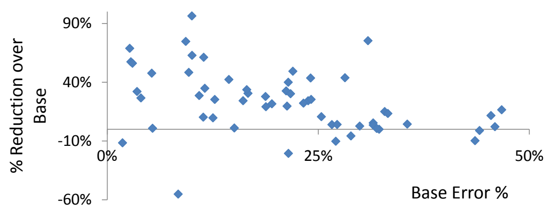
While SegPart, TreePart, and FullBP provide the best error reduction, TreePart has the best runtime (Figure 3(a)). For three training records, it is two and four times faster than SegPart and FullBP respectively, with larger gains for seven training records. This shows that SegPart creates too many partitions, while FullBP performs too many rounds of message passing. In terms of absolute runtime for three training records, Base takes 11 seconds on average per dataset and TreePart takes 75 seconds while providing a 26% error reduction. Another advantage of TreePart over FullBP is that it is trivial to parallelize. We expect parallelized TreePart to be adequately fast for interactive querying.

6.2 Comparison with the PR Objective

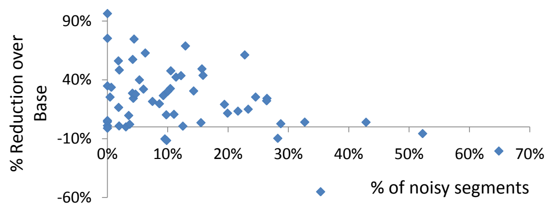
As discussed in Section 2, Posterior Regularization (PR) aims at minimizing the Bhattacharayya distance between the posteriors of the different views [12]. With only two sources s and s', and only one shared segment $\mathbf{c}$, their term is $\log \sum_{\mathbf{y_c}} \sqrt{P_s(\mathbf{y_c})P_{s'}(\mathbf{y_c})}$, $P_.(\mathbf{y_c})$ being the marginal of $\mathbf{c}$. This is maximized when the two marginals are identical. In contrast, our term of $\log \sum_{\mathbf{y_c}} P_s(\mathbf{y_c})P_{s'}(\mathbf{y_c})$ is maximized

Table 2: Percent error reduction over Base of various collective training and inference schemes.

Scheme	50F	50M	40F	40M	30F	30M	20F	20M	10F	10M	All 58
				Train size = 3 records							
Base Error	44.8	45.4	33.1	32.7	26.5	23.9	14.4	13.4	5.7	3.9	16.7
CI	1.7	3.2	10.4	3.3	-2.9	16.4	31.3	28.2	10.1	13.1	17.0
Staged	-25.4	6.1	-10.0	-11.9	29.3	-2.8	-75.3	23.1	32.5	-68.0	-10.2
TreePart	6.0	2.3	11.2	9.5	4.4	28.0	38.0	40.6	43.4	13.8	25.5
TreePart (No CI)	8.4	6.0	4.9	9.3	4.6	24.0	30.1	34.2	49.2	6.4	21.5
SegPart	6.6	0.6	14.3	9.8	4.5	31.5	38.8	42.7	36.2	9.3	**26.8**
SegPart (No CI)	8.4	4.9	7.5	10.2	5.2	27.0	30.5	38.3	48.6	9.2	23.5
FullBP	6.0	2.4	10.6	9.3	3.6	28.7	38.6	42.0	43.3	14.9	26.0
FullBP (No CI)	6.6	6.7	4.7	11.4	5.3	26.3	30.8	40.5	48.2	11.4	23.8
TR1	1.6	2.1	11.8	3.5	-3.1	18.6	34.3	35.0	13.2	-0.5	19.1
TR1 (No CI)	-1.3	3.9	3.0	0.1	-0.5	4.4	19.7	19.4	9.8	-5.9	8.9
PR	2.3	7.9	4.7	10.3	4.1	28.7	30.5	33.3	30.2	9.3	22.4
				Train size = 7 records							
Base Error	47.5	36.6	23.8	25.0	20.4	17.6	9.7	9.5	3.4	3.5	12.7
CI	6.6	7.3	10.4	2.1	1.3	13.4	23.7	16.6	5.7	13.7	12.7
Staged	-20.2	-0.6	-0.9	-13.2	11.3	-13.2	-39.5	6.0	2.1	-20.2	-10.0
TreePart	4.5	7.8	11.4	7.0	4.0	21.5	25.4	20.7	14.7	12.7	16.5
TreePart (No CI)	5.7	3.7	3.3	8.7	5.5	16.9	13.8	15.5	17.0	9.6	11.6
SegPart	9.3	11.1	16.7	5.5	4.4	22.6	25.3	25.0	16.1	15.2	**18.6**
SegPart (No CI)	10.0	7.8	7.8	7.4	6.6	19.4	16.4	28.2	18.3	15.4	16.0
FullBP	7.9	6.8	9.6	7.4	1.2	20.5	23.8	22.2	15.2	11.4	15.8
FullBP (No CI)	8.8	2.0	2.6	8.8	1.8	18.2	16.1	22.1	18.6	6.9	12.8
TR1	7.0	5.7	12.7	3.0	1.3	14.8	25.9	21.1	4.6	6.1	14.0
TR1 (No CI)	1.0	1.7	2.8	2.1	-0.4	2.4	6.1	9.3	1.4	5.8	4.0
PR	7.1	7.3	5.1	9.5	12.3	21.5	16.8	25.4	22.2	14.7	16.6

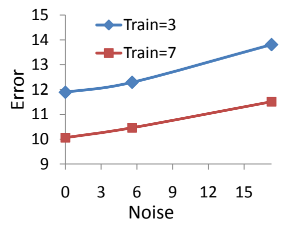


(a) Error reduction vs base error



(b) Error reduction vs % of noisy segments

Figure 4: % error reduction of TreePart over Base, plotted against (a) Base error (b) % of noisy segments in $\mathcal{A}$. Each point represents a dataset.



(a) Varying $\mathcal{A}$

Train size	3		7	
Noisy	Yes	No	Yes	No
TreePart	13.1	12.7	11.2	10.7
SegPart	12.8	12.5	10.6	10.0
FullBP	12.7	12.2	11.0	10.4

(b) With/Without noise

Figure 5: (a)F1-error of SegPart with single-token agreement sets of varying noise (b) Errors (%) of various methods with and without noisy segments.

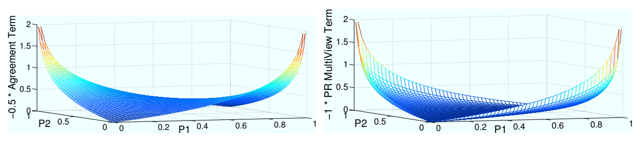


Figure 6: Comparing the agreement (left) and multiview (right) losses over two binomial posteriors.

when the marginals are identical *and peaked*. A maxima of our term is a maxima of their term but not vice versa. For example, starting with one peaked and one flat (i.e. weak) model, PR might make the peaked model flat as well, but our term will not and will strengthen the weak model instead. Figure 6.2 plots the two terms for two binomial posteriors. Empirically, PR is almost as accurate as TreePart and SegPart (Table 2). However, PR is eight times slower than TreePart, as its objective and gradient are quite hard to approximate and methods like Belief Propagation do not work with the Bhattacharayya distance. Hence EM is instead used, where E and M steps are iterative, with each iteration as costly as a gradient ascent step in TreePart.

6.3 Comparing Agreement Sets

We now compare our agreement set generation method (Section 5) vs the traditional method of choosing repeating unigrams as shared segments. For a fair comparison, we replace each of our shared segment of length two or more by single-token shared segments. This ensures that we compare only the quality of the unigrams. We also generate a third agreement set by removing all the noisy shared segments from our agreement set (using ground truth). Figure 5(a) shows the result of using SegPart with these three agreement sets. The traditional method generates 17.3% noisy segments (rightmost point in the two curves), whereas our method has a noise of only 5.6% (middle points). Further, the error with our agreement set is very close to that with the ideal noise-free agreement set, while the error with the traditional agreement set is significantly higher.

Table 5(b) shows the error of TreePart, SegPart, and FullBP with the original form of the agreement set, before and after removing noisy shared segments. We find that the limited noise in our agreement set causes only a slight error increase of less than 0.6% over the ideal scenario.

7. CONCLUSION

Open domain extraction on the Web throws up new challenges of limited supervision and new opportunities of content redundancy, not seen in classical extraction tasks. This paper addressed the challenge of training statistical extraction models jointly for multiple related Web sources to maximally exploit any content overlap. The basic premise of our framework was simple — maximizing the probability that the different sources agree on the labels of the overlapping content. However, many challenges arise while applying the premise on Web extraction tasks — designing a tractable training algorithm to exploit arbitrary patterns of overlap, scaling to many sources, and choosing a low-noise agreement set.

We equated the joint agreement likelihood to the log-partition of an aptly-defined graphical model. Instead of doing approximate inference on this graph, we designed an objective and an efficient algorithm for partitioning the agreement set into tree models. Through extensive experiments on diverse domains we showed that our method of partitioning the agreement set, coupled with our principled agreement set generation strategy provide the best trade-offs in terms of runtime and accuracy. More importantly, we have shown that a well-designed method of exploiting overlap can indeed compensate for the lack of labeled data. Future work includes parallelizing our algorithms and exploring the idea of exploiting overlap to other extraction tasks.

Acknowledgments. This work was partly supported by research grants from Yahoo! Research and IBM Faculty award.

References

[1] Google squared. http://www.google.com/squared, 2009.

[2] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plaintext collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000.

[3] D. Blei, D. Bagnell, and A. McCallum. Learning with scope, with application to information extraction and classification. In *UAI*, 2002.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[5] U. Brefeld, C. Büscher, and T. Scheffer. Multi-view hidden markov perceptrons. In *LWA*, 2005.

[6] R. Bunescu and R. J. Mooney. Collective information extraction with relational markov networks. In *ACL*, 2004.

[7] M. Cafarella, N. Khoussainova, D. Wang, E. Wu, Y. Zhang, and A. Halevy. Uncovering the relational web. In *WebDB*, 2008.

[8] A. Carlson, J. Betteridge, R. C. Wang, E. R. H. Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM*, 2010.

[9] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *EMNLP*, 1999.

[10] H. Elmeleegy, J. Madhavan, and A. Halevy. Harvesting relational tables from lists on the web. In *VLDB*, 2009.

[11] J. R. Finkel, T. Grenager, and C. D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

[12] K. Ganchev, J. Graça, J. Blitzer, and B. Taskar. Multi-view learning over structured and non-identical outputs. In *UAI*, 2008.

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

[14] T. Jebara, J. Wang, and S. Chang. Graph construction and b-matching for semi-supervised learning. In *ICML*, 2009.

[15] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009.

[16] V. Kolmogorov and M. J. Wainwright. On the optimality of tree-reweighted max-product message passing. In *UAI*, 2005.

[17] V. Krishnan and C. D. Manning. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *ACL-COLING*, 2006.

[18] P. Liang, D. Klein, and M. I. Jordan. Agreement-based learning. In *NIPS*, 2008.

[19] P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *HLT-NAACL*, 2006.

[20] T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms - a unifying view. In *UAI*, 2009.

[21] S. Sarawagi. Information extraction. *FnT Databases*, 1(3), 2008.

[22] C. Sutton and A. McCallum. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, 2004.

[23] B. Taskar, M. F. Wong, and D. Koller. Learning on the test data: Leveraging unseen features. In *ICML*, 2003.