New Regularized Algorithms for Transductive Learning

Partha Pratim Talukdar and Koby Crammer

Computer & Information Science Department
University of Pennsylvania
Philadelphia, PA 19104
{partha,crammer}@cis.upenn.edu

Abstract. We propose a new graph-based label propagation algorithm for transductive learning. Each example is associated with a vertex in an undirected graph and a weighted edge between two vertices represents similarity between the two corresponding example. We build on Adsorption, a recently proposed algorithm and analyze its properties. We then state our learning algorithm as a convex optimization problem over multi-label assignments and derive an efficient algorithm to solve this problem. We state the conditions under which our algorithm is guaranteed to converge. We provide experimental evidence on various real-world datasets demonstrating the effectiveness of our algorithm over other algorithms for such problems. We also show that our algorithm can be extended to incorporate additional prior information, and demonstrate it with classifying data where the labels are not mutually exclusive.

Key words: label propagation, transductive learning, graph based semi-supervised learning.

1 Introduction

Supervised machine learning methods have achieved considerable success in a wide variety of domains ranging from Natural Language Processing, Speech Recognition to Bioinformatics. Unfortunately, preparing labeled data for such methods is often expensive and time consuming, while unlabeled data are widely available in many cases. This was the major motivation that led to the development of semi-supervised algorithms which learn from limited amounts of labeled data and vast amounts of freely available unannotated data.

Recently, graph based semi-supervised algorithms have achieved considerable attention [2, 7, 11, 14, 17]. Such methods represent instances as vertices in a graph with edges between vertices encoding similarities between them. Graph-based semi-supervised algorithms often propagate the label information from the few labeled vertices to the entire graph. Most of the algorithms tradeoff between *accuracy* (initially labeled nodes should retain those labels, relaxations allowed by some methods) with *smoothness* (adjacent vertices in the graph should be assigned similar labels). Most algorithms only output label information to the unlabeled data in a transductive setting, while some algorithms are designed for the semi-supervised framework and build a classification model which can be applied to out-of-sample examples.

Adsorption [1] is one such recently proposed graph based semi-supervised algorithm which has been successfully used for different tasks, such as recommending YouTube videos to users [1] and large scale assignment of semantic classes to entities within Information Extraction [13]. Adsorption has many desirable properties: it can perform multiclass classification, it can be parallelized and hence can be scaled to handle large data sets which is of particular importance for semi-supervised algorithms. Even though Adsorption works well in practice, to the best of our knowledge it has never been analyzed before and hence our understanding of it is limited. Hoping to fill this gap, we make the following contributions in this paper:

- We analyze the Adsorption algorithm [1] and show that there does not exist an objective function whose local optimization would be the output of the Adsorption algorithm.
- Motivated by this negative result, we propose a new graph based semi-supervised algorithm (Modified Adsorption, MAD), which shares Adsorption's desirable properties, yet with some important differences.
- We state the learning problem as an optimization problem and develop efficient (iterative) methods to solve it. We also list the conditions under which the optimization algorithm – MAD – is guaranteed to converge.
- The transition to an optimization based learning algorithm provides a flexible and general framework that enables us to specify a variety requirements. We demonstrate this framework using data with non-mutually exclusive labels, resulting in the Modified Adsorption for Dependent Labels (MADDL, pronounced *medal*) algorithm.
- We provide experimental evidence demonstrating the effectiveness of our proposed algorithm on various real world datasets.

2 Adsorption Algorithm

Adsorption [1] is a general algorithmic framework for transductive learning where the learner is often given a small set of labeled examples and a very large set of unlabeled examples. The goal is to label all the unlabeled examples, and possibly under the assumption of label-noise, also to relabel the labeled examples.

As many other related algorithms [17, 12, 5], Adsorption assumes that the learning problem is given in a *graph* form, where examples or instances are represented as nodes or vertices and edges code *similarity* between examples. Some of the nodes are associated with a pre-specified label, which is correct in the noise-free case, or can be subject to label-noise. Additional information can be given in the form of *weights* over the labels. Adsorption propagates label-information from the labeled examples to the entire set of vertices via the edges. The labeling is represented using a non-negative score for each label, with high score for some label indicating high-association of a vertex (or its corresponding instance) with that label. If the scores are additively normalized they can be thought of as a conditional distribution over the labels given the node (or example) identity.

More formally, Adsorption is given an undirected graph $G = (V, E, W)$, where a node $v \in V$ corresponds to an example, an edge $e = (a, b) \in V \times V$ indicates that the label of the two vertices $a, b \in V$ should be similar and the weight $W_{ab} \in \mathbb{R}_+$ reflects the strength of this similarity.

We denote the total number of examples or vertices by $n = |V|$, by n_l the number of examples for which we have prior knowledge of their label and by n_u the number of unlabeled examples to be labeled. Clearly $n_l + n_u = n$. Let $\mathcal{L}$ be the set of possible labels, their total number is denoted by $m = |\mathcal{L}|$ and without loss of generality we assume that the possible labels are $\mathcal{L} = \{1 \ldots m\}$. Each instance $v \in V$ is associated with two row-vectors $\mathbf{Y}_v, \hat{\mathbf{Y}}_v \in \mathbb{R}^m_+$. The lth element of the vector $\mathbf{Y}_v$ encodes the prior knowledge for vertex v. The higher the value of $\mathbf{Y}_{vl}$ the stronger we a-priori believe that the label of v should be $l \in \mathcal{L}$ and a value of zero $\mathbf{Y}_{vl} = 0$ indicates no prior about the label l for vertex v. Unlabeled examples have all their elements set to zero, that is $\mathbf{Y}_{vl} = 0$ for $l = 1 \ldots m$. The second vector $\hat{\mathbf{Y}}_v \in \mathbb{R}^m_+$ is the output of the algorithm, using similar semantics as $\mathbf{Y}_v$. For example, a high value of $\hat{\mathbf{Y}}_{vl}$ indicates that the algorithm believes that the vertex v should have the label l. We denote by $\mathbf{Y}, \hat{\mathbf{Y}} \in \mathbb{R}^{n \times m}_+$ the matrices whose rows are $\mathbf{Y}_v$ and $\hat{\mathbf{Y}}_v$ respectively. Finally, we denote by $\mathbf{0}_d$ the all-zeros row vector of dimension d.

2.1 Random-Walk View

The Adsorption algorithm can be viewed as a controlled random walk over the graph G. The control is formalized via three possible actions: `inject`, `continue` and `abandon` (denoted by *inj, cont, abnd*) with pre-defined probabilities $p_v^{inj}, p_v^{cont}, p_v^{abnd} \geq 0$ per vertex $v \in V$. Clearly their sum is unit: $p_v^{inj} + p_v^{cont} + p_v^{abnd} = 1$. To label any vertex $v \in V$ (either labeled or unlabeled) we initiate a random-walk starting at v facing three options: with probability p_v^{inj} the random-walk stops and return (i.e. *inject*) the pre-defined vector information $\mathbf{Y}_v$. We constrain $p_v^{inj} = 0$ for unlabeled vertices v. Second, with probability p_v^{abnd} the random-walk *abandons* the labeling process and return the all-zeros vector $\mathbf{0}_m$. Third, with probability p_v^{cont} the random-walk *continues* to one of v's neighbors v' with probability proportional to $W_{v'v} \geq 0$. Note that by definition $W_{v'v} = 0$ if $(v, v') \notin E$. We summarize the above process with the following set of equations. The transition probabilities are,

$$\Pr\left[v'|v\right] = \begin{cases} \dfrac{W_{v'v}}{\sum\limits_{u\,:\,(u,v)\in E} W_{uv}} & (v',v) \in E \\ 0 & \text{otherwise} \end{cases} . \tag{1}$$

The (expected) score $\hat{\mathbf{Y}}_v$ for node $v \in V$ is given by,

$$\hat{\mathbf{Y}}_v = p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times \sum_{v'\,:\,(v',v)\in E} \Pr\left[v'|v\right] \hat{\mathbf{Y}}_{v'} + p_v^{abnd} \times \mathbf{0}_m \ . \tag{2}$$

2.2 Averaging View

For this view we add a designated symbol called the dummy label denoted by $\nu \notin \mathcal{L}$. This additional label explicitly encodes ignorance about the correct label and it means that a dummy label can be used instead. Explicitly, we add an additional column to all

Algorithm 1 Adsorption Algorithm

```
Input:
- Graph:           $G = (V, E, W)$
- Prior labeling:  $\mathbf{Y}_v \in \mathbb{R}^{m+1}$ for $v \in V$
- Probabilities:   $p_v^{inj}, p_v^{cont}, p_v^{abnd}$ for $v \in V$
Output:
- Label Scores: $\hat{\mathbf{Y}}_v$ for $v \in V$
1: $\hat{\mathbf{Y}}_v \leftarrow \mathbf{Y}_v$ for $v \in V$ {Initialization}
2:
3: repeat
4:     $D_v \leftarrow \frac{\sum_u W_{uv} \hat{\mathbf{Y}}_u}{\sum_u W_{uv}}$ for $v \in V$
5:     for all $v \in V$ do
6:         $\hat{\mathbf{Y}}_v \leftarrow p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times D_v + p_v^{abnd} \times \mathbf{r}$
7:     end for
8: until convergence
```

the vectors defined above, and have that $\mathbf{Y}_v, \hat{\mathbf{Y}}_v \in \mathbb{R}_+^{m+1}$ and $\mathbf{Y}, \hat{\mathbf{Y}} \in \mathbb{R}_+^{n \times (m+1)}$. We set $\mathbf{Y}_{v\nu} = 0$, that is, a-priori no vertex is associated with the dummy label, and replace the zero vector $\mathbf{0}_m$ with the vector $\mathbf{r} \in \mathbb{R}_+^{m+1}$ where $\mathbf{r}_l = 0$ for $l \neq \nu$ and $\mathbf{r}_\nu = 1$. In words, if the random-walk is abandoned, then the corresponding labeling vector is zero for all true labels in $\mathcal{L}$, and an arbitrary value of unit for the dummy label ν. This way, there is always positive score for at least one label, the ones in $\mathcal{L}$ or the dummy label.

The averaging view then defines a set of *fixed-point* equations to update the predicted labels. A summary of the equations appears in Algorithm 1. The algorithm is run until convergence which is achieved when the label distribution on each node ceases to change within some tolerance value. Since Adsorption is memoryless, it scales to tens of millions of nodes with dense edges and can be easily parallelized [1].

Baluja et. al. [1] show that up to the additional dummy label, these two views are equivalent. It remains to specify the values of p_v^{inj}, p_v^{cont} and p_v^{abnd}. For the experiments reported in Section 6, we set their value using the following heuristics (adapted from Baluja et. al. [1]) which depends on a parameter β which we set to $\beta = 2$. For each node v we define two quantities: c_v and d_v and define

$$p_v^{cont} \propto c_v \quad ; \quad p_v^{inj} \propto d_v \; .$$

The first quantity $c_v \in [0, 1]$ is monotonically decreasing with the number of neighbors for node v in the graph G. Intuitively, the higher the value of c_v, the lower the number of neighbors of vertex v and higher the information they contain about the labeling of v. The other quantity $d_v \geq 0$ is monotonically increasing with the entropy (for labeled vertices), and in this case we prefer to use the prior-information rather than the computed quantities from the neighbors.

Specifically we first compute the entropy of the transition probabilities for each node,

$$H[v] = -\sum_u \Pr[u|v] \log \Pr[u|v] \ ,$$

and then pass it through the following monotonically decreasing function,

$$f(x) = \frac{\log \beta}{\log(\beta \ + \ e^x))} \ .$$

Note that $f(0) = log(\beta)/\log(\beta + 1)$ and that $f(x)$ goes to zero, as x goes to infinity. We define,

$$c_v = f\left(H[v]\right) \ .$$

Next we define,

$$d_v = \begin{cases} (1 - c_v) \times \sqrt{H[v]} & \text{the vertex v is labeled} \\ 0 & \text{the vertex v is unlabled} \end{cases}$$

Finally, to ensure proper normalization of p_v^{cont}, p_v^{inj} and p_v^{abnd}, we define,

$$z_v = \max(c_v + d_v, 1) \ ,$$

and

$$p_v^{cont} = \frac{c_v}{z_v} \quad ; \quad p_v^{inj} = \frac{d_v}{z_v} \quad ; \quad p_v^{\text{abnd}} = 1 - p_v^{cont} - p_v^{inj} \ .$$

Thus, abandonment occurs only when the continuation and injection probabilities are low enough. This is most likely to happen at unlabeled nodes with high degree. Once the random walk reaches such a node (v), the walk is terminated with probability p_v^{abnd}. This, in effect, prevents the Adsorption algorithm from propagating information through high degree nodes. We note that the probabilities p_v^{inj}, p_v^{cont} and p_v^{abnd} for node v may be set with heuristics other than the fan-out entropy heuristics shown above to suit specific application contexts.

3 Analysis of the Adsorption Algorithm

Our next goal is to find an objective function that the Adsorption algorithm minimizes. Our starting point is line 6 of Algorithm 1. We note that when the algorithm converges, both sides of the assignment operator equal each other before the assignment takes place. Thus when the algorithm terminates, we have for all $v \in V$:

$$\hat{\mathbf{Y}}_v = p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times \frac{1}{N_v} \sum_u W_{uv} \hat{\mathbf{Y}}_u + p_i^{\text{abnd}} \times \mathbf{r} \ ,$$

where

$$N_v = \sum_{v'} W_{v'v} \ .$$

The last set of equalities is equivalent to,

$$G_v\left(\{\hat{\mathbf{Y}}_u\}_{u\in V}\right) = 0 \text{ for } v \in V \ , \tag{3}$$

where we define,

$$G_v\left(\{\hat{\mathbf{Y}}_u\}_{u\in V}\right) = p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times \frac{1}{N_v}\sum_u W_{uv}\hat{\mathbf{Y}}_u + p_i^{abnd} \times \mathbf{r} - \hat{\mathbf{Y}}_v \ .$$

Now, if the Adsorption algorithm was minimizing some objective function (denoted by $\mathcal{Q}\left(\{\hat{\mathbf{Y}}_u\}_{u\in V}\right)$), the termination condition of Eq. (3) was in fact a condition on the vector of its partial derivatives where we would identify

$$G_v = \frac{\partial}{\partial \hat{\mathbf{Y}}_v}\mathcal{Q} \ . \tag{4}$$

Since the functions G_v are linear (and thus has continuous derivatives), necessary conditions for the existence of a function $\mathcal{Q}$ such that (4) holds is that the derivatives of G_v are symmetric [8], that is,

$$\frac{\partial}{\partial \hat{\mathbf{Y}}_v}G_u = \frac{\partial}{\partial \hat{\mathbf{Y}}_u}G_v \ .$$

Computing and comparing the derivatives we get,

$$\frac{\partial}{\partial \hat{\mathbf{Y}}_u}G_v = p_v^{cont}\left(\frac{W_{uv}}{N_v} - \delta_{u,v}\right) \neq p_u^{cont}\left(\frac{W_{vu}}{N_u} - \delta_{u,v}\right) = \frac{\partial}{\partial \hat{\mathbf{Y}}_v}G_u \ ,$$

which is true since in general $N_u \neq N_v$ and $p_v^{cont} \neq p_u^{cont}$. We conclude:

Theorem 1. *There does not exists a function $\mathcal{Q}$ with continuous second partial derivatives such that the Adsorption algorithm convergences when gradient of $\mathcal{Q}$ are equal to zero.*

In other words, we searched for a (well-behaved) function $\mathcal{Q}$ such that its local optimal would be the output of the Adsorption algorithm, and showed that this search will always fail. We use this negative results to define a new algorithm, which builds on the Adsorption algorithm and is optimizing a function of the unknowns $\hat{\mathbf{Y}}_v$ for $v \in V$.

4 New Algorithm: Modified Adsorption (MAD)

Our starting point is Sec. 2.2 where we assume to have been given a weighted-graph $G = (V, E, W)$ and a matrix $\mathbf{Y} \in \mathbb{R}_+^{n\times(m+1)}$ and are seeking for a labeling-matrix $\hat{\mathbf{Y}} \in \mathbb{R}_+^{n\times(m+1)}$. In this section it is more convenient to decompose the matrices $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ into their columns, rather than rows. Specifically, we denote by $\mathbf{Y}_l \in \mathbb{R}_+^n$ the lth column of $\mathbf{Y}$ and similarly by $\hat{\mathbf{Y}}_l \in \mathbb{R}_+^n$ the lth column of $\hat{\mathbf{Y}}$. We distinguish the rows and columns of the matrices $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ using their indices, the columns are indexed with the label index l, while the rows are indexed are with a vertex index v (or u).

We build on previous research [3, 17, 11] and construct an objective that reflects three requirements as follows. First, for the labeled vertices we like the output of the algorithm to be close to the a-priori given labels, that is $\mathbf{Y}_v \approx \hat{\mathbf{Y}}_v$. Second, for pair of vertices that are close according to the input graph, we would like their labeling to be close, that is $\hat{\mathbf{Y}}_u \approx \hat{\mathbf{Y}}_v$ if W_{uv} is large. Third, we want the output to be as uninformative as possible, this serves as additional regularization, that is $\hat{\mathbf{Y}}_v \approx \mathbf{r}$. We now further develop the objective in light of the three requirements.

We use the Euclidian distance to measure discrepancy between two quantities, and start with the first requirement above,

$$\begin{aligned}\sum_v p_v^{inj} \sum_l \left(\mathbf{Y}_{vl} - \hat{\mathbf{Y}}_{vl}\right)^2 = & \sum_l \sum_v p_v^{inj} \left(\mathbf{Y}_{vl} - \hat{\mathbf{Y}}_{vl}\right)^2 \\ = & \sum_l \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right)^\top \mathbf{S} \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right) ,\end{aligned}$$

where we define the diagonal matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ and $\mathbf{S}_{vv} = p_v^{inj}$ if vertex v is labeled and $\mathbf{S}_{vv} = 0$ otherwise. The matrix $\mathbf{S}$ captures the intuition that for different vertices we enforce the labeling of the algorithm to match the a-priori labeling with different extent.

Next, we modify the similarity weight between vertices to take into account the difference in degree of various vertices. In particular we define $\mathbf{W}^{'}_{vu} = p_v^{cont} \times \mathbf{W}_{vu}$. Thus, a vertex u will *not* be similar to a vertex v if either the input weights $\mathbf{W}_{vu}$ are low or the vertex v has a large-degree (p_v^{cont} is low). We write the second requirement as,

$$\begin{aligned}&\sum_{v,u} \mathbf{W}^{'}_{vu} \left\|\hat{\mathbf{Y}}_v - \hat{\mathbf{Y}}_u\right\|^2 = \sum_{v,u} \mathbf{W}^{'}_{vu} \sum_l \left(\hat{\mathbf{Y}}_{vl} - \hat{\mathbf{Y}}_{ul}\right)^2 = \sum_l \sum_{v,u} \mathbf{W}^{'}_{vu} \left(\hat{\mathbf{Y}}_{vl} - \hat{\mathbf{Y}}_{ul}\right)^2 \\ &= \sum_l \sum_v \left(\sum_u \mathbf{W}^{'}_{vu}\right) \|\hat{\mathbf{Y}}_{vl}\|^2 + \sum_l \sum_u \left(\sum_v \mathbf{W}^{'}_{vu}\right) \|\hat{\mathbf{Y}}_{ul}\|^2 - 2 \sum_l \sum_{u,v} \mathbf{W}^{'}_{vu} \hat{\mathbf{Y}}_{ul} \hat{\mathbf{Y}}_{vl} \\ &= \sum_l \hat{\mathbf{Y}}_l^\top \mathbf{L} \hat{\mathbf{Y}}_l ,\end{aligned}$$

where,

$$\mathbf{L} = \mathbf{D} + \bar{\mathbf{D}} - \mathbf{W}^{'} - \mathbf{W}^{'\top},$$

and $\mathbf{D}$, $\bar{\mathbf{D}}$ are $n \times n$ diagonal matrices with

$$\mathbf{D}_{vv} = \sum_u \mathbf{W}^{'}_{uv} \quad , \quad \bar{\mathbf{D}}_{vv} = \sum_u \mathbf{W}^{'}_{vu} .$$

Finally we define the matrix $\mathbf{R} \in \mathbb{R}_+^{n \times (m+1)}$ where the vth row of $\mathbf{R}$ equals $p_v^{abnd} \times \mathbf{r}$ (we define $\mathbf{r}$ in Sec 2.2). In other words the first m columns of $\mathbf{R}$ equal zero, and the last (m+1th column) equal the elements of p_v^{abnd}. The third requirement above is thus written as,

$$\sum_{vl} \left(\hat{\mathbf{Y}}_{vl} - \mathbf{R}_{vl}\right)^2 = \sum_l \left\|\hat{\mathbf{Y}}_l - \mathbf{R}_l\right\|^2 .$$

We combine the three terms above into a single objective (which we would like to minimize), giving to each term a different importance using the weights μ_1, μ_2, μ_3.

$$C(\hat{\mathbf{Y}}) = \sum_l \left[\mu_1 \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right)^\top \mathbf{S} \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right) \ + \ \mu_2 \hat{\mathbf{Y}}_l^T \, \mathbf{L} \, \hat{\mathbf{Y}}_l \ + \ \mu_3 \left|\left|\hat{\mathbf{Y}}_l - \mathbf{R}_l\right|\right|_2^2 \right] \quad (5)$$

The objective in Equation 5 is similar to the Quadratic Cost Criteria [3], with the exception that the matrices $\mathbf{S}$ and $\mathbf{L}$ have different constructions. We remind the reader that $\hat{\mathbf{Y}}_l, \mathbf{Y}_l, \mathbf{R}_l$ are the lth columns (each of size $n \times 1$) of the matrices $\hat{\mathbf{Y}}, \mathbf{Y}$ and $\mathbf{R}$ respectively.

4.1 Solving the Optimization Problem

We now develop an algorithm to optimize (5) similar to the quadratic cost criteria [3]. Differentiating Equation 5 w.r.t. $\hat{\mathbf{Y}}_l$ we get,

$$\begin{aligned} \frac{1}{2} \frac{\delta C(\hat{\mathbf{Y}})}{\delta \hat{\mathbf{Y}}_l} &= \mu_1 \mathbf{S}(\hat{\mathbf{Y}}_l - \mathbf{Y}_l) + \mu_2 \mathbf{L} \hat{\mathbf{Y}}_l + \mu_3 (\hat{\mathbf{Y}}_l - \mathbf{R}_l) \\ &= (\mu_1 \mathbf{S} + \mu_2 \mathbf{L} + \mu_3 \mathbf{I}) \hat{\mathbf{Y}}_l - (\mu_1 \mathbf{S} \mathbf{Y}_l + \mu_3 \mathbf{R}_l) \ . \end{aligned} \quad (6)$$

Differentiating once more we get,

$$\frac{1}{2} \frac{\delta C(\hat{\mathbf{Y}})}{\delta \hat{\mathbf{Y}}_l \delta \hat{\mathbf{Y}}_l} = \mu_1 \mathbf{S} + \mu_2 \mathbf{L} + \mu_3 \mathbf{I} \ ,$$

and since both $\mathbf{S}$ and $\mathbf{L}$ are symmetric and positive semidefinite matrices (PSD), we get that the Hessian is PSD as well. Hence, the optimal minima is obtained by setting the first derivative (i.e. Equation (6)) to 0 as follows,

$$(\mu_1 \mathbf{S} + \mu_2 \mathbf{L} + \mu_3 \mathbf{I}) \, \hat{\mathbf{Y}}_l = (\mu_1 \mathbf{S} \mathbf{Y}_l + \mu_3 \mathbf{R}_l) \ .$$

Hence, the new labels ($\hat{\mathbf{Y}}$) can be obtained by a matrix inversion followed by matrix multiplication. However, this can be quite expensive when large matrices are involved. A more efficient way to obtain the new label scores is to solve a set of linear equations using Jacobi iteration which we now describe.

4.2 Jacobi Method

Given the following linear system (in x)

$$\mathbf{M} x = b$$

the Jacobi iterative algorithm defines the approximate solution at the (t+1)th iteration given the solution at tth iteration as follows,

$$x_i^{(t+1)} = \frac{1}{\mathbf{M}_{ii}} \left(b_i - \sum_{j \neq i} \mathbf{M}_{ij} x_j^{(t)} \right) \ . \quad (7)$$

We apply the iterative algorithm to our problem by substituting $x = \hat{\mathbf{Y}}_l$, $\mathbf{M} = \mu_1\mathbf{S} + \mu_2\mathbf{L} + \mu_3\mathbf{I}$ and $b = \mu_1\mathbf{S}\mathbf{Y}_l + \mu_3\mathbf{R}_l$ in (7),

$$\hat{\mathbf{Y}}_{vl}^{(t+1)} = \frac{1}{\mathbf{M}_{vv}}\left(\mu_1(\mathbf{S}\mathbf{Y}_l)_v + \mu_3\mathbf{R}_{vl} - \sum_{u\neq v}\mathbf{M}_{vu}\hat{\mathbf{Y}}_{ul}^{(t)}\right) \tag{8}$$

Let us compute the values of $(\mathbf{S}\mathbf{Y}_l)_i$, $\mathbf{M}_{ij(j\neq i)}$ and $\mathbf{M}_{ii}$. First,

$$\mathbf{M}_{vu(v\neq u)} = \mu_1\mathbf{S}_{vu} + \mu_2\mathbf{L}_{vu} + \mu_3\mathbf{I}_{vu}\ .$$

Note that since $\mathbf{S}$ and $\mathbf{I}$ are diagonal, we have that $\mathbf{S}_{vu} = 0$ and $\mathbf{I}_{vu} = 0$ for $u \neq v$. Substituting the value of $\mathbf{L}$ we get,

$$\mathbf{M}_{vu(v\neq u)} = \mu_2\mathbf{L}_{vu} = \mu_2\left(\mathbf{D}_{vu} + \bar{\mathbf{D}}_{vu} - \mathbf{W}^{'}_{vu} - \mathbf{W}^{'}_{uv}\right)\ ,$$

and as before the matrices $\mathbf{D}$ and $\bar{\mathbf{D}}$ are diagonal and thus $\mathbf{D}_{vu} + \bar{\mathbf{D}}_{vu} = 0$. Finally, substituting the values of $\mathbf{W}^{'}_{vu}$ and $\mathbf{W}^{'}_{uv}$ we get,

$$\mathbf{M}_{vu(v\neq u)} = -\mu_2 \times (p_v^{cont} \times \mathbf{W}_{vu} + p_u^{cont} \times \mathbf{W}_{uv})\ . \tag{9}$$

We now compute the second quantity,

$$(\mathbf{S}\mathbf{Y}_l)_{vu} = \mathbf{S}_{vv}\mathbf{Y}_{vv} + \sum_{t\neq v}\mathbf{S}_{vt}\mathbf{Y}_{tu} = p_v^{inj} \times \mathbf{Y}_{vv}\ ,$$

where the second term equals zero since $\mathbf{S}$ is diagonal. Finally, the third term,

$$\begin{aligned}\mathbf{M}_{vv} &= \mu_1\mathbf{S}_{vv} + \mu_2\mathbf{L}_{vv} + \mu_3\mathbf{I}_{vv}\\ &= \mu_1 \times p_v^{inj} + \mu_2(\mathbf{D}_{vv} + \bar{\mathbf{D}}_{vv} - \mathbf{W}^{'}_{vv} - \mathbf{W}^{'}_{vv}) + \mu_3\\ &= \mu_1 \times p_v^{inj} + \mu_2\sum_{u\neq v}(p_v^{cont}\mathbf{W}_{vu} + p_u^{cont}\mathbf{W}_{uv}) + \mu_3\ .\end{aligned}$$

Plugging the above equations into (8) and using the fact that the diagonal elements of $\mathbf{W}$ are zero, we get,

$$\hat{\mathbf{Y}}_v^{(t+1)} = \frac{1}{\mathbf{M}_{vv}}\left(\mu_1 p_v^{inj}\,\mathbf{Y}_v + \mu_2\sum_u\left(p_v^{cont}\mathbf{W}_{vu} + p_u^{cont}\mathbf{W}_{uv}\right)\hat{\mathbf{Y}}_u^{(t)} + \mu_3\, p_v^{abnd}\,\mathbf{r}\right)\ . \tag{10}$$

We call the new algorithm MAD for Modified-Adsorption and it is summarized in Algorithm 2. Note that for graphs G that are invariant to permutations of the vertices, and setting $\mu_1 = 2 \times \mu_2 = \mu_3 = 1$, MAD reduces to the Adsorption algorithm.

4.3 Convergence

A sufficient condition for the iterative process of Equation (7) to converge is that $\mathbf{M}$ is strictly diagonally dominant [10], that is if,

$$|\mathbf{M}_{vv}| > \sum_{u\neq v}|\mathbf{M}_{vu}| \quad \text{for all values of } v$$

Algorithm 2 Modified Adsorption (MAD) Algorithm

```
Input:
- Graph:           $G = (V, E, W)$
- Prior labeling:  $\mathbf{Y}_v \in \mathbb{R}^{m+1}$ for $v \in V$
- Probabilities:   $p_v^{inj}, p_v^{cont}, p_v^{abnd}$ for $v \in V$
Output:
- Label Scores: $\hat{\mathbf{Y}}_v$ for $v \in V$
1: $\hat{\mathbf{Y}}_v \leftarrow \mathbf{Y}_v$ for $v \in V$ {Initialization}
2: $\mathbf{M}_{vv} \leftarrow \mu_1 \times p_v^{inj} + \mu_2 \sum_{u \neq v} (p_v^{cont} \mathbf{W}_{vu} + p_u^{cont} \mathbf{W}_{uv}) + \mu_3$
3: repeat
4:     $D_v \leftarrow \sum_u \left(p_v^{cont} \mathbf{W}_{vu} + p_u^{cont} \mathbf{W}_{uv}\right) \hat{\mathbf{Y}}_u$
5:     for all $v \in V$ do
6:         $\hat{\mathbf{Y}}_v \leftarrow \frac{1}{\mathbf{M}_{vv}} \left(\mu_1 \times p_v^{inj} \times \mathbf{Y}_v + \mu_2 \times D_v + \mu_3 \times p_v^{abnd} \times \mathbf{r}\right)$
7:     end for
8: until convergence
```

We have,

$$\begin{aligned} |\mathbf{M}_{vv}| - \sum_{u \neq v} |\mathbf{M}_{vu}| &= \mu_1 \times p_v^{inj} + \mu_2 \times \sum_{u \neq v} \left(p_v^{cont} \times \mathbf{W}_{vu} + p_u^{cont} \times \mathbf{W}_{uv}\right) + \mu_3 - \\ &\qquad \mu_2 \times \sum_{u \neq v} \left(p_v^{cont} \times \mathbf{W}_{vu} + p_u^{cont} \times \mathbf{W}_{uv}\right) \\ &= \mu_1 \times p_v^{inj} + \mu_3 \end{aligned} \tag{11}$$

Note that $p_v^{inj} \geq 0$ for all v and that μ_3 is a free parameter in (11). Thus we can guarantee a strict diagonal dominance (and hence convergence) by setting $\mu_3 > 0$.

5 Extensions: Non-Mutually Exclusive Labels

In many learning settings, labels are not mutually exclusive. For example, in hierarchical classification, labels are organized in a tree. In this section, we extend the MAD algorithm to handle dependence among labels. This can be easily done using our new formulation which is based on objective optimization. Specifically, we shall add additional terms to the objective for each pair of dependent labels. Let C be a $m \times m$ matrix where m is the number of labels (excluding the dummy label) as before. Each entry, $C_{ll'}$, of this matrix C represents the dependence or similarity among the labels l and l'. By encoding dependence in this pairwise fashion, we can capture dependencies among labels represented as arbitrary graphs. The extended objective is shown in Equation 12.

$$\begin{aligned} C(\hat{\mathbf{Y}}) = \sum_l \Bigg[& \mu_1 \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right)^\top \mathbf{S} \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right) + \mu_2 \hat{\mathbf{Y}}_l^T \mathbf{L} \hat{\mathbf{Y}}_l + \mu_3 \left|\left|\hat{\mathbf{Y}}_l - \mathbf{R}_l\right|\right|_2^2 \\ & + \mu_4 \sum_i \sum_{l,l'} C_{ll'} (\hat{\mathbf{Y}}_{il} - \hat{\mathbf{Y}}_{il'})^2 \Bigg] \end{aligned} \tag{12}$$

The last term in Equation 12 penalizes the algorithm if similar labels (as determined by the matrix C) are assigned different scores, with severity of the penalty controlled by μ_4. Now, analyzing the objective in Equation 12 in the manner outlined in Section 4, we arrive at the update rule shown in Equation 13.

$$\hat{\mathbf{Y}}_{vl}^{(t+1)} = \frac{1}{\mathbf{M}_{vv}^{l}} \left(\mu_1 p_v^{inj}\, \mathbf{Y}_{vl} + \mu_2 \sum_u \left(p_v^{cont}\mathbf{W}_{vu} + p_u^{cont}\mathbf{W}_{uv}\right) \hat{\mathbf{Y}}_{ul}^{(t)} + \right.$$
$$\left. \mu_3\, p_v^{abnd}\, \mathbf{r}_l + \mu_4 \sum_{l'} C_{ll'} \hat{\mathbf{Y}}_{vl'}^{(t)} \right) \quad (13)$$

where,

$$\mathbf{M}_{vv}^{l} = \mu_1 \times p_v^{inj} + \mu_2 \times \sum_{u \neq v} (p_v^{cont}\mathbf{W}_{vu} + p_u^{cont}\mathbf{W}_{uv}) + \mu_3 + \mu_4 \sum_{l'} C_{ll'}$$

Replacing Line 6 in MAD (Algorithm 2) with Equation 13, we end up with a new algorithm: Modified Adsorption for Dependent Labels (MADDL). In Section 6.4, we shall use MADDL to obtain smooth ranking for sentiment classification.

6 Experimental Results

We compare MAD with various state-of-the-art learning algorithms on two tasks, text classification (Sec. 6.1) and sentiment analysis (Sec. 6.2), and demonstrate its effectiveness. In Sec. 6.3, we also provide experimental evidence showing that MAD is quite insensitive to wide variation of values of its hyper-parameters. In Sec. 6.4, we present evidence showing how MADDL can be used to obtain smooth ranking for sentiment prediction, a particular instantiation of classification with non-mutually exclusive labels. For the experiments reported in this section involving Adsorption, MAD and MADDL, the a-priori label matrix $\mathbf{Y}$ was column-normalized so that all labels have equal overall injection score. Also, the dummy label was ignored during evaluation as its main role is to add regularization during learning phase only.

6.1 Text Classification

World Wide Knowledge Base (WebKB) is a text classification dataset widely used for evaluating transductive learning algorithms. Most recently, the dataset was used by Subramanya and Bilmes [11], who kindly shared their preprocessed complete WebKB graph with us. There are a total of $4,204$ vertices in the graph, with the nodes labeled with one of four categories: *course, faculty, project, student*. A K-NN graph is created from this complete graph by retaining only top K neighbors of each node, where the value of K is treated as a hyper-parameter.

We follow the experimental protocol in [11]. The dataset was randomly partitioned into four sets. A transduction set was generated by first selecting one of the four splits at random and then sampling n_l documents from it; the remaining three sets are used as the

Class	SVM	TSVM	SGT	LP	AM	Adsorption	MAD
course	46.5	43.9	29.9	45.0	**67.6**	61.1	67.5
faculty	14.5	31.2	42.9	40.3	42.5	**52.8**	42.2
project	15.8	17.2	17.5	27.8	42.3	**52.6**	45.5
student	15.0	24.5	56.6	51.8	55.0	39.8	**59.6**
average	23.0	29.2	36.8	41.2	51.9	51.6	**53.7**

Table 1. PRBEP for the WebKB data set with n_l = 48 training and 3148 testing instances. All results are averages over 20 randomly generated transduction sets. The last row is the macro-average over all the classes. MAD is the proposed approach. Results for SVM, TSVM, SGT, LP and AM are reproduced from Table 2 of [11].

test set for evaluation. This process was repeated 21 times to generate as many training-test splits. The first split was used to tune the hyper-parameters, with search over the following: $K \in \{10, 50, 100, 500, 1000, 2000, 4204\}$, $\mu_2, \mu_3 \in \{1e{-}8, 1e{-}4, 1e{-}2, 1, 10, 1e2, 1e3\}$. The value of μ_1 was set to 1 for this experiment. Both for Adsorption and MAD, the optimal value of K was $1,000$. Furthermore, the optimal value for the other parameters were found to be $\mu_2 = \mu_3 = 1$. As in previous work [11], we use Precision-Recall Break Even Point (PRBEP) [9] as the evaluation metric. Same evaluation measure, dataset and the same experimental protocol makes the results reported here directly comparable to those reported previously [11]. For easier readability, the results from Table 2 of Subramanya and Bilmes [11] are cited in Table 1 of this paper, comparing performance of Adsorption based methods (Adsorption and MAD) to many previously proposed approaches: SVM [6], Transductive-SVM [6], Spectral Graph Transduction (SGT) [7], Label Propagation (LP) [16] and Alternating Minimization (AM) [11]. The first four rows in Table 1 shows PRBEP for individual categories, with the last line showing the macro-averaged PRBEP across all categories. The MAD algorithm achieves the best performance overall (for n_l = 48).

Performance comparison of MAD and Adsorption for increasing n_l are shown in Figure 1. Comparing these results against Fig. 2 in Subramanya and Bilmes [11], it seems that MAD outperforms all other methods compared (except AM [11]) for all values of n_l. MAD performs better than AM for n_l = 48, but achieves second best solution for the other three values of n_l. We are currently investigating why MAD is best for settings with fewer labeled examples.

6.2 Sentiment Analysis

The goal of sentiment analysis is to automatically assign polarity scores to text collections, with a high score reflecting positive sentiment (user likes) and a low score reflecting negative sentiment (user dislikes). In this section, we report results on sentiment classification in the transductive setting. From Section 6.1 and [11], we observe that Label Propagation (LP) [16] is one of the best performing L2-norm based transductive learning algorithm. Hence, we compare the performance of MAD against Adsorption and LP.

For the experiments in this section, we use a set of $4,768$ user reviews from the electronics domain [4]. Each review is assigned one of the four scores: 1 (worst), 2,

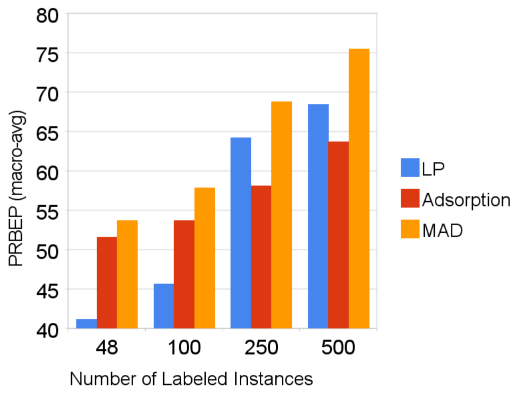


Fig. 1. PRBEP (macro-averaged) for the WebKB dataset with 3148 testing instances. All results are averages over 20 randomly generated transduction sets.

3, 4 (best). We create a K-NN graph from these reviews by using cosine similarity as the measure of similarity between reviews. We created 5 training-test splits from this data using the process described in Section 6.1. One split was used to tune the hyper-parameters while the rest were used for training and evaluation. Hyper-parameter search was carried over the following ranges: $K \in \{10, 100, 500\}$, $\mu_1 \in \{1, 100\}$, $\mu_2 \in \{1e{-}4, 1, 10\}$, $\mu_3 \in \{1e{-}8, 1, 100, 1e3\}$. Precision is used as the evaluation metric. Comparison of different algorithms for varying number of labeled instances are shown in Figure 2. From this, we note that MAD and Adsorption outperform LP, while Adsorption and MAD are competitive

6.3 Parameter Sensitivity

We evaluated the sensitivity of MAD to variations of its μ_2 and μ_3 hyper-parameters, with all other hyper-parameters fixed. We used a 2000-NN graph constructed from the WebKB dataset and a 500-NN graph constructed from the Sentiment dataset. In both cases, 100 nodes were labeled. We tried three values each for μ_2 and μ_3, ranging in at least 3 order of magnitude. For the WebKB, the PRBEP varied between $43.1 - 49.9$ and for the sentiment data, the precision varied in the range $31.4 – 36.4$ with $\mu_2 \leq \mu_3$ while precision dropped to 25 with $\mu_2 > \mu_3$. This underscores the need for regularization in these models, which is enforced with high μ_3. We note that in both cases the algorithm is less sensitive to the value of μ_2 than the value of μ_3. In general, we have found that setting μ_3 to one or two order magnitude more than μ_2 is a reasonable choice. We have also found that the MAD algorithm is quite insensitive to variations in μ_1. For example on the sentiment dataset, we tried two values for μ_1 ranging two order of magnitude, with other hyper-parameters fixed. In this case, precision varied in the range 36.2 - 36.3.

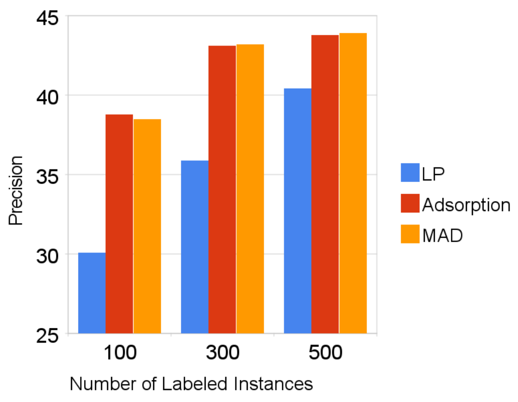


Fig. 2. Precision for the Sentiment Analysis dataset with 3568 testing instances. All results are averages over 4 randomly generated transduction sets.

	μ_4					
	0	1	10	100	1e3	1e4
Prediction Loss (L1) at rank 1	0.93	0.93	0.92	0.90	0.90	0.90
Prediction Loss (L1) at rank 2	1.21	1.20	1.12	0.96	0.97	0.97

Table 2. Average prediction loss at ranks 1 & 2 (for various values of μ_4) for sentiment prediction. All results are averaged over 4 runs. See Section 6.4 for details.

6.4 Smooth Ranking for Sentiment Analysis

We revisit the sentiment prediction problem in Section 6.2, but with the additional requirement that ranking of the labels (1, 2, 3, 4) generated by the algorithm should be smooth i.e. we prefer the ranking $1 > 2 > 3 > 4$ over the ranking $1 > 4 > 3 > 2$, where $3 > 2$ means that the algorithm ranks label 3 higher than label 2. The ranking $1 > 2 > 3 > 4$ is smoother as it doesn't involve rough transition $1 > 4$ which is present in $1 > 4 > 3 > 2$. We use the framework of stating requirements as an objective to be optimized. We use the MADDL algorithm of Sec. 5 initializing the matrix C as follows (assuming that labels 1 and 2 are related, while labels 3 and 4 are related):

$$C_{12} = C_{21} = 1 \quad , \quad C_{34} = C_{43} = 1$$

with all other entries in matrix C set to 0. Such constraints (along with appropriate μ_4 in Equation (12)) will force the algorithm to assign similar scores to dependent labels, thereby assigning them adjacent ranks in the final output. MAD and MADDL were then used to predict ranked labels for vertices on a 1000-NN graph constructed from the sentiment data used in Sec. 6.2, with 100 randomly selected nodes labeled. For this experiment we set $\mu_1 = \mu_2 = 1$, $\mu_3 = 100$. The $L1$-loss between the gold label and labels predicted at ranks $r = 1, 2$ for increasing values of μ_4 are given in Table 2. Note that, MADDL with $\mu_4 = 0$ corresponds to MAD. From Table 2 we observe that with

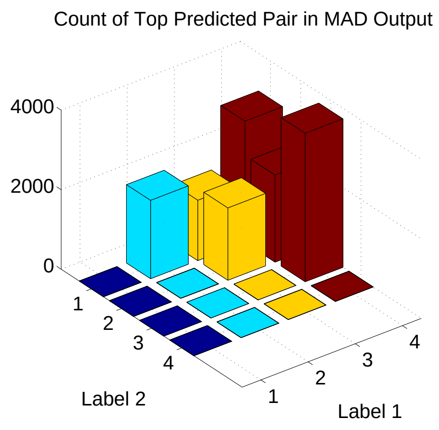


Fig. 3. Plot of counts of top predicted label pairs (order ignored) in MAD's predictions with $\mu_1 = \mu_2 = 1$, $\mu_3 = 100$.

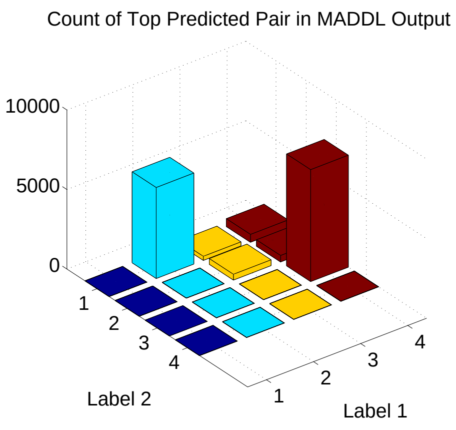


Fig. 4. Plot of counts of top label pairs (order ignored) in MADDL's predictions (Section 5), with $\mu_1 = \mu_2 = 1$, $\mu_3 = 100, \mu_4 = 1e3$.

increasing μ_4, MADDL is ranking at $r = 2$ a label which is related (as per C) to the top ranked label at $r = 1$, but at the same time maintain the quality of prediction at $r = 1$ (first row of Table 2), thereby ensuring a smoother ranking. From Table 2, we also observe that MADDL is insensitive to variations of μ_4 beyond a certain range. This suggests that μ_4 may be set to a (high) value and that tuning it may not be necessary.

Another view of the same phenomenon is shown in Fig. 3 and Fig. 4. In these figures, we plot the counts of top predicted label pair (order of prediction is ignored for better readability) generated by the MAD and MADDL algorithms. By comparing these two figures we observe that label pairs (e.g. (2,1) and (4,3)) favored by C (above) are more frequent in MADDL's predictions than in MAD's. At the same time, non-smooth predictions (e.g. (4, 1)) are virtually absent in MADDL's predictions while they are quite frequent in MAD's. These clearly demonstrate MADDL's ability to generate smooth predictions in a principled way, and more generally the ability to handle data with non-mutually exclusive or dependent labels.

7 Related Work

LP [16] is one of the first graph based semi-supervised algorithms. Even though there are several similarities between LP and MAD, there are important differences: (1) LP doesn't allow the labels on seeded nodes to change (while MAD does). As was pointed out previously [3], this can be problematic in case of noisy seeds. (2) There is no way for LP to express label uncertainty about a node. MAD can accomplish this by assigning high score to the dummy label. More recently, a KL minimization based algorithm was presented in [11]. Further investigation is necessary to determine the merits of each approach. For a general introduction to the area of graph-based semi-supervised learning, the reader is referred to a survey by Zhu [15].

8 Conclusion

In this paper we have analyzed the Adsorption algorithm [1] and proposed a new graph based semi-supervised learning algorithm, MAD. We have developed efficient (iterative) solution to solve our convex optimization based learning problem. We have also listed the conditions under which the algorithm is guaranteed to converge. Transition to an optimization based learning algorithm allows us to easily extend the algorithm to handle data with non-mutually exclusive labels, resulting in the MADDL algorithm. We have provided experimental evidence demonstrating effectiveness of our proposed methods. As part of future work, we plan to evaluate the proposed methods further and apply the MADDL method in problems with dependent labels (e.g. Information Extraction).

Acknowledgment

This research is partially supported by NSF grant #IIS-0513778. The authors would like to thank F. Pereira and D. Sivakumar for useful discussions.

References

1. S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *WWW*, 2008.
2. M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.
3. Y. Bengio, O. Delalleau, and N. Roux. Semi-Supervised Learning, chapter Label Propogation and Quadratic Criterion, 2007.
4. J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, 2007.
5. P. Indyk and J. Matousek. Low-distortion embeddings of finite metric spaces. *Handbook of Discrete and Computational Geometry*, 2004.
6. T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.
7. T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, 2003.
8. V. J. Katz. The history of stokes' theorem. *Mathematics Magazine*, 52(3):146–156, 1979.
9. V. Raghavan, P. Bollmann, and G. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM TOIS*, 7(3):205–229, 1989.
10. Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial Math., 2003.
11. A. Subramanya and J. Bilmes. Soft-Supervised Learning for Text Classification. In *EMNLP*, 2008.
12. M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. *NIPS*, 2002.
13. P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira. Weakly supervised acquisition of labeled class instances using graph random walks. In *EMNLP*, 2008.
14. J. Wang, T. Jebara, and S. Chang. Graph transduction via alternating minimization. In *ICML*, 2008.
15. X. Zhu. Semi-supervised learning literature survey. 2005.
16. X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, CMU CALD tech report, 2002.
17. X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *ICML*, 2003.

Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks

Partha Pratim Talukdar[*]
University of Pennsylvania
Philadelphia, PA 19104
partha@cis.upenn.edu

Joseph Reisinger[*]
University of Texas at Austin
Austin, TX 78712
joeraii@cs.utexas.edu

Marius Paşca
Google Inc.
Mountain View, CA 94043
mars@google.com

Deepak Ravichandran
Google Inc.
Mountain View, CA 94043
deepakr@google.com

Rahul Bhagat[*]
USC Information Sciences Institute
Marina Del Rey, CA 90292
rahul@isi.edu

Fernando Pereira
Google Inc.
Mountain View, CA 94043
pereira@google.com

Abstract

We present a graph-based semi-supervised label propagation algorithm for acquiring open-domain labeled classes and their instances from a combination of unstructured and structured text sources. This acquisition method significantly improves coverage compared to a previous set of labeled classes and instances derived from free text, while achieving comparable precision.

1 Introduction

1.1 Motivation

Users of large document collections can readily acquire information about the instances, classes, and relationships described in the documents. Such relations play an important role in both natural language understanding and Web search, as illustrated by their prominence in both Web documents and among the search queries submitted most frequently by Web users (Jansen et al., 2000). These observations motivate our work on algorithms to extract instance-class information from Web documents.

While work on named-entity recognition traditionally focuses on the acquisition and identification of instances within a small set of coarse-grained classes, the distribution of instances within query logs indicates that Web search users are interested in a wider range of more fine-grained classes. Depending on prior knowledge, personal interests and immediate needs, users submit for example medical queries about the symptoms of *leptospirosis* or the treatment of *monkeypox*, both of which are instances of *zoonotic diseases*, or the risks and benefits of *surgical procedures* such as *PRK* and *angioplasty*. Other users may be more interested in *African countries* such as *Uganda* and *Angola*, or *active volcanoes* like *Etna* and *Kilauea*. Note that *zoonotic diseases*, *surgical procedures*, *African countries* and *active volcanoes* serve as useful class labels that capture the semantics of the associated sets of class instances. Such interest in a wide variety of specific domains highlights the utility of constructing large collections of fine-grained classes.

Comprehensive and accurate class-instance information is useful not only in search but also in a variety of other text processing tasks including co-reference resolution (McCarthy and Lehnert, 1995), named entity recognition (Stevenson and Gaizauskas, 2000) and seed-based information extraction (Riloff and Jones, 1999).

1.2 Contributions

We study the acquisition of open-domain, labeled classes and their instances from both structured and unstructured textual data sources by combining and ranking individual extractions in a principled way with the Adsorption label-propagation algorithm (Baluja et al., 2008), reviewed in Section 3 below.

A collection of labeled classes acquired from text (Van Durme and Paşca, 2008) is extended in two ways:

1. Class label coverage is increased by identifying additional class labels (such as *public agencies* and *governmental agencies*) for existing

[*]Contributions made during internships at Google.

instances such as *Office of War Information*),

2. The overall instance coverage is increased by extracting additional instances (such as *Addison Wesley* and *Zebra Books*) for existing class labels (*book publishers*).

The WebTables database constructed by Cafarella et al. (2008) is used as the source of additional instances. Evaluations on gold-standard labeled classes and instances from existing linguistic resources (Fellbaum, 1998) indicate coverage improvements relative to that of Van Durme and Paşca (2008), while retaining similar precision levels.

2 First Phase Extractors

To show Adsorption's ability to uniformly combine extractions from multiple sources and methods, we apply it to: 1) high-precision open-domain extractions from free Web text (Van Durme and Paşca, 2008), and 2) high-recall extractions from WebTables, a large database of HTML tables mined from the Web (Cafarella et al., 2008). These two methods were chosen to be representative of two broad classes of extraction sources: free text and structured Web documents.

2.1 Extraction from Free Text

Van Durme and Paşca (2008) produce an open-domain set of instance clusters $C \in \mathcal{C}$ that partitions a given set of instances $\mathcal{I}$ using distributional similarity (Lin and Pantel, 2002), and labels using *is-a* patterns (Hearst, 1992). By filtering the class labels using distributional similarity, a large number of high-precision labeled clusters are extracted. The algorithm proceeds iteratively: at each step, all clusters are tested for label *coherence* and all coherent labels are tested for high cluster *specificity*. Label L is coherent if it is shared by at least $J\%$ of the instances in cluster C, and it is specific if the total number of other clusters $C' \in \mathcal{C}, C' \neq C$ containing instances with label L is less than K. When a cluster is found to match these criteria, it is removed from $\mathcal{C}$ and added to an output set. The procedure terminates when no new clusters can be removed from $\mathcal{C}$. Table 1 shows a few randomly chosen classes and representative instances obtained by this procedure.

2.2 Extraction from Structured Text

To expand the instance sets extracted from free text, we use a *table-based extraction* method that mines structured Web data in the form of HTML tables. A significant fraction of the HTML tables in Web pages is assumed to contain coherent lists of instances suitable for extraction. Identifying such tables from scratch is hard, but seed instance lists can be used to identify potentially coherent table columns. In this paper we use the WebTables database of around 154 million tables as our structured data source (Cafarella et al., 2008).

We employ a simple ranking scheme for candidate instances in the WebTables corpus $\mathcal{T}$. Each table $\mathbf{T} \in \mathcal{T}$ consists of one or more columns. Each column $g \in \mathbf{T}$ consists of a set of candidate instances $i \in g$ corresponding to row elements. We define the set of unique *seed matches* in g relative to semantic class $C \in \mathcal{C}$ as

$$M_C(g) \overset{\text{def}}{=} \{i \in I(C) : i \in g\}$$

where $I(C)$ denotes the set of instances in seed class C. For each column g, we define its α*-unique class coverage*, that is, the set of classes that have at least α unique seeds in g,

$$Q(g; \alpha) \overset{\text{def}}{=} \{C \in \mathcal{C} : |M_C(g)| \geq \alpha\}.$$

Using M and Q we define a method for scoring columns relative to each class. Intuitively, such a score should take into account not only the number of matches from class C, but also the total number of classes that contribute to Q and their relative overlap. Towards this end, we introduce the scoring function

$$score(C, g; \alpha) \overset{\text{def}}{=} \underbrace{|M_C(g)|}_{\text{seed matches}} \cdot \overbrace{\frac{|M_C(g)|}{|\bigcup_{C' \in Q(g;\alpha)} I(C')|}}^{\text{class coherence}}$$

which is the simplest scoring function combining the number of seed matches with the coherence of the table column. Coherence is a critical notion in WebTables extraction, as some tables contain instances across many diverse seed classes, contributing to extraction noise. The class coherence introduced here also takes into account class overlap; that

Class	Size	Examples of Instances
Book Publishers	70	crown publishing, kluwer academic, prentice hall, puffin
Federal Agencies	161	catsa, dhs, dod, ex-im bank, fsis, iema, mema, nipc, nmfs, tdh, usdot
Mammals	956	armadillo, elephant shrews, long-tailed weasel, river otter, weddell seals, wild goat
NFL Players	180	aikman, deion sanders, fred taylor, jamal lewis, raghib ismail, troy vincent
Scientific Journals	265	biometrika, european economic review, nature genetics, neuroscience
Social Issues	210	gender inequality, lack of education, substandard housing, welfare dependency
Writers	5089	bronte sisters, hemingway, kipling, proust, torquato tasso, ungaretti, yeats

Table 1: A sample of the open-domain classes and associated instances from (Van Durme and Paşca, 2008).

is, a column containing many semantically similar classes is penalized less than one containing diverse classes.[1] Finally, an extracted instance i is assigned a score relative to class C equal to the sum of all its column scores,

$$score(i, C; \alpha) \overset{\text{def}}{=} \frac{1}{Z_C} \sum_{g \in \mathbf{T}, \mathbf{T} \in \mathcal{T}} score(C, g; \alpha)$$

where Z_C is a normalizing constant set to the maximum score of any instance in class C. This scoring function assigns high rank to instances that occur frequently in columns with many seed matches and high class specificity.

The ranked list of extracted instances is post-filtered by removing all instances that occur in less than d unique Internet domains.

3 Graph-Based Extraction

To combine the extractions from both free and structured text, we need a representation capable of encoding efficiently all the available information. We chose a graph representation for the following reasons:

- Graphs can represent complicated relationships between classes and instances. For example, an ambiguous instance such as *Michael Jordan* could belong to the class of both *Professors* and *NBA players*. Similarly, an instance may belong to multiple nodes in the hierarchy of classes. For example, *Blue Whales* could belong to both classes *Vertebrates* and *Mammals*, because *Mammals* are a subset of *Vertebrates*.

- Extractions from multiple sources, such as Web queries, Web tables, and text patterns can be represented in a single graph.

- Graphs make explicit the potential paths of information propagation that are implicit in the more common local heuristics used for weakly-supervised information extraction. For example, if we know that the instance *Bill Clinton* belongs to both classes *President* and *Politician* then this should be treated as evidence that the class of *President* and *Politician* are related.

Each instance-class pair (i, C) extracted in the first phase (Section 2) is represented as a weighted edge in a graph $G = (V, E, W)$, where V is the set of nodes, E is the set of edges and $W : E \to \mathbb{R}^+$ is the weight function which assigns positive weight to each edge. In particular, for each (i, C, w) triple from the set of base extractions, i and C are added to V and (i, C) is added to E,[2] with $W(i, C) = w$. The weight w represents the total score of all extractions with that instance and class. Figure 1 illustrates a portion of a sample graph. This simple graph representation could be refined with additional types of nodes and edges, as we discuss in Section 7.

In what follows, all nodes are treated in the same way, regardless of whether they represent instances or classes. In particular, all nodes can be assigned class labels. For an instance node, that means that the instance is hypothesized to belong to the class; for a class node, that means that the node's class is hypothesized to be semantically similar to the label's class (Section 5).

We now formulate the task of assigning labels to nodes as graph label propagation. We are given a

[1]Note that this scoring function does not take into account class containment: if all seeds are both *wind Instruments* and *instruments*, then the column should assign higher score to the more specific class.

[2]In practice, we use two directed edges, from i to C and from C to i, both with weight w.

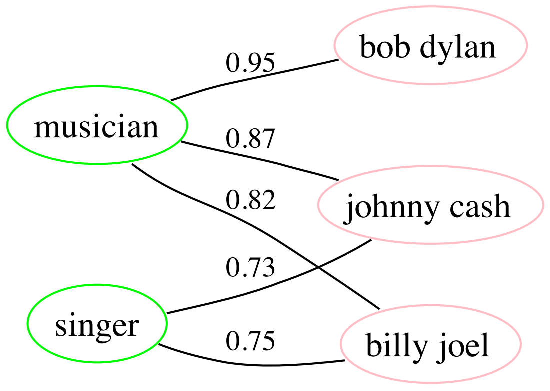


Figure 1: Section of a graph used as input into Adsorption. Though the nodes do not have any type associated with them, for readability, instance nodes are marked in pink while class nodes are shown in green.

set of instances $\mathcal{I}$ and a set of classes $\mathcal{C}$ represented as nodes in the graph, with connecting edges as described above. We annotate a few instance nodes with labels drawn from $\mathcal{C}$. That is, classes are used both as nodes in the graph and as labels for nodes. There is no necessary alignment between a class node and any of the (class) labels, as the final labels will be assigned by the Adsorption algorithm.

The Adsorption label propagation algorithm (Baluja et al., 2008) is now applied to the given graph. Adsorption is a general framework for label propagation, consisting of a few nodes annotated with labels and a rich graph structure containing the universe of all labeled and unlabeled nodes. Adsorption proceeds to label all nodes based on the graph structure, ultimately producing a probability distribution over labels for each node.

More specifically, Adsorption works on a graph $G = (V, E, W)$ and computes for each node v a *label distribution* L_v that represents which labels are more or less appropriate for that node. Several interpretations of Adsorption-type algorithms have appeared in various fields (Azran, 2007; Zhu et al., 2003; Szummer and Jaakkola, 2002; Indyk and Matousek, 2004). For details, the reader is referred to (Baluja et al., 2008). We use two interpretations here:

Adsorption through Random Walks: Let $G_r = (V, E_r, W_r)$ be the edge-reversed version of the original graph $G = (V, E, W)$ where $(a, b) \in E_r$ iff $(b, a) \in E$; and $W_r(a, b) = W(b, a)$. Now, choose a node of interest $q \in V$. To estimate L_q for q, we perform a random walk on G_r starting from q to generate values for a random label variable L. After reaching a node v during the walk, we have three choices:

1. With probability p_v^{cont}, continue the random walk to a neighbor of v.
2. With probability p_v^{abnd}, abandon the random walk. This abandonment probability makes the random walk stay relatively close to its source when the graph has high-degree nodes. When the random walk passes through such a node, it is likely that further transitions will be into regions of the graph unrelated to the source. The abandonment probability mitigates that effect.
3. With probability p_v^{inj}, stop the random walk and emit a label L from I_v.

L_q is set to the expectation of all labels L emitted from random walks initiated from node q.

Adsorption through Averaging: For this interpretation we make some changes to the original graph structure and label set. We extend the label distributions L_v to assign a probability not only to each label in $\mathcal{C}$ but also to the *dummy* label $\perp$, which represents lack of information about the actual label(s). We represent the initial knowledge we have about some node labels in an *augmented* graph $G' = (V', E', W')$ as follows. For each $v \in V$, we define an *initial* distribution $I_v = L^{\perp}$, where $L^{\perp}$ is the *dummy* distribution with $L^{\perp}(\perp) = 1$, representing lack of label information for v. In addition, let $V_s \subseteq V$ be the set of nodes for which we have some actual label knowledge, and let $V' = V \cup \{\bar{v} : v \in V_s\}, E' = E \cup \{(\bar{v}, v) : v \in V_s\}$, and $W'(\bar{v}, v) = 1$ for $v \in V_s$, $W'(u, v) = W(u, v)$ for $u, v \in V$. Finally, let $I_{\bar{v}}$ (seed labels) specify the knowledge about possible labels for $v \in V_s$. Less formally, the $\bar{v}$ nodes in G' serve to *inject* into the graph the prior label distributions for each $v \in V_s$.

The algorithm proceeds as follows: For each node use a fixed-point computation to find label distributions that are weighted averages of the label distributions for all their neighbors. This causes the non-dummy initial distribution of V_s nodes to be propagated across the graph.

Baluja et al. (2008) show that those two views are equivalent. Algorithm 1 combines the two views: instead of a random walk, for each node v, it iteratively computes the weighted average of label distributions from neighboring nodes, and then uses the random walk probabilities to estimate a new label distribution for v.

For the experiments reported in Section 4, we used the following heuristics from Baluja et al. (2008) to set the random walk probabilities:

- Let $c_v = \frac{\log \beta}{\log(\beta + \exp H(v))}$ where $H(v) = -\sum_u p_{uv} \times \log(p_{uv})$ with $p_{uv} = \frac{W(u,v)}{\sum_{u'} W(u',v)}$. $H(v)$ can be interpreted as the entropy of v's neighborhood. Thus, c_v is lower if v has many neighbors. We set $\beta = 2$.
- $j_v = (1 - c_v) \times \sqrt{H(v)}$ if $I_v \neq L^\top$ and 0 otherwise.
- Then let

$$\begin{aligned} z_v &= \max(c_v + j_v, 1) \\ p_v^{cont} &= c_v / z_v \\ p_v^{inj} &= j_v / z_v \\ p_v^{abnd} &= 1 - p_v^{cont} - p_v^{abnd} \end{aligned}$$

  Thus, abandonment occurs only when the continuation and injection probabilities are low enough.

The algorithm is run until convergence which is achieved when the label distribution on each node ceases to change within some tolerance value. Alternatively, the algorithm can be run for a fixed number of iterations which is what we used in practice[3].

Finally, since Adsorption is memoryless, it easily scales to tens of millions of nodes with dense edges and can be easily parallelized, as described by Baluja et al. (2008).

[3]The number of iterations was set to 10 in the experiments reported in this paper.

Algorithm 1 Adsorption Algorithm.
Input: $G' = (V', E', W')$, I_v $(\forall v \in V')$.
Output: Distributions $\{L_v : v \in V\}$.

```
1: $L_v = I_v \; \forall v \in V'$
2:
3: **repeat**
4:     $N_v = \sum_u W(u,v)$
5:     $D_v = \frac{1}{N_v} \sum_u W(u,v) L_u \;\; \forall v \in V'$
6:     **for all** $v \in V'$ **do**
7:         $L_v = p_v^{cont} \times D_v + p_v^{inj} \times I_v + p_v^{abnd} \times L^\top$
8:     **end for**
9: **until** convergence
```

4 Experiments

4.1 Data

As mentioned in Section 3, one of the benefits of using Adsorption is that we can combine extractions by different methods from diverse sources into a single framework. To demonstrate this capability, we combine extractions from free-text patterns and from Web tables. To the best of our knowledge, this is one of the first attempts in the area of minimally-supervised extraction algorithms where unstructured and structured text are used in a principled way within a single system.

Open-domain (instance, class) pairs were extracted by applying the method described by Van Durme and Paşca (2008) on a corpus of over 100M English web documents. A total of 924K (instance, class) pairs were extracted, containing 263K unique instances in 9081 classes. We refer to this dataset as A8.

Using A8, an additional 74M unique (instance,class) pairs are extracted from a random 10% of the WebTables data, using the method outlined in Section 2.2. For maximum coverage we set $\alpha = 2$ and $d = 2$, resulting in a large, but somewhat noisy collection. We refer to this data set as WT.

4.2 Graph Creation

We applied the graph construction scheme described in Section 3 on the A8 and WT data combined, resulting in a graph with 1.4M nodes and 75M edges. Since extractions in A8 are not scored, weight of all

Seed Class	Seed Instances
Book Publishers	millbrook press, academic press, springer verlag, chronicle books, shambhala publications
Federal Agencies	dod, nsf, office of war information, tsa, fema
Mammals	african wild dog, hyaena, hippopotamus, sperm whale, tiger
NFL Players	ike hilliard, isaac bruce, torry holt, jon kitna, jamal lewis
Scientific Journals	american journal of roentgenology, pnas, journal of bacteriology, american economic review, ibm systems journal

Table 2: Classes and seeds used to initialize Adsorption.

edges originating from A8 were set at 1[4]. This graph is used in all subsequent experiments.

5 Evaluation

We evaluated the Adsorption algorithm under two experimental settings. First, we evaluate Adsorption's extraction precision on (instance, class) pairs obtained by Adsorption but not present in A8 (Section 5.1). This measures whether Adsorption can add to the A8 extractions at fairly high precision. Second, we measured Adsorption's ability to assign labels to a fixed set of gold instances drawn from various classes (Section 5.2).

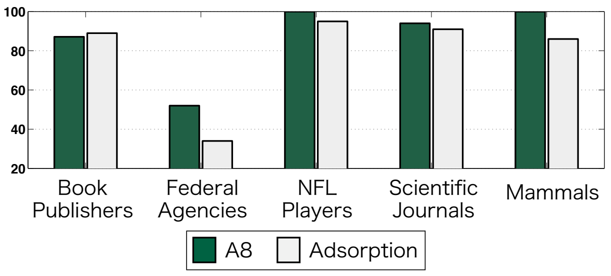


Figure 2: Precision at 100 comparisons for A8 and Adsorption.

5.1 Instance Precision

First we manually evaluated precision across five randomly selected classes from A8: Book Publishers, Federal Agencies, NFL Players, Scientific Journals and Mammals. For each class, 5 seed instances were chosen manually to initialize Adsorption. These classes and seeds are shown in Table 2. Adsorption was run for each class separately and the resulting ranked extractions were manually evaluated.

Since the A8 system does not produce ranked lists of instances, we chose 100 random instances from the A8 results to compare to the top 100 instances produced by Adsorption. Each of the resulting 500 instance-class pairs (i, C) was presented to two human evaluators, who were asked to evaluate whether the relation "i is a C" was correct or incorrect. The user was also presented with Web search link to verify the results against actual documents. Results from these experiments are presented in Figure 2 and Table 4. The results in Figure 2 show that the A8 system has higher precision than the Adsorption system. This is not surprising since the A8 system is tuned for high precision. When considering individual evaluation classes, changes in precision scores between the A8 system and the Adsorption system vary from a small increase from 87% to 89% for the class Book Publishers, to a significant decrease from 52% to 34% for the class Federal Agencies, with a decrease of 10% as an average over the 5 evaluation classes.

Class	Precision at 100 (non-A8 extractions)
Book Publishers	87.36
Federal Agencies	29.89
NFL Players	94.95
Scientific Journals	90.82
Mammal Species	84.27

Table 4: Precision of top 100 Adsorption extractions (for five classes) which were **not** present in A8.

Table 4 shows the precision of the Adsorption system for instances not extracted by the A8 system.

[4]A8 extractions are assumed to be high-precision and hence we assign them the highest possible weight.

Seed Class	Non-Seed Class Labels Discovered by Adsorption
Book Publishers	small presses, journal publishers, educational publishers, academic publishers, commercial publishers
Federal Agencies	public agencies, governmental agencies, modulation schemes, private sources, technical societies
NFL Players	sports figures, football greats, football players, backs, quarterbacks
Scientific Journals	prestigious journals, peer-reviewed journals, refereed journals, scholarly journals, academic journals
Mammal Species	marine mammal species, whale species, larger mammals, common animals, sea mammals

Table 3: Top class labels ranked by their similarity to a given seed class in Adsorption.

Seed Class	Sample of Top Ranked Instances Discovered by Adsorption
Book Publishers	small night shade books, house of anansi press, highwater books, distributed art publishers, copper canyon press
NFL Players	tony gonzales, thabiti davis, taylor stubblefield, ron dixon, rodney hannah
Scientific Journals	journal of physics, nature structural and molecular biology, sciences sociales et santé, kidney and blood pressure research, american journal of physiology–cell physiology

Table 5: Random examples of top ranked extractions (for three classes) found by Adsorption which were not present in A8.

Such an evaluation is important as one of the main motivations of the current work is to increase coverage (recall) of existing high-precision extractors without significantly affecting precision. Results in Table 4 show that Adsorption is indeed able to extraction with high precision (in 4 out of 5 cases) new instance-class pairs which were not extracted by the original high-precision extraction set (in this case A8). Examples of a few such pairs are shown in Table 5. This is promising as almost all state-of-the-art extraction methods are high-precision and low-recall. The proposed method shows a way to overcome that limitation.

As noted in Section 3, Adsorption ignores node type and hence the final ranked extraction may also contain classes along with instances. Thus, in addition to finding new instances for classes, it also finds additional class labels similar to the seed class labels with which Adsorption was run, at no extra cost. Some of the top ranked class labels extracted by Adsorption for the corresponding seed class labels are shown in Table 3. To the best of our knowledge, there are no other systems which perform both tasks simultaneously.

5.2 Class Label Recall

Next we evaluated each extraction method on its relative ability to assign labels to class instances. For each test instance, the five most probably class labels are collected using each method and the Mean Reciprocal Rank (MRR) is computed relative to a gold standard target set. This target set, WN-gold, consists of the 38 classes in Wordnet containing 100 or more instances.

In order to extract meaningful output from Adsorption, it is provided with a number of labeled seed instances (1, 5, 10 or 25) from each of the 38 test classes. Regardless of the actual number of seeds used as input, all 25 seed instances from each class are removed from the output set from all methods, in order to ensure fair comparison.

The results from this evaluation are summarized in Table 6; AD x refers to the adsorption run with x seed instances. Overall, Adsorption exhibits higher MRR than either of the baseline methods, with MRR increasing as the amount of supervision is increased. Due to its high coverage, WT assigns labels to a larger number of the instance in WN-gold than any other method. However, the average rank of the correct class assignment is lower, resulting is

Method	MRR (full)	MRR (found only)	# found
A8	0.16	0.47	2718
WT	0.15	0.21	**5747**
AD 1	0.26	0.45	4687
AD 5	0.29	0.48	4687
AD 10	0.30	0.51	4687
AD 25	**0.32**	**0.55**	4687

Table 6: Mean-Reciprocal Rank scores of instance class labels over 38 Wordnet classes (WN-gold). MRR (full) refers to evaluation across the entire gold instance set. MRR (found only) computes MRR only on recalled instances.

lower MRR scores compared to Adsorption. This result highlights Adsorption's ability to effectively combine high-precision, low-recall (A8) extractions with low-precision, high-recall extractions (WT) in a manner that improves *both* precision and coverage.

6 Related Work

Graph based algorithms for minimally supervised information extraction methods have recently been proposed. For example, Wang and Cohen (2007) use a random walk on a graph built from entities and relations extracted from semi-structured text. Our work differs both conceptually, in terms of its focus on open-domain extraction, as well as methodologically, as we incorporate both unstructured and structured text. The re-ranking algorithm of Bellare et al. (2007) also constructs a graph whose nodes are instances and attributes, as opposed to instances and classes here. Adsorption can be seen as a generalization of the method proposed in that paper.

7 Conclusion

The field of open-domain information extraction has been driven by the growth of Web-accessible data. We have staggering amounts of data from various structured and unstructured sources such as general Web text, online encyclopedias, query logs, web tables, or link anchor texts. Any proposed algorithm to extract information needs to harness several data sources and do it in a robust and scalable manner. Our work in this paper represents a first step towards that goal. In doing so, we achieved the following:

1. Improved coverage relative to a high accuracy instance-class extraction system while maintaining adequate precision.

2. Combined information from two different sources: free text and web tables.

3. Demonstrated a graph-based label propagation algorithm that given as little as five seeds per class achieved good results on a graph with more than a million nodes and 70 million edges.

In this paper, we started off with a simple graph. For future work, we plan to proceed along the following lines:

1. Encode richer relationships between nodes, for example instance-instance associations and other types of nodes.

2. Combine information from more data sources to answer the question of whether more data or diverse sources are more effective in increasing precision and coverage.

3. Apply similar ideas to other information extraction tasks such as relation extraction.

Acknowledgments

We would like to thank D. Sivakumar for useful discussions and the anonymous reviewers for helpful comments.

References

A. Azran. 2007. The rendezvous algorithm: multiclass semi-supervised learning with markov random walks. *Proceedings of the 24th international conference on Machine learning*, pages 49–56.

S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph.

K. Bellare, P. Talukdar, G. Kumaran, F. Pereira, M. Liberman, A. McCallum, and M. Dredze. 2007. Lightly-Supervised Attribute Extraction. *NIPS 2007 Workshop on Machine Learning for Web Search.*

M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. 2008. Webtables: Exploring the power of tables on the web. *VLDB.*

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539–545, Nantes, France.

P. Indyk and J. Matousek. 2004. Low-distortion embeddings of finite metric spaces. *Handbook of Discrete and Computational Geometry*.

B. Jansen, A. Spink, and T. Saracevic. 2000. Real life, real users, and real needs: a study and analysis of user queries on the Web. *Information Processing and Management*, 36(2):207–227.

D. Lin and P. Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics (COLING-02)*, pages 1–7.

K. McCarthy and W. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1050–1055, Montreal, Quebec.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479, Orlando, Florida.

M. Stevenson and R. Gaizauskas. 2000. Using corpus-derived name lists for named entity recognition. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-00)*, Seattle, Washington.

M. Szummer and T. Jaakkola. 2002. Partially labeled classification with markov random walks. *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 NIPS Conference*.

B. Van Durme and M. Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. *Twenty-Third AAAI Conference on Artificial Intelligence*.

R. Wang and W. Cohen. 2007. Language-Independent Set Expansion of Named Entities Using the Web. *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 342–350.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. *ICML-03, 20th International Conference on Machine Learning*.

SCAD: Collective Discovery of Attribute Values

Anton Bakalov[*]
U. Mass., Amherst
abakalov@cs.umass.edu

Ariel Fuxman
Microsoft Research
arielf@microsoft.com

Partha Pratim Talukdar
Microsoft Research[†]

Soumen Chakrabarti
IIT Bombay
soumen@cse.iitb.ac.in

ABSTRACT

Search engines today offer a rich user experience, no longer restricted to "ten blue links". For example, the query "Canon EOS Digital Camera" returns a photo of the digital camera, and a list of suitable merchants and prices. Similar results are offered in other domains like food, entertainment, travel, etc. All these experiences are fueled by the availability of structured data about the entities of interest.

To obtain this structured data, it is necessary to solve the following problem: given a category of entities with its schema, and a set of Web pages that mention and describe entities belonging to the category, build a structured representation for the entity under the given schema. Specifically, collect structured numerical or discrete attributes of the entities.

Most previous approaches regarded this as an information extraction problem on individual documents, and made no special use of numerical attributes. In contrast, we present an end-to-end framework which leverages signals not only from the Web page context, but also from a collective analysis of all the pages corresponding to an entity, and from constraints related to the actual values within the domain.

Our current implementation uses a general and flexible Integer Linear Program (ILP) to integrate all these signals into holistic decisions over all attributes. There is one ILP per entity and it is small enough to be solved in under 38 milliseconds in our experiments.

We apply the new framework to a setting of significant practical importance: catalog expansion for Commerce search engines, using data from Bing Shopping. Finally, we present experiments that validate the effectiveness of the framework and its superiority to local extraction.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms, Experimentation

[*]Work done while at Microsoft Research.
[†]On contract.

Keywords

Weak Supervision, Collective Information Extraction, Integer Linear Program, Commerce Search, Attribute Discovery

1. INTRODUCTION

Search engines today offer a rich user experience far beyond "ten blue links". For example, a query such as "canon rebel eos" produces a result that includes a photo of the digital camera, and a list of suitable merchants and prices. Similar results are offered for queries in domains such as entertainment, travel, etc. All these experiences are fueled by the availability of structured data about the entities of interest (restaurants, consumer products, etc.) In some cases, the structured data is obtained via data feeds from specialized providers, and constructed with extensive manual input. In other cases, wrappers are used to extract information from Web pages.

However, these data acquisition models have inherent limitations. Manual techniques cannot scale to the large number of entities that need to be supported, and cannot keep pace with the constant emergence of new entities (e.g., new products released to the market). Wrapper-based techniques [19] are sometimes an effective solution; but they are site-specific, and applicable only to sites whose structure and format is quite predictable. As a result, they tend to be brittle and require continual maintenance [7]. These limitations suggest the need for systems that satisfy the following requirements. First, structured data acquisition should be done in a fully automated way. Second, the techniques must be site-independent and should not make strong assumptions about Web page formatting or structure.

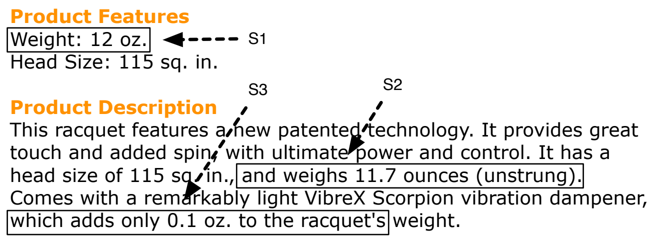
In this paper, we present SCAD[1], a system designed to address these requirements. SCAD learns to acquire structured data for entities of a given category (e.g., digital cameras). For each category, it is given a) a schema (including a set of attribute names), b) a small set of seed entities together with their values for the attributes in the schema, and c) a set of related Web pages (e.g., merchant offer pages for a product). SCAD then learns to automatically extract the values of attributes in the schema for other entities in the category. SCAD imposes very little cognitive burden on the trainer: in our experiments, we show that it suffices to provide seed sets with as few as 20 entities per category.

Our work is related to previous efforts in named entity

[1]SCAD stands for "Structured Collective Attribute Discovery". Discovery from scads of unstructured pages.

recognition (NER) [19]. However, with rare exceptions that we discuss later, NER depends exclusively on modeling the local left and right contexts of mentions of entities in unstructured text. SCAD goes beyond the local context modeling of NER systems, and incorporates various *global* signals into the extraction process. The first signal is that we find entity-level consensus among the candidate attribute values across multiple pages. E.g., even if we have an extraction error from one Web page, we may be able to recover by leveraging extractions for the same attribute from other Web pages. Second, we exploit category-level (i.e., across multiple entities) value distributions to reduce spurious extractions. Third, SCAD can effortlessly incorporate a wide variety of constraints from world knowledge, e.g., a value on a Web page is unlikely to refer to more than one attribute in the schema, or the disk capacity of a laptop is larger than RAM size.

Our current implementation uses a general and flexible integer program to integrate these global signals with local context information from the Web pages. Although it is a combinatorial optimization at an entity level, the number of constraints is modest and optimization takes only dozens of milliseconds per entity.


Product Features

Weight: 12 oz.

Head Size: 115 sq. in.

Product Description

This racquet features a new patented technology. It provides great touch and added spin with ultimate power and control. It has a head size of 115 sq. in., and weighs 11.7 ounces (unstrung). Comes with a remarkably light VibreX Scorpion vibration dampener, which adds only 0.1 oz. to the racquet's weight.

Figure 1: Sample snippets describing the tennis racquet *Wilson BLX Khamsin-Five*.

To illustrate the worth of these signals, consider the following example. Suppose that our goal is to produce structured data for consumer products. (This is a fundamental task in consumer product search engines like Yahoo! Product Search and Bing Shopping.) In this context, an entity is a product, such as the *Wilson BLX Khamsin-Five*, which belongs to the product category *Tennis racquets*. Salient attributes for this category would be “Brand”, “Manufacturer”, “Head Size”, “Strung Weight”, “Unstrung Weight”, etc. Assume that we are given a set of Web pages that correspond to this racquet. (We will explain later how Commerce search engines typically gather this information.) In Figure 1, we show an excerpt of such a page.

Suppose that the schema contains (among others) the following attributes: “Strung Weight” and “Unstrung Weight”; and that we would like to obtain their values for the racquet *Wilson BLX Khamsin-Five*. SCAD would start by spotting that “12 oz.” and “11.7 oz.” are values that might be associated to an attribute (for example, because the schema contains numeric attributes). It would then construct snippets of text centered around the values. The snippets for the values “12 oz.” (snippet s_1) and “11.7 oz.” (snippet s_2) are highlighted with rectangles in Figure 1. A local extractor will produce, for a given snippet, a probability distribution ϕ over the attributes in the input schema. In our example, the local extractor will be highly confident that snippet s_2 is about unstrung weight ($\phi(s_2,\text{“unstrung weight”}) = 1$, say), but it will be less certain about the attribute corresponding to s_1 (both “strung weight” and “unstrung weight” will have comparable probabilities). Since the local model is certain that snippet s_2 should be associated to “unstrung weight”, it is reasonable to assume that its value (11.7 oz.) is the right value for this attribute. But the local model is not conclusive about the strung weight of the racquet.

Now, let us show how the entity and category-level constraints help to reduce the uncertainty of the local model. Consider two of the entity-level constraints of our framework: (C1) each snippet should be assigned exactly one attribute; (C2) each attribute should be assigned at most one value. Since s_2 has been assigned “unstrung weight”, due to constraint C1, it cannot be assigned any other attribute. Furthermore, due to constraint C2, “unstrung weight” cannot be assigned the value “12oz”. (Otherwise, the same attribute would have multiple values.) Thus, s_1 cannot be assigned to the attribute “unstrung weight”. Since the only other plausible candidate for s_1 is “strung weight” and its value is 12 oz., we can now conclude that the value of “strung weight” should be 12 oz.

Notice that there is some chance that 12 vs 11.7 oz. is just precision variation, and both are unstrung weights. This is a valid configuration considered by SCAD, but in this case it is ruled out due to the fact that there is signal from the local model ϕ about the fact that 11.7 oz. is associated to “unstrung weight”. Another example of an entity-level constraint that can help us decide how to assign attributes to values is that “strung weight” should be greater than “unstrung weight.” Thus, assigning “11.7 oz.” and “12 oz.” to “strung weight” and “unstrung weight”, respectively, is not allowed. In the paper, we refer to these constraints as inter-attribute constraints.

To understand how the category-level constraints work, consider the snippet s_3 shown in Figure 1 – “which adds only 0.1 oz. to the racquet's”. From the textual content of s_3, the local model might assign it some probability of being associated to “strung weight”. However, from an understanding of the value domain for this attribute, it is obvious that the quantity 0.1 oz. cannot be associated to the weight of a tennis racquet.

The rest of the paper is organized as follows. In Section 2, we present related work; and in Section 3 we give an overview of our framework. In Section 4, we provide the details of the global assignment optimization problem solved by SCAD, and in Section 5 we present SCAD's local model. The experimental results are presented in Section 6.

2. RELATED WORK

Creating a structured entity description involves locating a mention of the entity, mentions of structured attributes, and evidence of suitable associations between the entity and attributes. Detecting mentions of entities is a well-developed field [19], as is detecting mentions of relations (not necessarily attribute relations) between entities [3]. However, there is relatively little work on attribute extraction that is collective across attribute mentions in a document and that aggregates evidence over multiple documents. (The quantity consensus query system [1] does only the latter.)

Several methods for incorporating non-local information for Information Extraction have been previously proposed [2, 10, 20]. These methods try to enforce label consistency where the same token mentioned multiple times in a document is assumed to have the same label. Unfortunately, this assumption is not valid in our application setting. For example, in a document from the Washing Machines category with multiple mentions of the token *3*, one mention may correspond to the "number of cycles" attribute, while another to the "number of doors".

As noted earlier, wrapper induction techniques [19] require sites (or documents) to be template-based [11], and they are expensive to maintain [7]. SCAD does not induce wrappers and it does not require documents to be template-based, and hence it is applicable more generally.

Whereas SCAD seeks to collectively extract attributes of entities, a large body of work, including KnowItAll [9] discover instances of types starting with a few seed instances. These are very different problems, and the building blocks from entity discovery (e.g., PMI [21]) are not necessarily applicable to our scenario. The work that is closest to ours is Kylin [23], since it also focuses on building structured representations (in their case, for Wikipedia infoboxes). Kylin extracts each attribute value in isolation, compared to the collective extraction in SCAD. One of the critical differences between SCAD and Kylin is that SCAD aggregates attribute evidence across multiple documents, while Kylin does not. As demonstrated through experiments in Section 6, we find such aggregation to be very effective in practice.

A semi-supervised method for extraction of attribute-value pairs from product descriptions is presented in [17]. In contrast to that work, SCAD not only assigns attributes to values in context, but it also constructs an entity-specific record consisting of attribute-value pairs, with one (or more) value(s) per attribute. We note that this is a harder task as it involves an additional step of resolving value ambiguity for a given attribute. While attribute extraction itself (without value) is the goal of [16], SCAD focuses on the extraction of *values* of attributes. Field compabilities learned by [22] can be used as Entity-level constraints in SCAD (Section 4.3.1). Unlike [22], we do not assume the availability of quantity-attribute mapping as input, and instead estimate it from the data by using a local model, as described in Section 4.4.

Some earlier work [1, 8, 15, 24] focused on quantity queries. Some of these exploited associative reinforcement between similar quantities, but not simultaneously the dissociative nature of quantities within a discourse/page (i.e., that two quantities within a discourse tend to pertain to *different* attributes).

As a global assignment problem between attributes and values, it is natural to express our problem using integer programs, and this has been done in the past, e.g., in the CCM framework [4] for NLP tasks. However, existing instantiations of the CCM framework [4, 18] appear not to handle all the constraints we would like to impose, including consensus and constraints involving quantitative attributes. For example, we cannot see how to express in CCM that assigning 0.3 pounds to battery weight forbids us from assigning 0.2 pounds to gross weight. Therefore, we use additional special variables in SCAD's integer program (see Section 4.2).

Like SCAD, most collective information extraction algorithms must reconcile local and global evidence, and aggregate them. EntityRank [5] extracts records with structured fields (e.g., email and phone number of a researcher) from multiple pages, and ranks tuples using a majority vote from source pages weighted by their PageRank. However, EntityRank does not exploit constraints involving value distributions or involving multiple attributes.

3. FRAMEWORK OVERVIEW

Attribute inference can be modeled as filling in an incomplete table. Each row corresponds to an **entity**, and the whole table is expected to pertain to a coherent **category** of entities. While the framework allows any kind of entities, in this paper we will usually draw examples from Commerce search, so entities will often be consumer products. Tennis racquets and laptop computers are examples of categories. The Wilson BLX Khamsin-Five and the Lenovo X300 Thinkpad are entities belonging to those respective categories. Entities have **attributes**, such as string tension, strung weight, and battery life. These attributes are represented as columns in the table for the category.

We will regard the cell values of the tables as structured data, either categorical or quantitative attributes of the entity represented by the row. While our framework can support both categorical and numeric attributes, our experiments will often focus on the latter. For simplicity we will also assume that there are no multivalued attributes.

3.1 Training phase overview

In the beginning, a few rows in the table have all columns filled out. In addition, each of these rows has associated with it a set of **contexts**. A context is an extent of text that provides evidence of the attribute values for the entity. For example, in the domain of consumer products, it is often easy to find Web pages that are dedicated to a product and come from merchants, manufacturers, reviewers, users, etc. (For sources that are grossly heterogeneous or have a lot of site boilerplate, we will assume that suitable page segmentation and key content panel extraction techniques exist.)

As Figure 2 shows, we collected contexts (where entities are mentioned together with attributes) from two sources: Web pages and offer pages. We used suitable keyword queries and the Bing Web search API[2] to collect Web pages, and offer pages were collected from online merchants. (More details on the collection of both kinds of pages in Section 6). The pages are first passed through a snippet extractor that locates text around attributes of desired types.

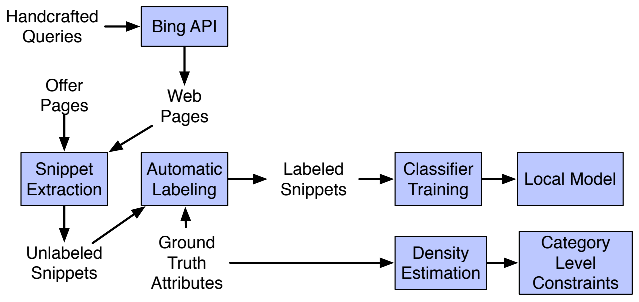


Figure 2: Training phase.

Our framework uses a *local model* of compatibility between a snippet and an attribute. The local model has to be trained using ground truth to associate snippets with attributes. The ground truth consists of structured, correct

[2]http://www.bing.com/developers

values for attributes. We look for these values in the collected snippets (see Section 5.2) to automatically generate training data. Similar mechanisms have been used in Kylin [23], and other systems. One novelty of our framework is that we also use the "gold data" to build density estimates (Section 4.3.3) of attributes, which is then used in our *global* optimization.

3.2 Deployment phase overview

Continuing with the incomplete table metaphor, the remaining rows have available unstructured contexts, but all of their cells are empty. Our task during the deployment phase, sketched in Figure 3, is to collectively mine the contexts in order to fill in the empty cells.

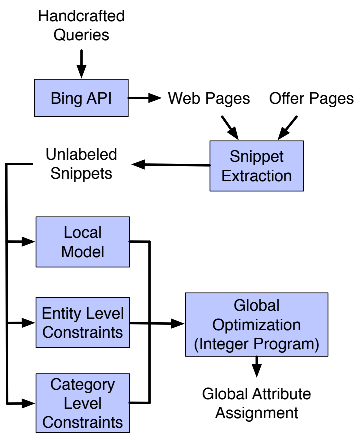


Figure 3: Deployement phase.

Context Web pages are obtained and snippets extracted as before. However, the local model now provides only one form of input to a global optimizer. Other inputs come from entity-level assignment constraints, as well as the densities estimated during the training phase.

In our current implementation of the framework, all these signals and constraints are input into a flexible and general integer linear program (ILP), which is then solved using the Microsoft Solver Foundation [3].

4. GLOBAL ASSIGNMENT OPTIMIZATION

In this section, we explain in detail the global optimization problem solved by SCAD. We first present terminology and notation, and then discuss the different constraints used for collective extraction. Finally, we introduce the objective function, and illustrate it with an example.

4.1 Terminology and notation

A **context** c is an extent of text that pertains to an entity and provides evidence for its attribute values. A context may span a page or a region of a page. A **snippet** $c.s$ is a segment of text in a context c which contains a **value** $c.s.v$ that is a candidate to be associated with an attribute of the entity. In general, a context can contain multiple snippets.

For example, the Web page region shown in Figure 1 is a context. In that Figure, we can see snippets such as "Weight: 12 oz." and "and weighs 11.7 ounces (unstrung)". The value for the former snippet is 12 ounces, while the value for the latter is 11.7 ounces. (Note that these are no longer considered as strings, but are converted into numbers with associated units; in particular, units can be converted.) We use the following rules to create snippets. Once we detect an attribute value, we consider the six words to the left and right of it. If there is a number, or some distinctive HTML tags (such as `<tr>, <li>, <title>`), then we include all the words up until that token. If we come across a mention of an attribute name that we know about, then we include it in the snippet and disregard everything beyond it.

[3]`http://code.msdn.microsoft.com/solverfoundation`

Each category is associated to a table. The table has a **schema**, consisting of a set of typed attributes. We will denote an attribute with a. The **types** of the attributes are domain-specific, and have **units** associated with them. For example, "strung weight" and "head size" are attributes of the Tennis Racquets category, whose types are "weight" and "area", respectively. Example of a units for weight and area are "lbs." and "square inches", respectively. Strictly speaking, an attribute is just an unambiguous column ID in the table, but it is described by one (or more, synonymous) names.

Let e be an entity and $a_1, \ldots, a_n$ be the attributes of its category's schema. The goal of SCAD is to produce a tuple $\langle v_1, \ldots, v_n \rangle$ such that each v_i is associated to its corresponding a_i. We will talk about a snippet $c.s$ *being assigned* to an attribute a, or vice versa. This will mean that $c.s.v$ is a *candidate value* for attribute a. The reported value/s of an attribute depend on candidate values and other considerations such as global (exact or approximate) agreement.

Not all mentioned quantities will map to attributes in the schema. For those, we create a special **background/no attribute** NA (similar to "no assignment" [12]).

Our approach is to encode assignment decisions as 0/1 variables in a suitable integer linear program (ILP). Constraints for the ILP will come from some natural snippet and value assignment considerations. The objective will be designed using a measure of local compatibility between a snippet and an attribute.

4.2 Decision variables

We use two types of variables. The first type models the assignment of snippets to attributes. The assignment of snippet $c.s$ to attribute a is recorded in the binary variable $x(c.s, a) \in \{0, 1\}$. It is equal to 1 if attribute a is assigned snippet $c.s$; and 0 otherwise. Here a ranges over all attributes, including NA.

While the x variables associate snippets to attributes, the actual goal of catalog expansion is to associate *values* to attributes. This is denoted with the second type of decision variables: $z(v', a) \in \{0, 1\}$. It is set to 1 if attribute a is assigned any context $c.s$ with value $c.s.v = v'$, and to 0, otherwise. Because NA has no associated value, here a ranges over all attributes except NA. In contrast to current instantiations of the CCM framework [4, 18], these additional z variables enable SCAD to be more expressive and enforce additional domain-specific constraints.

4.3 Constraints

We introduce two kinds of constraints: *entity-level constraints* that keep the assignment of values to attributes globally consistent, and *category-level constraints* that prevent gross outlier values from being assigned to an attribute.

Entity-level constraints themselves come in two flavors: *consensus constraints* and *inter-attribute constraints*. The former model the agreement between multiple sources of evidence for the same attribute, and they are generic in the sense that the same constraints are used for all attributes and categories. The latter model relationships that can be enforced between values of different attributes using domain knowledge.

4.3.1 Entity-level consensus constraints

The following are the consensus constraints we use.

$$\sum_a x(c.s, a) = 1 \quad \forall c\, \forall q \qquad \text{(OneTargetAttr)}$$

This constraint ensures that each snippet $c.s$ is assigned at most one attribute from the schema. We have a strict equality because the sum is over all attributes, including NA. The intuition is that when the author of a Web page writes a value, she has a single attribute in mind. Even if the same *value* appears multiple times on a page, each mention (*snippet*) is associated to a different attribute. For example, suppose that a table (furniture) is square and both its depth and width are 3 feet. The value 3 feet might appear multiple times on the page. However, a snippet "Depth: 3 feet" is associated exclusively to the depth of the table.

For many attributes we can also assert:

$$\sum_v z(v, a) \le 1 \quad \forall a \qquad \text{(NoValueConflict)}$$

This constraint says that an attribute should be assigned at most one distinct value. (NoValueConflict) will not be appropriate if the product ID is insufficiently differentiated, as in a DVD player available in different colors, all of which are assigned the same product ID. In such cases, we can replace 1 on the rhs with a suitable upper bound to the number of distinct values we expect.

Snippet assignments induce value assignments, which we model as follows:

$$z(v', a) \ge x(c.s, a) \quad \forall a,\ \forall v',\ \forall c.s:\ c.s.v = v' \qquad \text{(ValueAssignment)}$$

This constraint ensures that if snippet $c.s$ is assigned to attribute a, then this attribute should be assigned the value $c.s.v$ embedded in the snippet. Essentially, it transfers the agreement from the snippets to the attributes.

Finally, we need to make sure that the assignments of values to attributes are sound: they must be backed up by at least one evidence snippet. This is done with the following constraint:

$$z(v', a) \le \sum_{c.s: c.s.v = v'} x(c.s, a) \quad \forall a, v' \qquad \text{(ValueEvidence)}$$

This constraint ensures that if attribute a is assigned value v', then attribute a should be assigned at least one snippet $c.s$ whose value $c.s.v$ is equal to v'.

4.3.2 Entity-level inter-attribute constraints

While the consensus constraints are generic, the inter-attribute constraints are assumed to be given together with the schema. For example, a reasonable constraint for the category *Tennis Racquets* is that the strung weight is always larger than the unstrung weight. Similarly, the hard disk capacity of a laptop is almost always (much) larger than the RAM size. Such domain knowledge can be modeled as follows.

$$\sum_v v \cdot z(v, \text{“strung weight”}) \ge \sum_v v \cdot z(v, \text{“unstrung weight”}) \qquad \text{(AttribValueComparison)}$$

This is just one example of a very general constraint template. If we know that hard disks are at least 20 times larger than RAM sizes, or that the battery life of a laptop is at least 2 hours, or the weight of a laptop is at least one pound more than the battery weight, (AttribValueComparison) can be easily adapted to represent such domain knowledge. Any such constraint has the potential to eliminate spurious assignments and improve accuracy.

Inequality (AttribValueComparison) has a problem as posed above. For brevity, let a_b and a_s be the "big" and "small" attributes, with decision variables $z(a_b, v)$ and $z(a_s, v)$, where we wanted $\sum_v vz(a_b, v) \ge \sum_v vz(a_s, v)$. The problem arises when a_b is not assigned, i.e., all $z(a_b, v) = 0$. Then, the lhs is 0 and so the rhs is also forced to be 0. Consequently, a_s is not assigned either.

To address this problem we add more constraints:

$$v_{\max}[1 - \sum_v z(a_b, v)]_+ + \sum_v vz(a_b, v) \ge \sum_v vz(a_s, v),$$

where $[C]_+ = \max\{0, C\}$ and $v_{\max}$ is the maximum candidate value for a_s. So if all $z(a_b, v) = 0$, the first term kicks in and allows a_s to be assigned. The extra constraint can be returned to benign linear form using another variable $c \in \{0, 1\}$, $c \ge 1 - \sum_v z(a_b, v)$ for all v, etc. It would be of interest to compile a catalog of such tricks for all common inter-attribute constraints.

4.3.3 Category-level constraints

Suppose we are expanding a catalog of steel nuts and bolts. The typical weight of a bolt, as seen in the training examples, may be between 1 and 5 grams. A human would unconsciously note this range. Faced with a page that offers a nut that can hold up a weight up to 1000 kilograms, a human would never be misled into interpreting 1000 kilograms as the weight of the nut itself. We can "fake" this form of intelligence without deep language understanding, by instead modeling a distribution of values over training data for each attribute, and then incorporating this distribution into the earlier integer program.

For a given attribute a, let $V_a = \{v_{a1}, v_{a2}, \dots v_{an}\}$ ($|V_a| = n$) be the set of values of attribute a corresponding to products in the training data. Now consider a test snippet $c.s$ with its value $c.s.v$. Intuitively, if $c.s.v$ is "close" to (the distribution estimated from) V_a, then we should feel more confident assigning this value to a.

We could measure the number of standard deviations by which $c.s.v$ differs from the average of values in V_a. This policy is not very robust in the face of multimodal distributions.

A more robust approach is to model a (Gaussian) kernel density on V_a, and then express the support at $c.s.v$ as the density at that point:

$$S(c.s.v, V_a) = \frac{1}{Z} \sum_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma_{ak}^2}} \exp\left(-\frac{(c.s.v - v_{ak})^2}{2\sigma_{ak}^2}\right),$$

where σ^2_{ak} is the variance of the k^{th} Gaussian, and Z is a constant to make sure $S(c.s.v, V_a) \in [0, 1]$. This is not necessarily desirable for multimodal data, so, for simplicity, we just set $Z = 1$ and call it *unnormalized support*. Also, we set $\sigma^2_{a1} = \sigma^2_{a2} = \cdots = \sigma^2_a$, to a value computed using Lashkari and Golland's [13] estimator.

We want an assignment of $c.s$ to a to be possible only if $c.s.v$ has adequate support from (the density of) V_a. We estimate an attribute-specific lower threshold τ_a of support below which an assignment is not allowed. This is implemented using the following simple added set of constraints:

$$(S(c.s.v, V_a) - \tau_a)\, x(c.s, a) \geq 0.$$

The threshold τ_a is estimated as follows:

$$\tau_a = \min_i \{S(v_{ai} - \alpha\sigma_a, V_a)\}$$

where $\alpha \geq 0$ is a constant, with $\alpha = 4$ in the current set of experiments. Roughly speaking, we allow values that are at most α standard deviations away from one of the values in V_a.

4.4 Objective

Multiple assignments will generally be feasible while satisfying all the above constraints. Which one should we prefer? Input to this decision comes from a *local model* for compatibility between a snippet $c.s$ and an attribute a, represented as a conditional probability model $\Pr(a|c.s)$ that we learn through a training process. We call this a *local* model because the strength of association depends exclusively on the content of the snippet, without taking other snippets (or contexts) into account.

The object of the ILP is to maximize local compatibility subject to all the constraints already discussed. We thus define the following objective function over choices of the binary decision variables (the sum over a includes NA):

$$\sum_{c.s,a} x(c.s, a) \log \Pr(a|c.s). \qquad \text{(Objective)}$$

We will sometimes shorthand $\phi(c.s, a) = \Pr(a|c.s)$.

4.5 Example

Now that we have introduced all the elements of the global optimization problem, we can illustrate it using the example presented in the Introduction. In that example, there was a snippet s_2 such that $\phi(s_2,$"unstrung weight") had a high value of 1. Given (Objective), the ILP would have an incentive to set $x(s_2,$"unstrung weight") to 1. Once this is done, (OneTargetAttr) precludes any other attribute from being associated to snippet s_2. That is, $x(s_2, a) = 0$, for every attribute a such that $a \neq$ "unstrung weight". Since $x(s_2,$"unstrung weight"$) = 1$ and $s_2.v =$ "11.7$oz.$", by constraint (ValueAssignment) z("unstrung weight", *"11.7 oz."*$) =$ 1. Recall that, according to the local model, snippet s_1 could be associated either with "unstrung weight" or "strung weight". However, by constraint (NoValueConflict), each attribute is assigned exactly one value. Thus, z("unstrung *weight", "12 oz."*$) = 0$. By the (ValueAssignment) constraint, $x(s_1,$"unstrung weight"$) = 0$ (it would otherwise set z("unstrung weight", *"12 oz."*) to 1 and reach a contradiction). Since snippet s_1 cannot be associated with "unstrung weight", the objective function now has a strong incentive to set $x(s_1,$"strung weight") to 1. By (ValueAssignment), z("strung weight", *"12 oz."*$) = 1$.

5. LOCAL MODEL

In this section, we present the details of the local model used by SCAD. In Section 5.1, we present the features used by the classifier; and in Section 5.2 we described the approach used for training it.

5.1 Features

In Section 4.1, we explained how the snippets are created. In order to associate a snippet to an attribute, the model takes into consideration various measures of matches between the text in the snippet and various elements such as word distributions in previously seen snippets, and attribute metadata (types, attribute name synonyms, etc.).

Each such match signal is called a **feature** and becomes an element in a feature vector $f(c.s, a) \in \mathbb{R}^d$ for some suitable vector space of d dimension. The overall compatibility between the context $c.s$ and the attribute a is obtained by a linear combination of the features through a **local model** $w \in \mathbb{R}^d$, and written as the dot/inner product $w^\top f(c.s, a)$.

To combine local compatibility of various $(c.s, a)$ pairs, we calibrate these to probabilities using a multiclass logistic regression

$$\Pr(a|c.s) = \frac{\exp(w^\top f(c.s, a))}{\sum_{a'} \exp(w^\top f(c.s, a'))}, \qquad (1)$$

where a is regarded as the class label. Often, feature elements are non-negative. We say a specific element of f "fires" if its value is positive, as against zero. We now describe the actual features used in our system.

Features based on attribute names.

For some attributes, we may know a priori some of the ways that they may be referred to in Web documents. For example, the attribute "Model Part Number" can appear as "MPN" and "Manufacturer Part Number".[4] For each attribute a, we create a feature as follows. Let Names(a) be the set of possible names that we have identified for attribute a. We then create a feature function that returns 1 if the input snippet contains an element from Names(a), and 0 otherwise.

Features based on word distributions.

The features based on attribute names need to be complemented with other features because it might be impossible to determine a priori all the different ways in which an attribute can be mentioned. Furthermore, some attribute values appear without any mention of an attribute name. Thus, we also employ softer measures of textual similarity between the snippet of interest $c.s$; and snippets from training contexts already known to be assigned to attribute a for other entities.

We use two type of features based on word distributions, using TF-IDF [14] and Jensen Shannon divergence [6] as measures. For the TF-IDF measure, we can consider all the snippet in the training data for an attribute a as a "document" for a. Then, we can find the tf-idf score of a word. Given a snippet $c.s$, we compute the tf-idf value corresponding to attribute a by summing over the scores of the words

[4]In our system, we use a method for detecting these attribute name synonyms based on distributional similarity, which is outside the scope of this paper.

w_i in $c.s$. That is,

$$\text{tf-idf}(c.s, a) = \sum_{w \in c.s} \text{tf-idf}(w, a),$$

where tf-idf(w, a) is computed with respect to the training snippets for attribute a.

To compute the features based on Jensen Shannon divergence, we do the following. For each attribute a, collect all training snippets associated to a in the training data into one word bag and represent it as a distribution P_a over words (unigrams). In particular, for a word w, $P_a(w)$ is computed as follows. Let x be the number of times that x appears in the training snippets for attribute a. Let y be the total number of tokens in those snippets. Then $P_a(w) = \frac{x}{y}$. Similarly, we compute distributions for the snippet of interest $c.s$ and denote it $P_{c.s}$. The feature is then computed as $JS(P_a || P_{c.s})$.

Features based on units.

Additional strong local clues come from the units of quantity attributes. Most quantities of interest in a catalog have units like money amount, linear dimension, duration, and weight. As additional evidence for matching up $c.s$ with a, we add features that detect the typical units in which snippet $c.s$ is written.

We start with some basic features that associated to the value $c.s.v$:

- Is the token a (decimal) integer?
- Does the number represented by the token have a fractional part?
- Is the token alphanumeric? Note that some attribute values like *802.11n* may not be purely numerical.

Then we add features that draw upon the typical units in which a specific attribute a is expressed. Consider the dimensions of cameras, specifically, width. It can be expressed in millimeters, centimeters, or inches. For larger objects like furniture, feet, meters or yards may be used. Some attributes, like the number of USB ports in a motherboard, are unitless counts. To capture both classes, we use these features:

- Does the snippet contain any unit?
- For each unit u, does the snippet contain u?

5.2 Automated Training

As a training set, SCAD is given a set of "gold" entities, together with their corresponding contexts. In order to train the local model classifier, it is also necessary to assign attribute labels to the snippets in the contexts. Doing so in a manual way would be overly laborious. Thus, SCAD does it an automated fashion. In particular, for each document (context), SCAD looks at the occurrences of values, and tries to relate them to the gold data that it has available for the corresponding entity. Let us say that we have the snippet $c.s$ associated to entity e. SCAD considers three cases:

1. The snippet has a numerical value $c.s.v$ whose unit is u. If the gold data for entity e contains an attribute value pair $\langle a, v' \rangle$ such that $v' = c.s.v$ and the unit of v' is u, then the snippet $c.s$ is labeled with attribute a.
2. If $c.s.v$ is not associated to any unit, SCAD also looks for occurrence of $c.s.v$ in the gold data for e. But as an extra assurance SCAD looks for suitable attribute names in the text of snippet $c.s$. More specifically, suppose that there is an attribute value pair $\langle a, v' \rangle$ in the gold data for e such that $v' = c.s.v$. The extra assurance is that SCAD checks whether one of the known names for a appears consecutive to the value $c.s.v$ in the text for snippet $c.s$. If that is the case, snippet $c.s$ is labeled with attribute a.
3. If none of the two cases above hold, then the snippet is associated to the the background class NA.

Category	Attributes
Tennis Racquets	Head Size, Length, Beam Width, Unstrung Weight, Strung Weight
Digital Cameras	Depth, Digital Zoom, Effective Camera Resolution, Flash Memory, Height, Interpolated Resolution, Max Focal Length, Min Focal Length, Optical Zoom, Screen Size, UPC, Weight, Width
Washing Machines	Number of Cycles, Number of Temperature Settings, Tub Capacity, Energy Used, Depth, Height, Width, UPC, Weight, Water Used, Max Spin Speed
Computer Memory	RAM Storage Capacity, RAM Memory Speed

Table 1: Representative set of product attributes used in the experiments.

Since the number of snippets labeled as background usually dominate, SCAD downsamples them, in such a way that, in the end, the training set contains as many background snippets as the number the snippets for all non-background attributes.

6. EXPERIMENTS

6.1 Experimental Framework

The experiments were conducted in the Commerce search domain, with the goal of building structured descriptions for commercial products. Each description becomes a record in the catalog maintained by the Commerce search engine. All the experiments were done using data from Bing Shopping.

We considered seven product categories: TVs, washing machines, microwave ovens, refrigerators, computer memory, digital cameras, and tennis racquets. A representative set of the attributes that we focused on is shown in Table 1.

Recall that in SCAD each entity is associated to a *context*. In order to obtain the contexts for each entity (product), we considered two scenarios of significant practical importance in Commerce search, described next.

6.1.1 *Contexts from merchant offers*

In this scenario, we make use of offer feeds provided by merchants to Bing Shopping. For each product, there is a set of merchant offers; and each offer is associated to a Web page where the product can be bought. The Web pages pointed to by the links become the contexts for the entities used in SCAD.

To understand why the offers and merchant Web pages are readily available, consider the business model of Commerce search engines. In this model, users search for a product, and once they find it, they are presented with a list of links to merchant Web pages where they can buy it. The Commerce search engine is paid by the merchants for each click to their sites, and the merchants benefit by increasing the visits to their site. Thus, both the Commerce search engine and the merchants have an strong incentive to ensure that products have associated offers.

6.1.2 Contexts from Web search

In this scenario, the Commerce Search engine wishes to build product descriptions even before the merchants provide offers for the corresponding products. One reasonable way of doing so is by leveraging the results of a general Web search engine. For example, if we want to create a description for the racquet Wilson BLX Khamsin-Five, we can issue the query "wilson blx khamsin-five" to the search engine, which would return a set of results related to the product. We can then fetch the pages corresponding to the results, and consider them as the contexts for the entity. Notice that the quality of the results depends on how effective the query is in retrieving documents that are relevant to the product. For example, the query "wilson racquets" would clearly be less effective than the query "wilson blx khamsin-five". In our experiments, we always construct the queries using the name for the product, which includes identifiers such as brand and model. This approach proved to be effective in our experiments; we leave the study of other query constructions for context retrieval as future work.

In SCAD, we assume the availability of a training set of products within a category. In all the experiments, we used a training set of 20 products per category. For six categories (TVs, washing machines, microwave ovens, refrigerators, computer memory, digital cameras), the attribute-value pairs for the training products were obtained from data existing in the Bing Shopping catalog. To stress the fact that the training data is easy to gather, for an additional category (tennis racquets) we obtained the training data directly from a site that specializes on racquets (`www.tennisracquets.com`). In the experiments, we report results for test sets of 60 products for each category, except for refrigerators and washing machines for which we used 45 and 35 products[5], respectively. When using merchant offers, we consider products that have at least eight offers associated to them. For Web Search results, we use the Bing search API to fetch top-50 results for each product/query.

6.1.3 Metrics

We measure the quality of the results by using the standard precision, recall, and $F1$ measure [14]. As a parametric knob, we use a threshold θ applied to the scores of the local model classifier. The metrics at each value of θ are then computed as follows. Let E be a set of entities (products). Let X be the set of attribute-value pairs predicted by a system (either SCAD or a baseline) for the products in E, when the local model outputs only the predictions whose score is above θ. Let Y be the ground truth attribute-value pairs for the products in E. Then, we define precision (P), recall (R) and F1 at θ as follows:

$$\mathrm{P}_\theta = \frac{|X \cap Y|}{|X|}, \quad \mathrm{R}_\theta = \frac{|X \cap Y|}{|Y|}, \quad \mathrm{F1}_\theta = \frac{2 \times \mathrm{P}_\theta \times \mathrm{R}_\theta}{\mathrm{P}_\theta + \mathrm{R}_\theta}$$

We note that the local model's prediction scores are not calibrated to confidence and to the diverse number of attribute options a snippet can have, and this may result in P-R plots that do not always show smooth inverse relations. For presentation purposes, we sometimes report the highest $F1$ measure that is obtained across values of θ (i.e., the highest $F1_\theta$, for $0 \leq \theta \leq 1$).

[5]This is due to the limited coverage of the catalog for the refrigerators and washing machines categories.

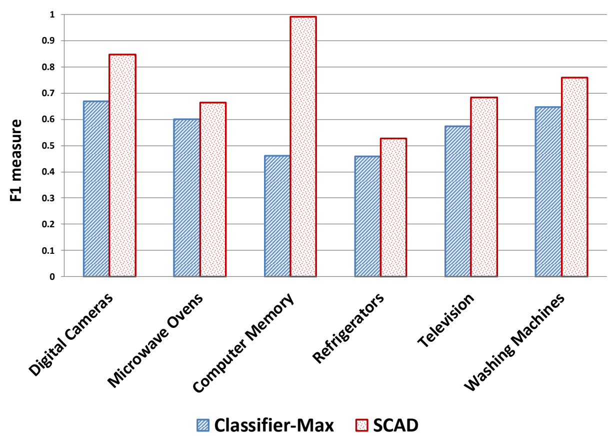


Figure 4: Comparison with Classifier-Max on all categories using merchant offers.

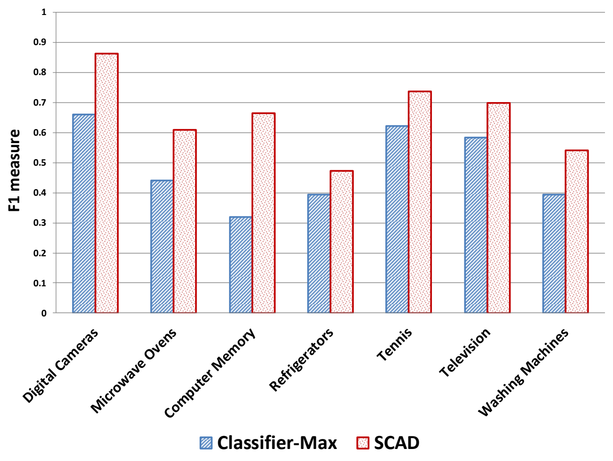


Figure 5: Comparison with Classifier-Max on all categories using Web search results.

6.2 Benefits of Collective Extraction

We now demonstrate the benefits of collective extraction by comparing SCAD against a baseline that makes assignments without requiring consensus among snippets. We call this baseline **Classifier-Max** because it uses the same local model (classifier) as SCAD, but instead of making a collective assignment of attribute values, it chooses for each attribute the value in the snippet with the the highest score according to the local model. More specifically, for each entity (product) e and attribute a, let C be the set of contexts associated to e. Then, **Classifier-Max** chooses a value $c.s.v$ such that $c \in C$ and there is no $c'.s'.v'$ such that $\phi(c'.s', a) > \phi(c.s, a)$. This baseline is inspired by the Kylin system [23], where the same approach is taken to com-

Dataset	NameMatch-Majority	Classifier-Majority	SCAD
Web search contexts			
Microwave Ovens	0.39	0.58	**0.60**
Computer Memory	0.24	0.65	**0.66**
Refrigerators	0.31	0.41	**0.47**
Digital Cameras	0.82	**0.86**	**0.86**
Tennis	0.59	0.61	**0.73**
Television	0.67	0.68	**0.69**
Washing Machines	0.51	0.52	**0.54**
Merchant offer contexts			
Microwave Ovens	0.36	0.64	**0.66**
Computer Memory	0.06	0.98	**0.99**
Refrigerators	0.50	0.49	**0.52**
Digital Cameras	0.82	**0.86**	0.84
Television	**0.68**	**0.68**	**0.68**
Washing Machines	0.67	0.70	**0.75**

Table 2: F1 comparison of SCAD with NameMatch-Majority and Classifier-Majority on all configurations. Best performance is marked in bold.

bine scores from a local model (although their local model is different from ours).

In Figures 4 and 5, we compare SCAD and **Classifier-Max** on contexts from merchant offers and Web search results, respectively. For each category, we show the highest $F1$ obtained across values of parametric knob θ. Notice that our system performs significantly better than **Classifier-Max** across all categories. The largest improvement is in the category Computer Memory on merchant offers, where the $F1$ increases by 0.52. For 10 out of the 13 category/source considered configurations, the increase is larger than 0.10; and in all cases the increase is above 0.06.

Except for Computer Memory, in all other categories the $F1$ increase on Web search results is even more pronounced than in merchant offers. For example, for Microwave Ovens, the increase on merchant offers is 0.06; whereas it is 0.17 on Web search results. The reason is that pages from Web search results are generally noisier than merchant pages: the latter tend to describe exactly one product and have prominent product descriptions. When the pages are noisier, the local model makes more mistakes, and provides greater room for improvement to the collective constraints.

To understand the benefits of SCAD in terms of precision and recall, see Figures 6 and 7. These figures present the precision/recall curves for Tennis Racquets, and Washing Machines on offers and Web search results, respectively. Notice that SCAD consistently outperforms **Classifier-Max** across values of the parametric knob. The reason for this is that **Classifier-Max** is rather brittle: a wrong local model prediction with high confidence leads to an error. On the other hand, SCAD is able to exploit redundancy in the contexts to recover from local model errors.

6.3 Justification of design decisions

We now compare SCAD against a number of baselines, where we aim for the following:

1. Evaluation of the Local Model
2. Understand the benefits of our ILP-based Global Optimization

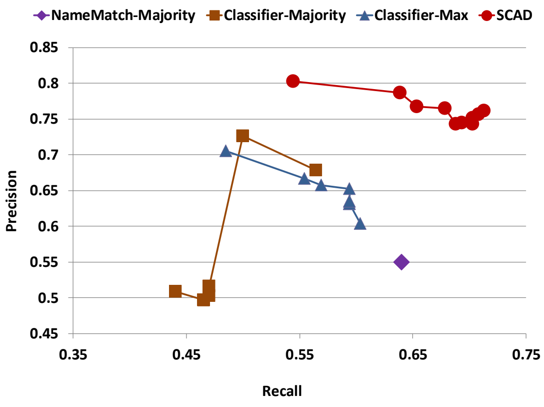


Figure 6: Precision and recall results for Tennis Racquets category.

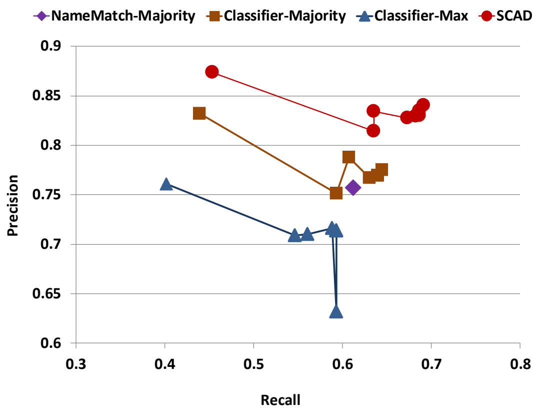


Figure 7: Precision and recall results for Washing Machines category on merchant offers.

3. Perform a constraint ablation study to show the need for the actual constraints used in the linear program
4. Understand the running time of the ILP

6.3.1 *Evaluation of Local Model*

While in SCAD the results of the local model are adjusted by the collective constraints, we must ensure the local model produces reasonable results. We show that this is the case by comparing against a system where the local model is not classification-based: rather, it associates value $c.s.v$ to attribute a if the name of attribute a appears in the text of snippet $c.s$. Furthermore, it performs aggregation by majority voting. We call this implementation **NameMatch-Majority**.

In Table 2, we show the highest $F1$ on all categories for SCAD and **NameMatch-Majority**. We can observe that SCAD consistently outperforms **NameMatch-Majority**. By examining the data, we observed that the improvement is more pronounced for categories where the attribute names do not appear explicitly on the Web pages, but SCAD's local model is able to get the signal from other features. For example, for Computer Memory, the attributes names are rare on Web pages (e.g., there is an attribute "RAM Memory Speed" but the word "RAM" never appears next to the attribute values). However, these attributes are easy to detect by the local model by using the distributional similarity features with respect to previously seen snippets (e.g., the term "speed" tends to appear near the memory speed) and features related to the units (e.g., memory speed is the only attribute whose unit is in Mhz.)

6.3.2 *Benefits of our ILP-based Global Optimization*

To show that the aggregation approach of SCAD is effective, we compare against an implementation that uses the same local model as in our system (i.e., classifier-based) but does aggregation by majority voting (unlike our system, which uses an ILP). We call this implementation **Classifier-Majority**. In Table 2, we show the $F1$ measure on all categories for our system and **Classifier-Majority**. Notice that with the exception of one case (Cameras on merchant offers), our system consistently outperforms **Classifier-Majority**. The largest improvement is on the Tennis Racquets category, where the $F1$ measure increases from 0.61 to 0.73. On Figures 6, and 7, we show the precision and recall for some of the categories (Tennis Racquets, Washing machines on Offers, and Washing Machines on Web search results). It can be seen that SCAD outperforms **Classifier-Majority** across all values of the parametric knob.

6.3.3 *Constraint Ablation Study*

In order to justify the need for the actual constraints used in the linear program, we considered an ablation of SCAD's constraints, where only the consensus constraints are used, but not the category-level constraints. We observed that for 12 out of the 13 category/source configurations, SCAD outperforms the ablated system. Furthermore, the ablated system outperforms **Classifier-Majority**. The differences are significant in some cases. For example, for Tennis Racquets, the ablated system has an F1 of 0.68 while SCAD has an F1 of 0.73. This shows that category-level constraints are useful. Furthermore, the ablated system outperforms **Classifier-Majority**, which has an F1 of 0.62. This shows that SCAD's consensus constraints help. As another example, for Washing Machines on merchant offers, the F_1 for **Classifier-Majority** is 0.70. When we use an ILP with consensus constraints, it jumps to 0.72. If we use the entire SCAD system, the F_1 measure is 0.75.

6.3.4 *Running time of ILP*

Recall that in SCAD, each entity has an associated ILP. That is, the system solves n ILPs, where n is the number of entities to be processed. In the different experiments, the average running time to solve an individual ILP was 37.5 ms. per entity. The reason we do not incur a higher cost is that we have one ILP per entity and the total number of variables per ILP is only 1323 on average. All this implies that use of ILP is efficient and practical in our framework.

7. CONCLUSIONS

We have presented SCAD, a general and flexible optimization-based framework for collective extraction of attributes from unstructured text. An important motivation for the problem is to expand and complete product catalogs in e-commerce sites. SCAD requires limited amount of supervision and can be bootstrapped quickly, even in the absence of data providers. Experimental results with several product categories from Bing Shopping demonstrate the merits of our approach. As part of future work, we plan to apply SCAD on more categories, exploiting different category-specific constraints, and exploring how such constraints can be estimated reliably from limited training data.

8. REFERENCES

[1] S. Banerjee, S. Chakrabarti, and G. Ramakrishnan. Learning to rank for quantity consensus queries. In *SIGIR Conference*, 2009.

[2] R. Bunescu and R. Mooney. Collective information extraction with relational Markov networks. In *ACL*, 2004.

[3] R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *EMNLP Conference*, pages 724–731. ACL, 2005.

[4] M. Chang, L. Ratinov, N. Rizzolo, and D. Roth. Learning and inference with constraints. In *AAAI*, 2008.

[5] T. Cheng, X. Yan, and K. Chang. EntityRank: searching entities directly and holistically. In *VLDB*, 2007.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley and Sons, Inc., 1991.

[7] N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *SIGMOD*, 2009.

[8] D. Davidov and A. Rappoport. Extraction and Approximation of Numerical Attributes from the Web. In *ACL*, 2010.

[9] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-scale information extraction in knowitall:(preliminary results). In *WWW*, 2004.

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

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

[12] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in Web text. In *SIGKDD Conference*, 2009.

[13] D. Lashkari and P. Golland. Convex clustering with exemplar-based models. In *NIPS Conference*, 2007.

[14] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[15] V. Moriceau. Numerical data integration for cooperative question-answering. In *EACL Workshop on Knowledge and Reasoning for Language Processing*, pages 42–49, 2006.

[16] M. Paşca. Organizing and searching the world wide web of facts–step two: harnessing the wisdom of the crowds. In *WWW*, 2007.

[17] K. Probst, R. Ghani, M. Krema, A. Fano, and Y. Liu. Semi-supervised learning of attribute-value pairs from product descriptions. In *IJCAI*, 2007.

[18] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *CoNLL*, 2004.

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

[20] C. Sutton and A. McCallum. Collective segmentation and labeling of distant entities in information extraction. In *ICML workshop on Statistical Relational Learning*, 2004.

[21] P. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *ECML*, 2001.

[22] M. Wick, A. Culotta, and A. McCallum. Learning field compatibilities to extract database records from unstructured text. In *EMNLP*, 2006.

[23] F. Wu and D. S. Weld. Automatically semantifying Wikipedia. In *CIKM*, pages 41–50, 2007.

[24] M. Wu and A. Marian. Corroborating answers from multiple web sources. In *WebDB*, 2007.

Coupled Temporal Scoping of Relational Facts

Partha Pratim Talukdar
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
ppt@cs.cmu.edu

Derry Wijaya
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
dwijaya@cs.cmu.edu

Tom Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
tom.mitchell@cs.cmu.edu

ABSTRACT

Recent research has made significant advances in automatically constructing knowledge bases by extracting relational facts (e.g., Bill Clinton-presidentOf-US) from large text corpora. Temporally scoping such relational facts in the knowledge base (i.e., determining that Bill Clinton-presidentOf-US is true only during the period 1993 - 2001) is an important, but relatively unexplored problem. In this paper, we propose a joint inference framework for this task, which leverages fact-specific temporal constraints, and weak supervision in the form of a few labeled examples. Our proposed framework, CoTS (Coupled Temporal Scoping), exploits temporal containment, alignment, succession, and mutual exclusion constraints among facts from within and across relations. Our contribution is multi-fold. Firstly, while most previous research has focused on micro-reading approaches for temporal scoping, we pose it in a macro-reading fashion, as a change detection in a time series of facts' features computed from a large number of documents. Secondly, to the best of our knowledge, there is no other work that has used joint inference for temporal scoping. We show that joint inference is effective compared to doing temporal scoping of individual facts independently. We conduct our experiments on large scale open-domain publicly available time-stamped datasets, such as English Gigaword Corpus and Google Books Ngrams, demonstrating CoTS's effectiveness.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms,Experimentation

Keywords

Temporal Scoping,Joint Inference,Knowledge Base

1. INTRODUCTION

There has been much research on extracting relational facts from both structured and unstructured text. Systems such as YAGO [21], KnowItAll [10], TextRunner [4], and NELL [7] gather entities and factual relations between entities from Web sources. However, much of the effort has been focused on gathering facts without their temporal scope. Facts are treated as time-invariant when in reality they dynamically change with time. New facts arise while others cease to be valid or change over time. Knowledge grows in various dimensions, and completely new entity types, relation types or knowledge structures may arise with time [28]. For example, the fact that Bill Clinton is a US President is only valid from the year 1993 to 2001. Such temporally scoped facts are useful for many reasons. Temporal information can be used as a dimension along which facts can be organized, ranked, or explored. Time can be helpful for relevancy ranking purposes. Presenting facts in a timeline can greatly benefit user experience in their exploration of knowledge evolution [2]. In a search or question answering system, time-sensitive queries such as business-intelligence queries (e.g., when did certain companies acquire other companies?) or medical queries (e.g., when did a certain vaccine become available?) will also benefit from temporally scoped facts. Temporal scoping of facts can also benefit other natural language applications, such as document summarization where the temporal information of facts mentioned in sentences can be used to generate better sentence ordering.

Despite the importance of time in any information space, gathering and distilling temporal knowledge from Web sources remains a major research challenge [28]. To the best of our knowledge, Timely YAGO [27] and PRAVDA [26], two recently proposed techniques, are the only systems which try to harvest temporal facts. Timely YAGO tries to automatically scope facts using regular expressions in Wikipedia infoboxes, and hence is not applicable to widely available free text. PRAVDA, a promising recent approach, uses a combination of textual patterns and graph-based re-ranking techniques to harvest facts and their temporal points at the same time. However, it is not immediately clear how this approach could be used to temporally scope facts in an existing knowledge base. Other works on temporal information extraction have tackled partial aspects of the problem such as temporal relations identification between events [6, 14, 5, 16, 8, 29, 13]. However, these other works are not sufficient to temporally scope facts as they are all focused on micro-reading of time at a single document or sentence level, i.e., temporal expressions and relations are normalized and identified based

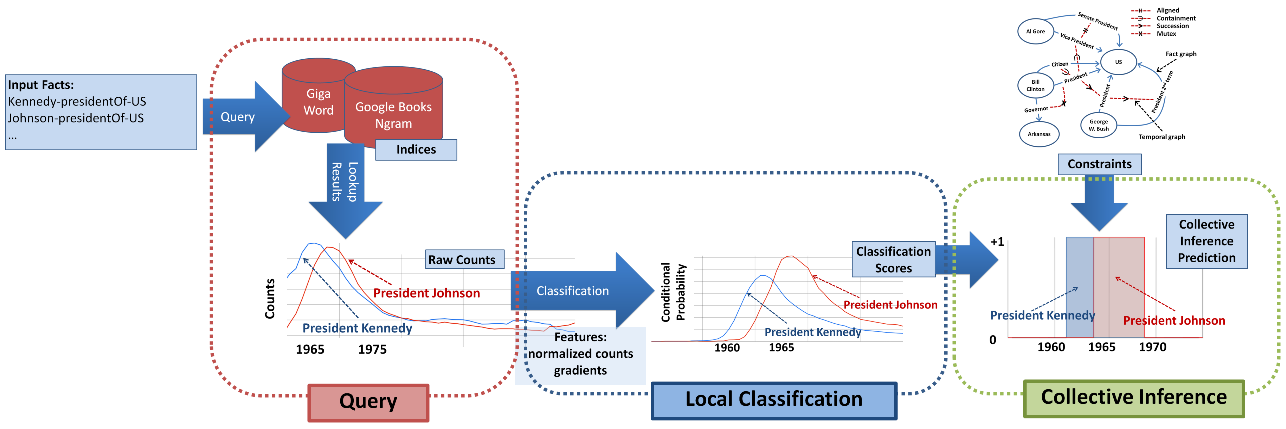


Figure 1: Architecture of the CoTS system (see Section 2 for an overview). Details of the Local Classification and Collective Inference modules are presented in Section 3.1 and Section 3.2, respectively.

only on features derived from the document. However, web is unique in that it typically offers ample redundancy, and hence it only seems natural to aggregate many observed cues for a temporal fact in a statistical manner [28]. This is what we attempt to do in our work, a macro-reading of temporal information from large-scale sources such as Google Books Ngram [15] and newswire text in Gigaword [12] using their document creation times (DCTs) to temporally scope facts.

Our contribution to this important but largely unexplored problem of temporal scoping is to propose a novel system, CoTS (Coupled Temporal Scoping), with multi-fold benefits.

- Firstly, in the spirit of macro-reading, CoTS uses a statistical approach to the problem, by using simple counts of facts in documents over time as cues to their temporal validity. Without going into document content, CoTS represents a fact by a time series of its counts over time. We believe time series is a natural way of representing a fact as it models directly the dynamic nature of the fact, its rise and fall over time. Evidence of change in time series is used as cues to classify whether the fact is active or not at any given time.

- Secondly, CoTS is novel in that we introduce collective[1] temporal inference over multiple temporally correlated facts to aggregate many observed cues for improved time scoping. Independent activation scores of facts are input to a collective inference framework, and Integer Linear Programming (ILP) is used to infer the temporal scope of facts while respecting various temporal dependencies among those facts, such containment, alignment, succession, and mutual exclusion.

- Thirdly, CoTS is weakly supervised: we need only a few labeled examples to train a local classifier (one for each relation), the only supervised component in CoTS. Moreover, CoTS provides a flexible framework where prior knowledge about the temporal dependencies among facts can be easily specified.

[1]In this paper, we shall use the terms collective inference, coupled inference, and joint inference interchangeably.

- Lastly, through experiments on interesting open-domain datasets such as the Google Books Ngram Corpus and the Gigaword Newswire Corpus, we demonstrate CoTS' effectiveness in improved temporal scoping, highlighting benefits of scoping multiple temporally related facts jointly, rather than scoping each such fact in isolation.

2. CoTS OVERVIEW

2.1 Challenges & Motivation

The problem of macro-reading the temporal scope of a fact from its counts in documents is a difficult one as these counts can be noisy, lagging in time, or sparse. A fact may still be found in documents (i.e., its count is not zero) even after it ceases to be valid. For example, some documents discuss the presidency of Kennedy even after his death. The document creation time may also lag behind a fact's activation time. For example, books about President Clinton may only be published some months after his inauguration (since books often take longer time to print and publish than news). Some facts are not mentioned enough in documents, leading to the sparsity of their counts. For example, unlike the President relation, the US Secretary of State relation may not be mentioned as frequently in documents.

These challenges of the problem motivate our approach. Firstly, to deal with the issue of time lag, especially in books data, we use also the change in the counts over time to capture the activation of a fact. A positive gradient in the time series of a fact's counts indicates that the fact is increasingly being talked about in documents, which may signal its activation time. Using positive gradients as additional information can capture change points based on the moment they start to be increasingly mentioned in data.

Secondly, to deal with the noise and sparsity of counts, we conduct collective inference to temporally scope several facts jointly. The advantage of doing collective inference is multifold. By utilizing temporal constraints between correlated facts, collective inference can bind the temporal scope of noisy facts by ensuring temporal consistency between facts, or inform the time scope of a sparse fact from the time scope of other, temporally correlated facts.

For example, in a functional relation (e.g., US presiden-

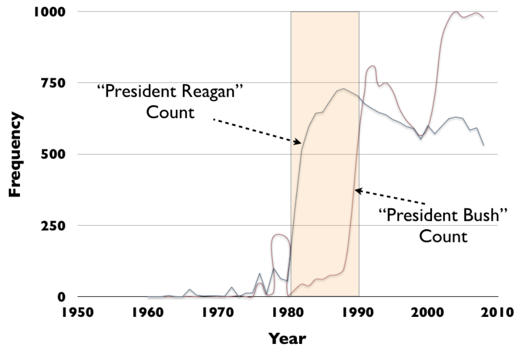


Figure 2: Temporal profiles of two facts demonstrating potential benefit of collective inference: considering 'President Bush' during inference can help determine the end of Reagan's presidency (shaded region).

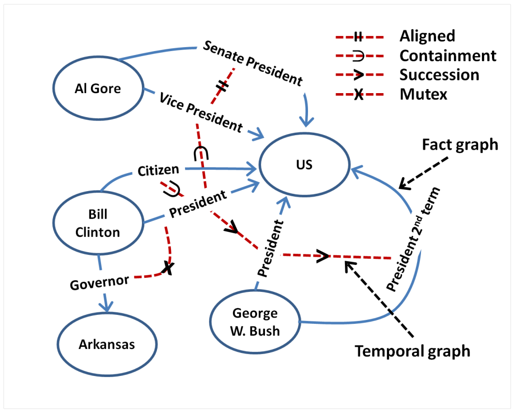


Figure 3: Temporal graph (dotted red edges) imposed over a factual graph (blue solid edges). Edges in the temporal graph correspond to constraints in CoTS, the proposed system.

tial relation), knowing the time scope of one instance of the relation can bind the time scope of another instance of the same relation since no two instances of a functional relation can be true at the same time (i.e., no two persons can be US president at the same time). This constraint is great to bind noisy, frequently mentioned facts. For example, although documents may still refer to Kennedy as 'President Kennedy' even after his death, knowing that Johnson's presidency began in 1963 and that his presidency succeeded Kennedy's can help bind the time scope of Kennedy's presidency. For example, in Figure 2, we can see how knowing the start of Bush's presidency can bind the end of Reagan's presidency, even though 'President Reagan' continues to be mentioned in documents even after his presidency.

Conversely, for facts that are sparse (e.g., those belonging to the US vice president relation), knowing the time scope of a correlated fact can help infer the time scope of the sparse fact. For example, the time scope of Clinton's presidency can be used to infer the time scope of Gore's vice presidency knowing that Gore served under Clinton's presidency.

Allowing users to specify temporal constraints between facts is also a natural way of adding prior knowledge to the task of temporal scoping. In Figure 3, we illustrate various temporal constraints that can be specified for US administration relations. Solid lines indicate factual relations (entity-relation-entity triplets) while dashed lines indicate temporal constraints between these facts. As we can see from Figure 3, temporal alignment between VicePresident and SenatePresident relations indicates that Al Gore must be vice president of the US at the same time that he is the president of US senate. Temporal mutual exclusion between President and Governor relations indicates that Bill Clinton cannot be both President of the US and Governor of Arkansas at the same time. Temporal containment between Citizen and President relations indicates that Bill Clinton must be a President of the US within the period that he is a citizen of the US. Temporal succession between facts indicates that the time scope of Bush-presidentOf-US follows the time scope of Clinton-presidentOf-US.

2.2 System Architecture

Figure 1 summarizes the high level architecture of our work to temporally scope facts, which involves the following pipeline of operations:

1. **Query**: To macro-read the temporal scope of a fact, we first construct a query to represent the fact. For example, the fact: 'Kennedy-presidentOf-US' is represented by the query 'President Kennedy'. Then we do a lookup of the query on the indices we have built for Google Books Ngram and Gigaword datasets. We gather raw counts of the results: i.e., the number of n-grams containing the query (for Google Books) and the number of times a query is found in the news documents (for GigaWord). Each fact is thus represented by a time series of its query counts, normalized over its total query counts.

2. **Local Classification**: Normalized counts and gradients of each fact are then input to a Maximum Entropy classifier which computes the conditional probability that the fact s from relation r is active at a given time t, i.e., $p_r(+1|s,t)$. As discussed previously, gradient information is included as a feature in this classification to capture the informative signal of increasing counts which may indicate the start of the fact's activation. Using only normalized counts and where they peak to classify start time may not be sufficient, as the peak may not always coincide with the start time due to possible time difference (lag) between the actual start time and the document creation time.

3. **Collective Inference**: Individual classification scores of the facts, $p_r(+1|s,t)$ and $p_r(-1|s,t)$, are then input to a collective inference engine (in our case, an Integer Linear Program (ILP)) together with the temporal constraints we have described and illustrated in Figure 3. ILP then predicts which facts are active at which times, based on the input classification scores and the specified temporal constraints. The outputs are then the facts and the times for which they are active.

3. SYSTEM DESCRIPTION

In this section, we present detailed descriptions of the three main modules in the CoTS system.

Notations

- Let $\{S^1, \ldots, S^m\}$ be sets of facts, one set for each of the m relations. We would like to temporally scope each fact $s \in S^r$, $\forall 1 \leq r \leq m$. In this paper, we shall use the relation index, r, also to refer to the relation name itself.
- Let B and E be the beginning and end of the timespan over which the temporal scoping is currently performed. We shall assume that any time instant $t \in \{B, \ldots, E\}$ is discretized at an appropriate granularity, which for the experiments in this paper is at the year level.
- $p_r(+1|s,t)$ is the local classification score specific to relation r, which measures whether the fact $s \in S^r$ is true at time t. Please refer to Section 3.1 for a discussion on estimating such scores.
- $x_{r,s,t} \in \{0, 1\}$ is a binary integer variable, where $x_{r,s,t} = 1$ indicates that fact $s \in S^r$ is true at time $t \in \{B \ldots E\}$, and false otherwise.
- $z^b_{r,s,t} \in \{0, 1\}$ is a binary integer variable, where $z^b_{r,s,t} = 1$ indicates that fact s started to be true at time t.
- Similarly, $z^e_{r,s,t} \in \{0, 1\}$ is a binary integer variable, where $z^e_{r,s,t} = 1$ indicates that fact s ceased to be true at time t.

3.1 Local Classification Scores

In this section, we describe how we estimate the initial score, $p_r(+1|s,t) \in [0, 1]$, which measures the probability of fact $s \in S^r$ being true at time t. We obtain these initial scores by training a Maximum Entropy (MaxEnt) classifier, separately for each relation r. For each relation r, we assume that we are given a set of training instances $T_r = \{(\phi_r(s,t,y), y)\}$, where $\phi_r(s,t,y) \in R^d$ is the d-dimensional representation of fact $s \in S^r$ at time t, and $y \in \{-1, +1\}$ is its label, with $y = 1$ suggesting that the fact is true at time t, and -1 otherwise.

For each relation r, these labeled instances are then used to estimate parameters $\mathbf{w}_r$ of corresponding MaxEnt classifier. The classifier can then be applied to classify an unlabeled instance u at time t using Equation 1.

$$p_r(+1|u,t) = \frac{\exp(\mathbf{w}_r^T \cdot \phi_r(u,t,+1))}{\sum_y \exp(\mathbf{w}_r^T \cdot \phi_r(u,t,y))} \quad (1)$$

This score gives an estimate of how likely it is that fact u from relation r is true at time t. We note that $p_r(+1|u,t)$ is a probability distribution and hence it is bounded in $[0, 1]$. We use both positive and negative real numbers as the values for our features. The actual features that we use for the experiments in this paper are described below.

Features:

Using the Query module described in Section 2.2, we obtain a timeline L_u for each fact u, where $L_u(t)$ returns the (normalized) count of fact u at time t. We use $L_u(t)$ as the **count feature**. We also compute the gradient over L_u at t, and use the gradient value as the **gradient feature**. This feature helps in adapting the classifier to adapt to time-lag issues as described in Section 2.1. We use only these two features in the local classifier. Since the local classifier estimates its parameters from a very small number of labeled instances, we wanted to make sure the the local classifier doesn't overfit severely which is possible in the presence of large numbers of features.

3.2 Collective Inference

In our system, time scoping each fact in isolation corresponds to making the final scoping decisions based purely on the scores $p_r(+1|u,t)$ as described in previous section. While such scores provide useful discriminating information, they are often noisy, leading to incorrect time scoping decisions. Lack of discriminating features, paucity of labeled training data, and inherent hardness of the classification task itself could be one of several reasons leading to these noisy predictions. While obtaining large amounts of labeled data may not be practical, incremental improvements may be possible by employing more sophisticated features. We take a different approach and as motivated in previous sections, we introduce here a way to use these prediction scores, and at the same time exploit available domain-specific constraints. To this effect, in this section, we present the Integer Linear Program (ILP)-based collective inference module of our system. First, we present the constraints that can be specified in our system, and then present the objective which is ultimately optimized subject to the constraints.

3.2.1 Constraints

Constraints in the CoTS system can be categorized into the following two classes:

1. **Intra Relation Constraints**: These constraints regulate the temporal scoping of one or more facts from a single relation. For example, Functional constraints (described below) belong to this category, which can enforce the requirement that at most one fact from the relation can be true at any given time. For example, there is only one US President at any given time. Please note that such requirements can be enforced only if multiple facts from the same relation are considered jointly during temporal scoping, which is strictly not possible in case of temporal scoping of each fact in isolation.
2. **Cross Relation Constraints**: Constraints in this category couple temporal scoping of facts from multiple relations. For example, we may want to enforce the requirement that Al Gore's Vice Presidency aligned exactly with Bill Clinton's Presidency. Aligned constraints (described below) operating over the Vice President and President relations can be used to enforce such requirement.

Please note that all our constraints are specified at the fact level, even though they may affect one or more facts from the same or multiple relations. Below, we present examples of different constraints exploited by the CoTS system for the experiments in this paper.

- Consistency: The following constraints make sure that the begin ($z^b_{r,s,t}$) and end ($z^e_{r,s,t}$) variables are consistent with the $x_{r,s,t}$ variables.
  - BeginConsistency1: For a fact to start at a given time, the fact should also be true at that time.

$$z^b_{r,s,t} \leq x_{r,s,t},\ \forall s \in S^r,\ t \in \{B, \ldots, E\}$$

- BeginConsistency2: For a fact to begin at time t, the fact should not be true at time $t-1$ and become true at t. We use the boundary condition $x_{r,s,(B-1)} = 0$.

$$z^b_{r,s,t} \geq x_{r,s,t} - x_{r,s,(t-1)},\ \forall s, t \in \{B \ldots E\}$$

- EndConsistency1: For a fact to end at a given time, the fact should also be true until that time.

$$z^e_{r,s,t} \leq x_{r,s,t},\ \forall s \in S^r, \text{and } t$$

- EndConsistency2: For a fact to end at time t, the fact should be true at time t and not true at $t+1$. We use the boundary condition $x_{r,s,(E+1)} = 0$.

$$z^e_{r,s,t} \geq x_{r,s,t} - x_{r,s,(t+1)},\ \forall t \in \{B \ldots E\}$$

- Functional: For a given relation r, these constraints enforce the requirement that no two facts from r be true at the same time.

$$\textstyle\sum_s x_{r,s,t} \leq 1, \quad \forall t \in B \ldots E$$

As an example, Functional constraints can be used to enforce the fact that there can be at most one Vice President in USA at any given time. Functional constraints are Intra Relation in nature.

- SingleSpan: These constraints make sure that any fact from a relation r is true continuously for a single span of time, without any interruption in between. For example, US presidencies tend to be a single continuous span of time. SingleSpan, another member of the Intra Relation constraint class, is actually a set of constraints, which we describe below. Below, we shall assume that the fact belongs to relations r.

  - SingleBegin: For each fact, there is at most one beginning. This is "at most" and not "equal" as the fact may not be activated during the time interval $B \ldots E$.

$$\textstyle\sum_t z^b_{r,s,t} \leq 1,\ \forall s \in S^r$$

  - SingleEnd: For each facts, there is at most one end.

$$\textstyle\sum_t z^e_{r,s,t} \leq 1,\ \forall s \in S^r$$

  - EndAfterBegin: End of the fact should happen after its beginning.

$$\textstyle\sum_t t * z^e_{r,s,t} - \sum_t t * z^b_{r,s,t} \geq 0,\ \forall s \in S^r$$

- Point: This constraint is useful when the fact is true only at one instant of time, i.e., the temporal span has unit length. For example, even though Steven Spielberg has multiple Academy Award for Best Director, length of temporal scope of each win is of unit length, at the year granularity.

$$\textstyle\sum_t x_{r,u,t} \leq 1,\ \forall u \in S^r$$

Please note that this is different from the Functional constraint as the sum here is over all $t \in \{B \ldots E\}$, as opposed to all facts s in case of Functional. Point constraints are Intra Class.

- Aligned: This constraint is useful whenever two facts (from same or different relations) have exactly same temporal span. For example, George H.W. Bush was the Vice President throughout Ronald Reagan's presidency. In other words, if facts $u \in S^a$ and $v \in S^c$ are temporally aligned, then $x_{a,u,t} = x_{c,v,t},\ \forall t$. We enforce this through two inequality constraints.

$$x_{a,u,t} \leq x_{c,v,t}, \text{ and } x_{c,v,t} \leq x_{a,u,t},\ \forall t \in \{B \ldots E\}$$

- Containment: Suppose, we want to express the fact that Al Gore's Vice Presidency was contained within Bill Clinton's Presidency, even though we don't exactly know the time span of either fact. The Containment temporal constraint can be used to achieve this. If the timespan of fact $u \in S^a$ is contained within the time span of fact $v \in S^c$, then we the Containment constraint is of the form:

$$x_{a,u,t} \leq x_{c,v,t},\ \forall t \in \{B \ldots E\}$$

We can verify that this constraint will be violated only when $x_{a,u,t} = 1$ and $x_{c,v,t} = 0$, the only undesirable case. Containment can be both Intra Relation as well as Cross Relation.

- Succession: This constraint is useful when we want to express the requirement that one fact ($v \in S^c$) from relation c happened after another fact ($u \in S^a$) from relation a, e.g., Ronald Reagan became president after Henry Kissinger was the Secretary of State.

$$\textstyle\sum_t t * z^e_{a,u,t} - \sum_t t * z^b_{c,v,t} \leq 0$$

Please note that the above constraint is effective only in conjunction with SingleSpan constraint.

- Mutex: Suppose, we want to enforce the requirement that George H. W. Bush can't be US President and Vice President at the same time. This can be achieved through the Mutex constraint. This constraint ensures that two facts, $u \in S^a$ and $v \in S^c$, are not true at the same time.

$$x_{a,u,t} + x_{c,v,t} \leq 1,\ \forall t \in \{B \ldots E\}$$

Mutex can be both Intra ($a = c$) as well as Cross Relational ($a \neq c$).

Please note that choice of constraints finally used in any given inference is dependent on the type of relation(s) involved. Moreover, the temporal constraints presented above are not meant to be exhaustive, as depending on the domain and type of relation(s), new constrains may have to introduced. We hope that CoTS' linear constraint specification is flexible enough to support a majority of such extensions.

3.2.2 Objective

Subject to the constraints mentioned above, CoTS optimizes the following objective,

$$\max_{\{x_{r,s,t}\}} \sum_{r,s,t} p_r(+1|s,t) * x_{r,s,t} + \lambda * p_r(-1|s,t) * (1 - x_{r,s,t})$$

where $\lambda \in [0, 1]$ is the tradeoff weight which controls the relative importance of the two terms in the objective. The first term in the objective encourages the optimization to respect the classification scores $p_r(+1|s,t)$; while the second term encourages the inference to abstain from over-predicting in case the local MaxEnt classifier is not confident enough. This become more apparent after the following analysis.

$$\begin{aligned}
&p_r(+1|s,t) * x_{r,s,t} + \lambda * p_r(-1|s,t) * (1 - x_{r,s,t}) \\
&\quad = \quad p_r(+1|s,t) * x_{r,s,t} + \lambda * (1 - p_r(+1|s,t) * (1 - x_{r,s,t}) \\
&\quad = \quad ((1+\lambda) * p_r(+1|s,t) - \lambda) * x_{r,s,t} + \lambda * (1 - p_r(+1|s,t)) \\
&\quad = \quad ((1+\lambda) * p_r(+1|s,t) - \lambda) * x_{r,s,t} + \text{Constant}
\end{aligned}$$

The optimization can ignore the second constant term as it is not dependent on $x_{r,s,t}$. Since the objective is one of maximization sense, there is incentive to set $x_{r,s,t} = 1$ (subject to constraint satisfaction) iff

$$\begin{aligned}
(1+\lambda) * p_r(+1|s,t) - \lambda \quad &> \quad 0 \\
p_r(+1|s,t) \quad &> \quad \frac{\lambda}{1+\lambda}
\end{aligned}$$

With $\lambda = 1$, we observe that $x_{r,s,t} = 1$ is a candidate for prediction if $p_r(+1|s,t) > 0.5$, i.e., $p_r(+1|s,t) > p_r(-1|s,t)$. By varying λ, we can control how much confidence collective inference rests on the local classification scores. It is interesting to note that all these fall out naturally from the formulation of the objective. This built-in mechanism against over-prediction is a desirable property of CoTS, which is lacking in the other systems we compare against, and as we shall see in Section 4, it results in better temporal scoping.

4. EXPERIMENTS

In this section, we investigate the following:

- Does adding gradient-based features in the local classifier lead to improved temporal scoping? (Section 4.2)
- Does coupled temporal scoping (i.e., joint inference involving multiple instances from same as well different relations) help improve performance? (Section 4.3)
- What are the effects of different types of constraints on CoTS's performance? (Section 4.4)
- Is CoTS's Collective inference module fast enough? (Section 4.5)

4.1 Experimental Setup

4.1.1 Relations and Facts

Relation names and the number of facts from these relations which were used in the experiments in this section are presented in Table 1. Please note that CoTS assumes that these facts have already been extracted, and it instead focuses only on temporally scoping them. Facts from the US Administration domain (i.e., facts from the relations US President, US Vice President, and US Secretary of State) were temporally scoped together, while facts from the Academy Awards domain (i.e., Best Director and Best Movie) were scoped jointly. True temporal scope of each fact was determined manually, which in turn was used for training and evaluation purposes. Facts from the US Administration domain spanned the period 1961 - 2008, while facts from the Academy Awards domain spanned the period 1995 - 2008.

Relation Name	Number of Facts
US President	9
US Vice President	12
US Secretary of State	27
Best Director	14
Best Movie	14

Table 1: Relations and the number of facts from these relations used in the experiments in Section 4.

4.1.2 Query Module

Queries: As described in Section 2, in order to get the temporal profile of a fact (see Figure 2), we first construct a query to represent the fact. In the following description, we shall use the notation {q} to denote a query, whose keywords and operators are stored in q as per Lucene's query syntax [11]. For the three US Administration office relations in Table 1, we use the query template: Office LastName. So, the fact Bill Clinton-presidentOf-US is represented by the query {"president clinton"}. Similarly, {"vice president gore"}, and {"secretary albright"}. For the Academy Awards Best Movie relation, we used the query template {"academy award" AND movieName}, which resulted in queries of the form {"academy award" AND "a beautiful mind"}. The goal behind using this query is to retrieve documents containing both phrases: "academy award" and "a beautiful mind". Similarly, for the Academy Award Best Directory relation, the query template {"academy award" AND directorLastName} was used, which resulted in queries of the form {"academy award" AND "spielberg"}.

Time-Stamped Corpus: Once a query is generated from a fact as described above, we construct a temporal profile of the fact from the document creation times (DCTs) of documents retrieved by its query. In order to retrieve such documents for a given query, we use Lucene [11] to index time-stamped documents and look the query up against that index. We use the following two sources of time-stamped documents:

- **Google Books Ngram**[2] [15] is a corpus of about 5 million digitized books. The resulting corpus contains over 500 billion words in several languages. The data is released in the form of n-gram (n = 1 to 5) and three different counts of each of these n-grams are available per year: (1) the number of times the n-gram is found in books published in the year; (2) the number of pages that contain the n-gram in books published in the year; and (3) the number of books published in the year that contain the n-gram. An interesting feature of this data set is its extended time coverage spanning from 1500s to 2008.
- **Gigaword**[3] [12] contains English newswire text data acquired from four international newswire services, spanning the period from 1994 to 2008. Unlike Google Books Ngrams, Gigaword contains full time-stamped newswire documents and not just ngrams extracted from them. However, compared to the Google Books Ngrams dataset, it is much smaller in size and has smaller time coverage.

[2]http://books.google.com/ngrams/datasets
[3]LDC Gigaword: http://bit.ly/vZFoJ6

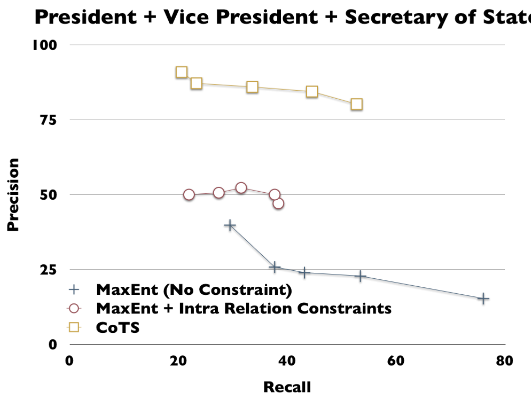


Figure 4: Precision-Recall plot for temporal scoping of US President, Vice President and Secretary of State relations. CoTS is the system proposed in this paper. (see Section 4.3)

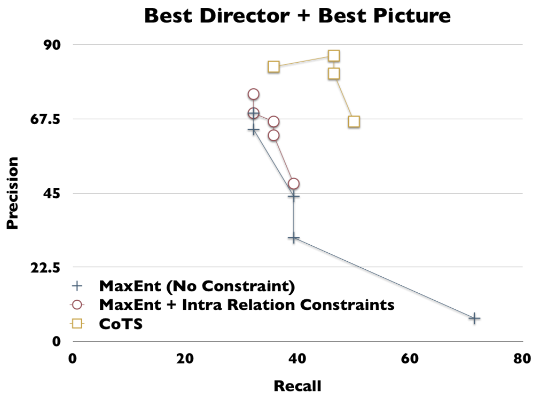


Figure 5: Precision-Recall plot for temporal scoping of Academy Award Best Director and Best Picture relations. CoTS is the system proposed in this paper. (see Section 4.3)

In case of Google Books Ngram dataset, each (unique) ngram is considered a document. Per year data from these two datasets are (separately) indexed using Lucene as mentioned above. For Google Books Ngrams dataset, we index only the English 5-grams published during the period 1960 - 2008. We consider 5-grams as they can accommodate longer queries. We use the Google Books Ngram dataset for the three US Administration relations in Table 1, as temporal scope of facts from these three relations span 1961 - 2008, which is beyond the temporal coverage of Gigaword (1994 - 2008). We use the Gigaword corpus for the two Academy Awards relations in Table 1, as the combined temporal span of facts from these two relations (as considered in this section) are aligned with that of Gigaword.

4.1.3 Training Local Classifier

Once the temporal profiles of facts are generated as described above, we next train a relation-specific local classifier by deriving features from these temporal profiles (Section 3.1). For each relation experimented with in this paper (Table 1), we use a single temporally scoped fact from this relation to generate all training data needed to train the corresponding relation-specific local classifier (MaxEnt, see Section 3.1). For example, for the US President relation, we derive training data from the temporally scoped fact: President Kennedy (1961 - 1963). In this case, all instances in the span [1961,1963] correspond to positive instances ($y = +1$), while everything outside this range correspond to negative instances ($y = +1$). Please note that all the classifiers in these experiments are trained from such limited amount of training data, demonstrating the real-world applicability of CoTS, the proposed method.

4.1.4 Metrics

As mentioned in Section 4.1.1, true temporal scopes of all facts in Table 1 were determined by a human annotator. Prediction from CoTS (or any of the baselines) that a fact is true at a given time is matched against this gold-standard to determine prediction correctness. Based on this, Precision (P), Recall (R), and F1 ($\frac{2*P*R}{(P+R)}$) are computed, which are used as the final evaluation metrics. Also, all evaluations are performed at the year level, as that is the finest granularity common to the two time-stamped datasets (Section 4.1.2).

In order to generate different Precision-Recall plots, we vary the λ parameter over the local classifier's scores, as described in Section 3.2.2.

4.2 Effect of Gradient Feature

Relations	F1 (No Gradient)	F1 (With Gradient)
President, V. President, Sec. of State	42.8	**63.63**
Best Director, Best Picture	4.26	**60.47**

Table 2: Effect of using gradient information during temporal scoping of different relations from two domains.

In this section, we evaluate the effect of gradient-based features in the local classifier (Section 3.1) can have on collective inference. We use the full CoTS with and without gradient features in the local classifier, and apply it on relations from two domains. F1 results are presented in Table 2. We observe that gradient features can significantly improve temporal scoping performance. This validates our motivation behind using gradient-based features in the local classifier, as it can help get rid of the publication lag issues in the books data and for certain domains in the newswire.

Based on these results, in all subsequent experiments, we include gradient-based features in CoTS's local classifier.

4.3 Effect of Coupling Constraints

In this section, we evaluate the effect of Cross Relational coupling constraints on temporal scoping. We compare the performance of CoTS against other systems exploiting either no constraint (MaxEnt, the local classifier in Section 3.1), or a subset of constraints (Intra Relation, Section 3.2.1) but without any cross relation coupling constraints. Examples of a few manually-specified coupling constraints used by CoTS for these experiments are presented in Table 3.

President, Vice President, Secretary of State		
Fact	Temporal Constraint	Fact
President Clinton	Containment	Vice President Gore
President Reagan	Containment	Vice President Bush
President Reagan	Succession	Secretary Kissinger
Best Director, Best Picture		
Director Cameron	Aligned	Titanic
Director Howard	Aligned	A Beautiful Mind

Table 3: Examples of coupling constraints used by CoTS for the experiments in Section 4.3. For brevity, we represent the facts by the queries used to compute their counts from the timestamped datasets (Section 4.1.2).

The Precision-Recall plots for these comparisons over facts from the two sets of relations in Table 1 are reported in Figure 4 and Figure 5, respectively. From these plots, we observe that CoTS, which exploits cross relational coupling constraints, significantly outperforms other baselines which either don't use any constraint (MaxEnt), or use only a subset of constraints exploited by CoTS.

4.4 Constraint Ablation Study

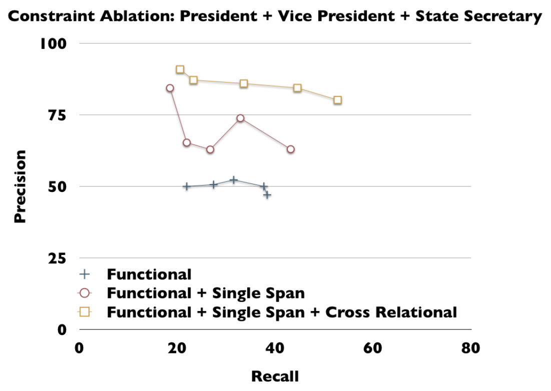


Figure 6: Constraint ablation results for temporal scoping over US President, Vice President and Secretary of State relations.

In this section, we report results from constraint ablation studies involving CoTS. The goal is to study the effect of different constraint types on CoTS's performance. Results from two different sets of relations are reported in Figure 6 and Figure 7. For the politics domain (President, Vice President, Sec. of State), we use the SingleSpan constraint as facts in that domain tend to be true for a contiguous single span (e.g., presidency usually lasts multiple consecutive years, and one is president for a single span). This is in contrast to the temporal spans of the two relations in the movies domain whose spans are usually a unit length, and where one could win the same award multiple times in one's lifetime, and hence the choice of the Point constraint.

From the results in Figure 6 and Figure 7, we observe that CoTS's performance improves as more prior knowledge is injected through additional coupling constraints. This

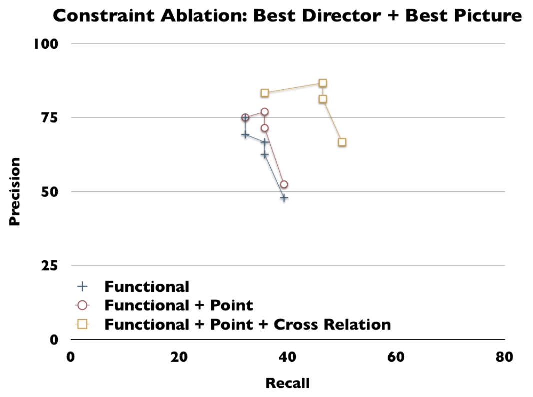


Figure 7: Constraint ablation results for temporal scoping over Academy Award winner Best Director and Best Picture relations.

justifies our design choice of using coupled (joint) inference for more accurate temporal scoping.

4.5 Running time of ILP

For all the problems in our experiments, CoTS's collective inference took on average 0.25 sec, with inference over the largest problem involving 2334 variables and 3804 constraints taking only 2.6 sec. This demonstrates the practicality of CoTS's inference scheme.

5. RELATED WORK

Even though time is an important dimension in any information space, and knowledge of temporal scopes of facts can be useful for better information retrieval systems, and user experience [2], temporal scoping of knowledge base facts is an area that is still largely unexplored. Only recently, a few research papers have started to address this problem [13, 27, 26]. Otherwise, most previous work on temporal information extraction has been focused largely on temporal representations [17], temporal relation identification [6, 14, 5, 16], and ensuring temporal consistency between extracted temporal relations [8, 29, 13]. We shall review some of the more recently proposed (and more relevant) approaches first, and then compare CoTS against the other previous work on temporal relation identification.

Timely YAGO [27] is similar in spirit to CoTS in that its objective is also temporal grounding of facts. As in CoTS, a fact in Timely YAGO refers to an instance of a binary relation between entities (entity-relation-entity triplet). Facts and their temporal information in Timely YAGO are extracted from semi-structured text in the infoboxes and category information in Wikipedia using regular expression matching. In contrast, CoTS is applicable more widely and is not limited to Wikipedia texts alone. Moreover, instead of using any regular expression based extractors, CoTS aggregates document-level metadata (viz., document creation time) based evidences to temporally scope multiple temporally related facts at the same time.

A method to temporally scope facts and reason over such scoped facts is presented in [25]. Given a target fact, the method in [25] attempts to gather count-based evidence for the begin, active and end time of this fact. These possibly inconsistent evidences are then aggregated using a set of heuristics to determine the final interval over which the target fact is likely to be true. Instead of temporally scoping each fact in isolation as in [25], CoTS performs joint inference to temporally scope multiple facts at the same time while exploiting temporal dependencies among those facts, resulting in more accurate temporal scoping as demonstrated through the experiments in this paper.

The Temporal Information Extraction (TIE) system [13] attempts to output a maximal set of events and their temporal relations as directly implied in a given sentence. TIE uses joint transitivity inference to bind the start and end times for each event. As in TIE, and instead of Allen-style intervals [1], CoTS uses a real-valued point-based temporal representation. While TIE is a micro-reading system which processes a single (or a set of) sentences at once, CoTS works at the macro-reading level, aggregating evidences from a large number of documents to temporally scope a set of facts.

PRAVDA is a recently proposed method to harvest temporal facts from free text [26]. PRAVDA uses textual patterns to generate candidate temporal facts, which are then re-ranked using a graph-based label propagation algorithm adapted from a recently proposed graph transduction algorithm [22]. PRAVDA is probably the prior work which is closest in spirit to CoTS. However, unlike PRAVDA, CoTS exploits temporal dependencies among facts to scope them jointly, a strategy that we have found to be quite effective as demonstrated in Section 4.

We note that Timely Yago [27], TIE [13], and PRAVDA [26] are complementary to CoTS, as extractions from these three systems can be used as additional evidence (features) in CoTS's local classifier, with potential for more accurate temporal scoping.

Learning and inference in CoTS is similar in spirit to the CCM framework [9, 20]. However, unlike existing CCM models, CoTS uses additional variables to define more expressive constraints, and applies them to the novel problem of temporally scoping relation facts, which is beyond the scope of current CCM-based models. Similar to CoTS, the SCAD system [3] also uses ILP to improve on the predictions from a weakly-supervised local classifier. However, the two systems address very different problems with completely different motivations.

Previous works on temporal relation identification have been largely spurred by the release of TimeML [17], a notable markup language for events and temporal expressions in natural language, the availability of tools such as TARSQI [24] to automatically annotate events and times in TimeML, and the release of TimeBank [18], a TimeML annotated corpora for training and testing. The TimeBank corpus consists of 186 news articles that are annotated for events, time expressions and temporal relations between events and times. Most previous work on temporal relation identification is based on the TimeBank corpus [6, 14, 8]. These approaches build temporal relation classifier over the corpus relations and making either local pairwise decisions of temporal relations between events [6, 14], or jointly informed decision to impose transitivity constraints (A before B and B before C implies A before C) between locally discovered temporal relations [8]. Aside from transitivity constraint, temporal expression normalization (e.g., "last year" is before "last month") are also used to discover implicit time-time relation in the document to enrich temporal network between events and times in TimeBank.

Other works on temporal relation identification [5, 16, 29] have been conducted for the tasks of TempEval Challenge [23]. TempEval challenge uses training and test sets encoded in a subset of TimeML, and is divided into three subtasks: to identify temporal relations 1) between a specific event and a time expression in the same sentence, 2) between a specific event and the Document Creation Time (DCT), and 3) between the provided main events in two adjacent sentences. The event is specified per sentence by the main verb in the sentence. Some approaches use relation classification on each subtask independently by using either a pure machine learning approach [5], or a combination of rule based and machine learning [16]. Another approach attempts to solve the three subtasks jointly [29] by learning a single probabilistic model for all three tasks, incorporating formulas of temporal transitivity that should hold across tasks.

Please note that there are fundamental differences between these temporal relation identification approaches and CoTS. Firstly, an event in these approaches is usually a tensed verb, which is different from the notion of relational facts in CoTS. Moreover, unlike these previous approaches, CoTS attempts to temporally ground facts on the timeline, and not just infer temporal dependencies among events (and time). However, these approaches could be complementary to CoTS in the sense that the temporal relations inferred by these approaches could potentially be used as additional constraints in CoTS's collective inference engine.

6. CONCLUSION

Despite the benefits of temporally scoped facts in knowledge bases, temporal scoping of facts is a research area that has been largely unexplored, with the exception of a few recent proposals [27, 26]. In this paper we propose CoTS (Coupled Temporal Scoping), a novel way of temporally scoping facts by exploiting a variety of signals: counts of mentions of the fact in large open domain data sources such as Google Books Ngram corpus and Gigaword Corpus; and by exploiting temporal relationships between the fact and other facts from the same or different relations.

CoTS poses the task of temporal scoping as a change detection in the temporal profile of the fact. To aggregate redundant observed cues (e.g., gradients and counts from the temporal profile) for a fact, CoTS uses Integer Linear Program (ILP) to jointly infer the time scopes of multiple temporally related facts while respecting any temporal constraints among them. CoTS is a weakly supervised system in that it only needs a few labeled examples per relation to make decision on the temporal scope of facts.

There are several avenues for improving CoTS, and overcoming fact count sparseness is one of them. Some queries such as those related to the 'defenseSecretaryOf' relation return very few results in both Google Books and Gigaword datasets. Although temporal scope of sparse facts can be inferred from the temporal scope of other related facts via coupling, sparsity of counts may still hamper temporal scoping of facts that have little or no other correlated facts. To overcome this problem, we are planning to (1) derive counts from a larger number of time-stamped sources; and (2) use semi-supervised learning algorithms to gather fact counts even from non time-stamped documents.

Furthermore, in the current CoTS system, temporal constraints among facts are specified manually based on users' prior knowledge. In the future, we would like to automatically acquire such relation and fact specific constraints, for example, by learning temporal relationships among facts in a semi-supervised fashion, such as in NELL [7].

Acknowledgments

This research has been supported in part by DARPA (under contract number FA8750-09-C-0179), Google, and Fulbright. Any opinions, findings, conclusions and recommendations expressed in this paper are the authors' and do not necessarily reflect those of the sponsors. We are thankful to Lei Li (CMU) for useful discussions, and to the four anonymous reviewers for their constructive comments.

7. REFERENCES

[1] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

[2] O. Alonso, M. Gertz, and R. Baeza-Yates. On the value of temporal information in information retrieval. In *ACM SIGIR Forum*, volume 41. ACM, 2007.

[3] A. Bakalov, A. Fuxman, P.P. Talukdar, and S. Chakrabarti. Scad: Collective discovery of attribute values. In *Proceedings of WWW*, 2011.

[4] M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. *Proceedings of IJCAI*, 2007.

[5] S. Bethard and J.H. Martin. Cu-tmp: Temporal relation classification using syntactic and semantic features. In *In SemEval-2007*, 2007.

[6] B. Boguraev and R.K. Ando. Timeml-compliant text analysis for temporal reasoning. In *Proceedings of IJCAI*, 2005.

[7] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr, and T.M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of AAAI*, 2010.

[8] N. Chambers and D. Jurafsky. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP*, 2008.

[9] M.W. Chang, L. Ratinov, N. Rizzolo, and D. Roth. Learning and inference with constraints. In *Proceedings of the 23rd National Conference on Artificial intelligence*, 2008.

[10] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of WWW*, 2004.

[11] O. Gospodnetic, E. Hatcher, et al. *Lucene in action.* Manning, 2005.

[12] D. Graff, J. Kong, K. Chen, and K. Maeda. English gigaword. *Linguistic Data Consortium, Philadelphia*, 2003.

[13] X. Ling and D.S. Weld. Temporal information extraction. In *Proceedings of AAAI*, 2010.

[14] I. Mani, M. Verhagen, B. Wellner, C.M. Lee, and J. Pustejovsky. Machine learning of temporal relations. In *Proceedings of the ACL*, 2006.

[15] J.B. Michel, Y.K. Shen, A.P. Aiden, A. Veres, M.K. Gray, J.P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, et al. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014), 2011.

[16] G. Puscasu. Wvali: Temporal relation identification by syntactico-semantic analysis. In *Proceedings of the 4th International Workshop on SemEval*, 2007.

[17] J. Pustejovsky, J. Castano, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, G. Katz, and D. Radev. Timeml: Robust specification of event and temporal expressions in text. In *Fifth International Workshop on Computational Semantics*, 2003.

[18] J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, et al. The timebank corpus. In *Corpus Linguistics*, 2003.

[19] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1), 2006.

[20] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks.

[21] F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of WWW*, 2007.

[22] P. Talukdar and K. Crammer. New regularized algorithms for transductive learning. In *Proceedings of ECML*, 2009.

[23] M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, 2007.

[24] M. Verhagen, I. Mani, R. Sauri, R. Knippen, S.B. Jang, J. Littman, A. Rumshisky, J. Phillips, and J. Pustejovsky. Automating temporal annotation with tarsqi. In *Proceedings of the ACL Session on Interactive poster and demonstration sessions*, 2005.

[25] Y. Wang, M. Yahya, and M. Theobald. Time-aware reasoning in uncertain knowledge bases. In *MUD Workshop*, 2010.

[26] Y. Wang, B. Yang, L. Qu, M. Spaniol, and G. Weikum. Harvesting facts from textual web sources by constrained label propagation. In *Proceedings of CIKM*, 2011.

[27] Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia. In *Proceedings of the 13th International Conference on Extending Database Technology*, 2010.

[28] G. Weikum, S. Bedathur, and R. Schenkel. Temporal knowledge for timely intelligence. *Enabling Real-Time Business Intelligence*, 2011.

[29] K. Yoshikawa, S. Riedel, M. Asahara, and Y. Matsumoto. Jointly identifying temporal relations with markov logic. In *Proceedings of ACL*, 2009.