Experiments in Graph-based Semi-Supervised Learning Methods for Class-Instance Acquisition

Partha Pratim Talukdar*
Search Labs, Microsoft Research
Mountain View, CA 94043
partha@talukdar.net

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

Abstract

Graph-based semi-supervised learning (SSL) algorithms have been successfully used to extract class-instance pairs from large unstructured and structured text collections. However, a careful comparison of different graph-based SSL algorithms on that task has been lacking. We compare three graph-based SSL algorithms for class-instance acquisition on a variety of graphs constructed from different domains. We find that the recently proposed MAD algorithm is the most effective. We also show that class-instance extraction can be significantly improved by adding semantic information in the form of instance-attribute edges derived from an independently developed knowledge base. All of our code and data will be made publicly available to encourage reproducible research in this area.

1 Introduction

Traditionally, named-entity recognition (NER) has focused on a small number of broad classes such as person, location, organization. However, those classes are too coarse to support important applications such as sense disambiguation, semantic matching, and textual inference in Web search. For those tasks, we need a much larger inventory of specific classes and accurate classification of terms into those classes. While supervised learning methods perform well for traditional NER, they are impractical for fine-grained classification because sufficient labeled data to train classifiers for all the classes is unavailable and would be very expensive to obtain.

To overcome these difficulties, seed-based information extraction methods have been developed over the years (Hearst, 1992; Riloff and Jones, 1999; Etzioni et al., 2005; Talukdar et al., 2006; Van Durme and Paşca, 2008). Starting with a few seed instances for some classes, these methods, through analysis of unstructured text, extract new instances of the same class. This line of work has evolved to incorporate ideas from graph-based semi-supervised learning in extraction from semi-structured text (Wang and Cohen, 2007), and in combining extractions from free text and from structured sources (Talukdar et al., 2008). The benefits of combining multiple sources have also been demonstrated recently (Pennacchiotti and Pantel, 2009).

We make the following contributions:

- Even though graph-based SSL algorithms have achieved early success in class-instance acquisition, there is no study comparing different graph-based SSL methods on this task. We address this gap with a series of experiments comparing three graph-based SSL algorithms (Section 2) on graphs constructed from several sources (Metaweb Technologies, 2009; Banko et al., 2007).
- We investigate whether semantic information in the form of instance-attribute edges derived from an independent knowledge base (Suchanek et al., 2007) can improve class-instance acquisition. The intuition behind this is that instances that share attributes are more likely to belong to the same class. We demonstrate that instance-attribute edges significantly improve the accuracy of class-instance extraction. In addition, useful class-attribute relationships are learned as a by-product of this process.
- In contrast to previous studies involving pro-

* Research carried out while at the University of Pennsylvania, Philadelphia, PA, USA.

prietary datasets (Van Durme and Paşca, 2008; Talukdar et al., 2008; Pennacchiotti and Pantel, 2009), all of our experiments use publicly available datasets and we plan to release our code[1].

In Section 2, we review three graph-based SSL algorithms that are compared for the class-instance acquisition task in Section 3. In Section 3.6, we show how additional instance-attribute based semantic constraints can be used to improve class-instance acquisition performance. We summarize the results and outline future work in Section 4.

2 Graph-based SSL

We now review the three graph-based SSL algorithms for class inference over graphs that we have evaluated.

2.1 Notation

All the algorithms compute a soft assignment of labels to the nodes of a graph $G = (V, E, W)$, where V is the set of nodes with $|V| = n$, E is the set of edges, and W is an edge weight matrix. Out of the $n = n_l + n_u$ nodes in G, n_l nodes are labeled, while the remaining n_u nodes are unlabeled. If edge $(u, v) \notin E$, $W_{uv} = 0$. The (unnormalized) Laplacian, L, of G is given by $L = D - W$, where D is an $n \times n$ diagonal degree matrix with $D_{uu} = \sum_v W_{uv}$. Let S be an $n \times n$ diagonal matrix with $S_{uu} = 1$ iff node $u \in V$ is labeled. That is, S identifies the labeled nodes in the graph. C is the set of labels, with $|C| = m$ representing the total number of labels. Y is the $n \times m$ matrix storing training label information, if any. $\hat{Y}$ is an $n \times m$ matrix of soft label assignments, with $\hat{Y}_{vl}$ representing the score of label l on node v. A graph-based SSL computes $\hat{Y}$ from $\{G, SY\}$.

2.2 Label Propagation (LP-ZGL)

The label propagation method presented by Zhu et al. (2003), which we shall refer to as LP-ZGL in this paper, is one of the first graph-based SSL methods. The objective minimized by LP-ZGL is:

$$\min_{\hat{Y}} \sum_{l \in C} \hat{Y}_l^\top L \hat{Y}_l, \text{ s.t. } SY_l = S\hat{Y}_l \quad (1)$$

where $\hat{Y}_l$ of size $n \times 1$ is the l^{th} column of $\hat{Y}$. The constraint $SY = S\hat{Y}$ makes sure that the supervised labels are not changed during inference. The above objective can be rewritten as:

$$\sum_{l \in C} \hat{Y}_l^\top L \hat{Y}_l = \sum_{u,v \in V, l \in C} W_{uv} (\hat{Y}_{ul} - \hat{Y}_{vl})^2$$

From this, we observe that LP-ZGL penalizes any label assignment where two nodes connected by a highly weighted edge are assigned different labels. In other words, LP-ZGL prefers *smooth* labelings over the graph. This property is also shared by the two algorithms we shall review next. LP-ZGL has been the basis for much subsequent work in the graph-based SSL area, and is still one of the most effective graph-based SSL algorithms.

2.3 Adsorption

Adsorption (Baluja et al., 2008) is a graph-based SSL algorithm which has been used for open-domain class-instance acquisition (Talukdar et al., 2008). Adsorption is an iterative algorithm, where label estimates on node v in the $(t+1)^{\text{th}}$ iteration are updated using estimates from the t^{th} iteration:

$$\hat{Y}_v^{(t+1)} \leftarrow p_v^{inj} \times Y_v + p_v^{cont} \times B_v^{(t)} + p_v^{abnd} \times \mathbf{r} \quad (2)$$

where,

$$B_v^{(t)} = \sum_u \frac{W_{uv}}{\sum_{u'} W_{u'v}} \hat{Y}_u^{(t)}$$

In (2), $p_v^{inj}, p_v^{cont}, \text{and } p_v^{abnd}$ are three probabilities defined on each node $v \in V$ by Adsorption; and $\mathbf{r}$ is a vector used by Adsorption to express label uncertainty at a node. On each node v, the three probabilities sum to one, i.e., $p_v^{inj} + p_v^{cont} + p_v^{abnd} = 1$, and they are based on the random-walk interpretation of the Adsorption algorithm (Talukdar et al., 2008). The main idea of Adsorption is to control label propagation more tightly by limiting the amount of information that passes through a node. For instance, Adsorption can reduce the importance of a high-degree node v during the label inference process by increasing p_v^{abnd} on that node. For more details on these, please refer to Section 2 of (Talukdar and Crammer, 2009). In contrast to LP-ZGL, Adsorption allows labels on labeled (seed) nodes to change, which is desirable in case of noisy input labels.

[1] http://www.talukdar.net/datasets/class_inst/

2.4 Modified Adsorption (MAD)

Talukdar and Crammer (2009) introduced a modification of Adsorption called MAD, which shares Adsorption's desirable properties but can be expressed as an unconstrained optimization problem:

$$\min_{\hat{Y}} \quad \sum_{l \in C} \Big[\mu_1 \left(Y_l - \hat{Y}_l \right)^\top S \left(Y_l - \hat{Y}_l \right) + \mu_2 \hat{Y}_l^\top L' \hat{Y}_l + \mu_3 \left\| \hat{Y}_l - R_l \right\|^2 \Big] \quad (3)$$

where μ_1, μ_2, and μ_3 are hyperparameters; L' is the Laplacian of an undirected graph derived from G, but with revised edge weights; and R is an $n \times m$ matrix of per-node label prior, if any, with R_l representing the l^{th} column of R. As in Adsorption, MAD allows labels on seed nodes to change. In case of MAD, the three random-walk probabilities, p_v^{inj}, p_v^{cont}, and p_v^{abnd}, defined by Adsorption on each node are folded inside the matrices S, L', and R, respectively. The optimization problem in (3) can be solved with an efficient iterative algorithm described in detail by Talukdar and Crammer (2009).

These three algorithms are all easily parallelizable in a MapReduce framework (Talukdar et al., 2008; Rao and Yarowsky, 2009), which makes them suitable for SSL on large datasets. Additionally, all three algorithms have similar space and time complexity.

3 Experiments

We now compare the experimental performance of the three graph-based SSL algorithms reviewed in the previous section, using graphs constructed from a variety of sources described below. Following previous work (Talukdar et al., 2008), we use Mean Reciprocal Rank (MRR) as the evaluation metric in all experiments:

$$\text{MRR} = \frac{1}{|Q|} \sum_{v \in Q} \frac{1}{r_v} \quad (4)$$

where $Q \subseteq V$ is the set of test nodes, and r_v is the rank of the gold label among the labels assigned to node v. Higher MRR reflects better performance. We used iterative implementations of the graph-based SSL algorithms, and the number of iterations was treated as a hyperparameter which was tuned, along with other hyperparameters, on separate held-out sets, as detailed in a longer version of this paper. Statistics of the graphs used during experiments in this section are presented in Table 1.

3.1 Freebase-1 Graph with Pantel Classes

Table ID: people-person

Name	Place of Birth	Gender
...	...	...
Isaac Newton	Lincolnshire	Male
Bob Dylan	Duluth	Male
Johnny Cash	Kingsland	Male
...	...	...

Table ID: film-music_contributor

Name	Film_Music_Credits
...	...
Bob Dylan	No Direction Home
...	...

Figure 1: Examples of two tables from Freebase, one table is from the *people* domain while the other is from the *film* domain.

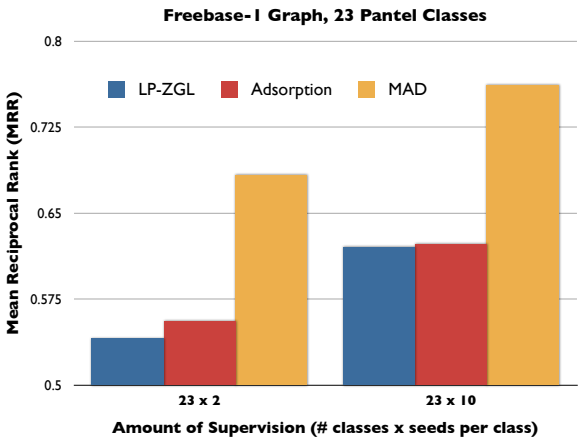


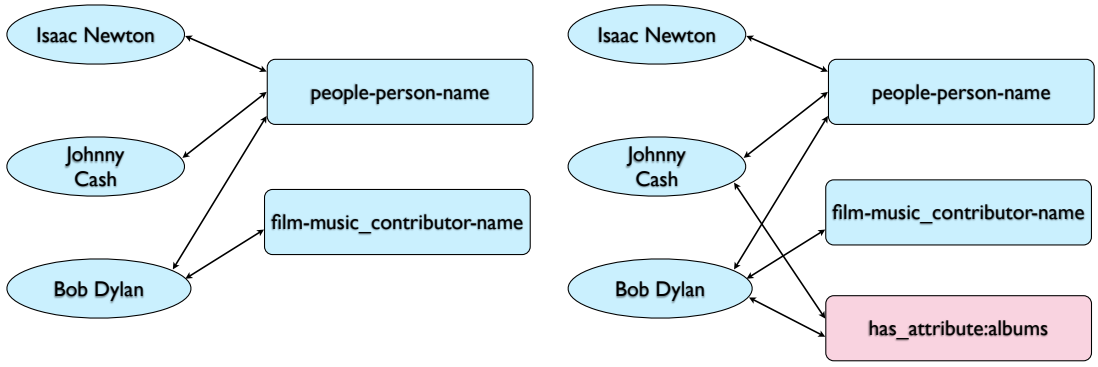
Figure 3: Comparison of three graph transduction methods on a graph constructed from the Freebase dataset (see Section 3.1), with 23 classes. All results are averaged over 4 random trials. In each group, MAD is the rightmost bar.

Freebase (Metaweb Technologies, 2009)[2] is a large collaborative knowledge base. The knowledge base harvests information from many open data sets (for instance Wikipedia and MusicBrainz), as well as from user contributions. For our current purposes, we can think of the Freebase

[2] http://www.freebase.com/

Graph	Vertices	Edges	Avg. Deg.	Min. Deg.	Max. Deg.
Freebase-1 (Section 3.1)	32970	957076	29.03	1	13222
Freebase-2 (Section 3.2)	301638	2310002	7.66	1	137553
TextRunner (Section 3.3)	175818	529557	3.01	1	2738
YAGO (Section 3.6)	142704	777906	5.45	0	74389
TextRunner + YAGO (Section 3.6)	237967	1307463	5.49	1	74389

Table 1: Statistics of various graphs used in experiments in Section 3. Some of the test instances in the YAGO graph, added for fair comparison with the TextRunner graph in Section 3.6, had no attributes in YAGO KB, and hence these instance nodes had degree 0 in the YAGO graph.



(a) (b)

Figure 2: (a) Example of a section of the graph constructed from the two tables in Figure 1. Rectangular nodes are properties, oval nodes are entities or cell values. (b) The graph in part (a) augmented with an attribute node, *has_attribue:albums*, along with the edges incident on it. This results is additional constraints for the nodes *Johnny Cash* and *Bob Dylan* to have similar labels (see Section 3.6).

dataset as a collection of relational tables, where each table is assigned a unique ID. A table consists of one or more *properties* (column names) and their corresponding *cell values* (column entries). Examples of two Freebase tables are shown in Figure 1. In this figure, *Gender* is a property in the table *people-person*, and *Male* is a corresponding cell value. We use the following process to convert the Freebase data tables into a single graph:

- Create a node for each unique cell value
- Create a node for each unique property name, where unique property name is obtained by prefixing the unique table ID to the property name. For example, in Figure 1, *people-person-gender* is a unique property name.
- Add an edge of weight 1.0 from cell-value node v to unique property node p, iff value v is present in the column corresponding to property p. Similarly, add an edge in the reverse direction.

By applying this graph construction process on the first column of the two tables in Figure 1, we end up with the graph shown in Figure 2 (a). We note that even though the resulting graph consists of edges connecting nodes of different types: cell value nodes to property nodes; the graph-based SSL methods (Section 2) can still be applied on such graphs as a cell value node and a property node connected by an edge should be assigned same or similar class labels. In other words, the label smoothness assumption (see Section 2.2) holds on such graphs.

We applied the same graph construction process on a subset of the Freebase dataset consisting of topics from 18 randomly selected domains: *astronomy, automotive, biology, book, business,*

chemistry, comic_books, computer, film, food, geography, location, people, religion, spaceflight, tennis, travel, and wine. The topics in this subset were further filtered so that only cell-value nodes with frequency 10 or more were retained. We call the resulting graph Freebase-1 (see Table 1).

Pantel et al. (2009) have made available a set of gold class-instance pairs derived from Wikipedia, which is downloadable from `http://ow.ly/13B57`. From this set, we selected all classes which had more than 10 instances overlapping with the Freebase graph constructed above. This resulted in 23 classes, which along with their overlapping instances were used as the gold standard set for the experiments in this section.

Experimental results with 2 and 10 seeds (labeled nodes) per class are shown in Figure 3. From the figure, we see that that LP-ZGL and Adsorption performed comparably on this dataset, with MAD significantly outperforming both methods.

3.2 Freebase-2 Graph with WordNet Classes

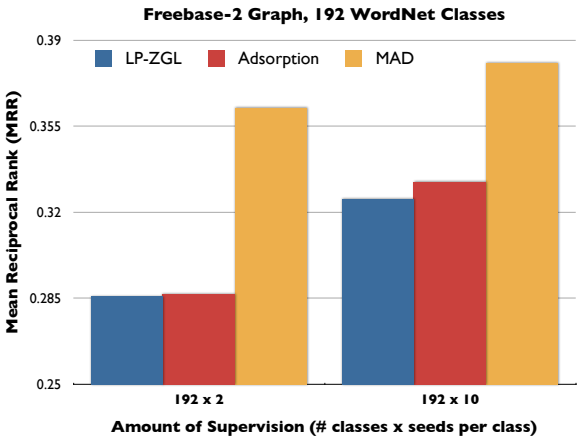


Figure 4: Comparison of graph transduction methods on a graph constructed from the Freebase dataset (see Section 3.2). All results are averaged over 10 random trials. In each group, MAD is the rightmost bar.

To evaluate how the algorithms scale up, we construct a larger graph from the same 18 domains as in Section 3.1, and using the same graph construction process. We shall call the resulting graph Freebase-2 (see Table 1). In order to scale up the number of classes, we selected all Wordnet (WN) classes, available in the YAGO KB (Suchanek et al., 2007), that had more than 100 instances overlapping with the larger Freebase graph constructed above. This resulted in 192 WN classes which we use for the experiments in this section. The reason behind imposing such frequency constraints during class selection is to make sure that each class is left with a sufficient number of instances during testing.

Experimental results comparing LP-ZGL, Adsorption, and MAD with 2 and 10 seeds per class are shown in Figure 4. A total of 292k test nodes were used for testing in the 10 seeds per class condition, showing that these methods can be applied to large datasets. Once again, we observe MAD outperforming both LP-ZGL and Adsorption. It is interesting to note that MAD with 2 seeds per class outperforms LP-ZGL and adsorption even with 10 seeds per class.

3.3 TextRunner Graph with WordNet Classes

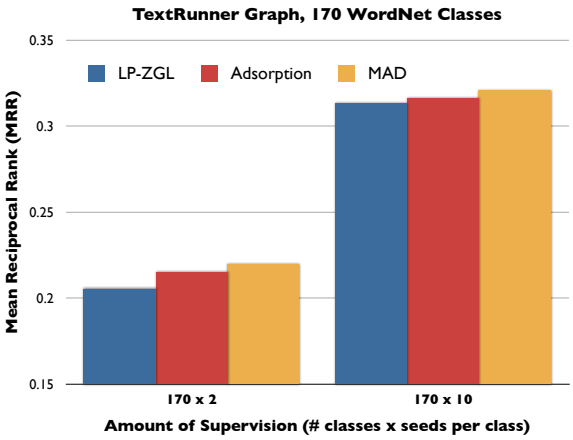


Figure 5: Comparison of graph transduction methods on a graph constructed from the hypernym tuples extracted by the TextRunner system (Banko et al., 2007) (see Section 3.3). All results are averaged over 10 random trials. In each group, MAD is the rightmost bar.

In contrast to graph construction from structured tables as in Sections 3.1, 3.2, in this section we use hypernym tuples extracted by TextRunner (Banko et al., 2007), an open domain IE system, to construct the graph. Example of a hypernym tuple extracted by TextRunner is *(http, protocol, 0.92)*, where 0.92 is the extraction confidence. To convert such a tuple into a graph, we create a node for the instance (*http*) and a node for the class (*protocol*), and then connect the nodes with two

directed edges in both directions, with the extraction confidence (*0.92*) as edge weights. The graph created with this process from TextRunner output is called the TextRunner Graph (see Table 1). As in Section 3.2, we use WordNet class-instance pairs as the gold set. In this case, we considered all WordNet classes, once again from YAGO KB (Suchanek et al., 2007), which had more than 50 instances overlapping with the constructed graph. This resulted in 170 WordNet classes being used for the experiments in this section.

Experimental results with 2 and 10 seeds per class are shown in Figure 5. The three methods are comparable in this setting, with MAD achieving the highest overall MRR.

3.4 Discussion

If we correlate the graph statistics in Table 1 with the results of sections 3.1, 3.2, and 3.3, we see that MAD is most effective for graphs with high average degree, that is, graphs where nodes tend to connect to many other nodes. For instance, the Freebase-1 graph has a high average degree of 29.03, with a corresponding large advantage for MAD over the other methods. Even though this might seem mysterious at first, it becomes clearer if we look at the objectives minimized by different algorithms. We find that the objective minimized by LP-ZGL (Equation 1) is *under-regularized*, i.e., its model parameters ($\hat{Y}$) are not constrained enough, compared to MAD (Equation 3, specifically the third term), resulting in overfitting in case of highly connected graphs. In contrast, MAD is able to avoid such overfitting because of its minimization of a well regularized objective (Equation 3). Based on this, we suggest that average degree, an easily computable structural property of the graph, may be a useful indicator in choosing which graph-based SSL algorithm should be applied on a given graph.

Unlike MAD, Adsorption does not optimize any well defined objective (Talukdar and Crammer, 2009), and hence any analysis along the lines described above is not possible. The heuristic choices made in Adsorption may have lead to its sub-optimal performance compared to MAD; we leave it as a topic for future investigation.

3.5 Effect of Per-Node Class Sparsity

For all the experiments in Sections 3.1, 3.2, and 3.6, each node was allowed to have a maximum of 15 classes during inference. After each update

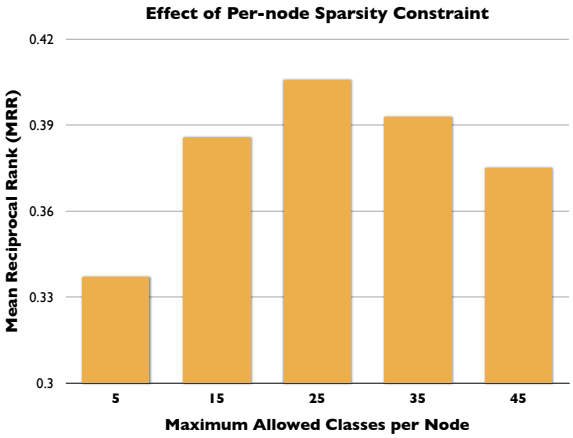


Figure 6: Effect of per node class sparsity (maximum number of classes allowed per node) during MAD inference in the experimental setting of Figure 4 (one random split).

on a node, all classes except for the top scoring 15 classes were discarded. Without such sparsity constraints, a node in a connected graph will end up acquiring all the labels injected into the graph. This is undesirable for two reasons: (1) for experiments involving a large numbers of classes (as in the previous section and in the general case of open domain IE), this increases the space requirement and also slows down inference; (2) a particular node is unlikely to belong to a large number of classes. In order to estimate the effect of such sparsity constraints, we varied the number of classes allowed per node from 5 to 45 on the graph and experimental setup of Figure 4, with 10 seeds per class. The results for MAD inference over the development split are shown in Figure 6. We observe that performance can vary significantly as the maximum number of classes allowed per node is changed, with the performance peaking at 25. This suggests that sparsity constraints during graph based SSL may have a crucial role to play, a question that needs further investigation.

3.6 TextRunner Graph with additional Semantic Constraints from YAGO

Recently, the problem of instance-attribute extraction has started to receive attention (Probst et al., 2007; Bellare et al., 2007; Pasca and Durme, 2007). An example of an instance-attribute pair is (*Bob Dylan*, *albums*). Given a set of seed instance-attribute pairs, these methods attempt to extract more instance-attribute pairs automatically

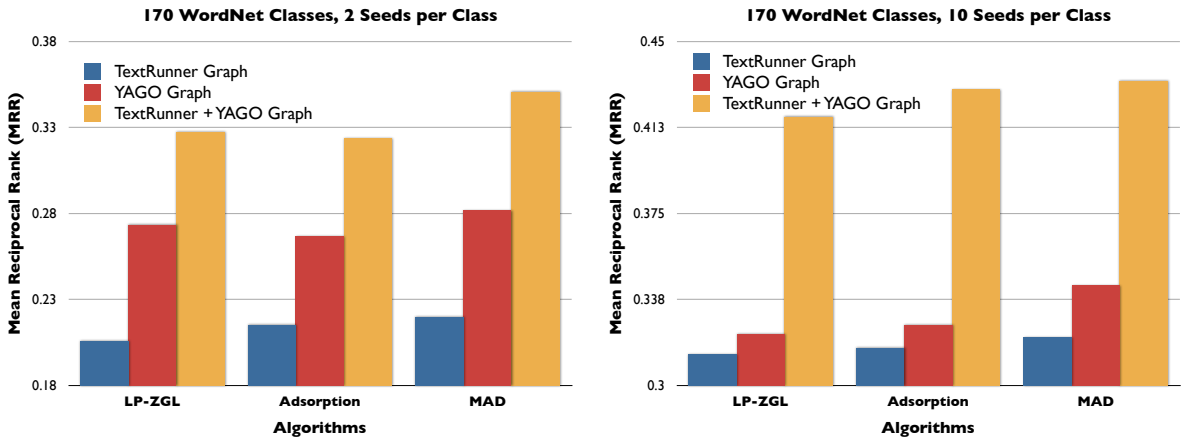


Figure 7: Comparison of class-instance acquisition performance on the three different graphs described in Section 3.6. All results are averaged over 10 random trials. Addition of YAGO attributes to the TextRunner graph significantly improves performance.

YAGO Attribute	**Top-2 WordNet Classes Assigned by MAD** (example instances for each class are shown in brackets)
has_currency	**wordnet_country_108544813 (Burma, Afghanistan)** wordnet_region_108630039 (Aosta Valley, Southern Flinders Ranges)
works_at	**wordnet_scientist_110560637 (Aage Niels Bohr, Adi Shamir)** wordnet_person_100007846 (Catherine Cornelius, Jamie White)
has_capital	**wordnet_state_108654360 (Agusan del Norte, Bali)** wordnet_region_108630039 (Aosta Valley, Southern Flinders Ranges)
born_in	**wordnet_boxer_109870208 (George Chuvalo, Fernando Montiel)** wordnet_chancellor_109906986 (Godon Brown, Bill Bryson)
has_isbn	**wordnet_book_106410904 (Past Imperfect, Berlin Diary)** wordnet_magazine_106595351 (Railway Age, Investors Chronicle)

Table 2: Top 2 (out of 170) WordNet classes assigned by MAD on 5 randomly chosen YAGO attribute nodes (out of 80) in the TextRunner + YAGO graph used in Figure 7 (see Section 3.6), with 10 seeds per class used. A few example instances of each WordNet class is shown within brackets. Top ranked class for each attribute is shown in bold.

from various sources. In this section, we explore whether class-instance assignment can be improved by incorporating new semantic constraints derived from (instance, attribute) pairs. In particular, we experiment with the following type of constraint: two instances with a common attribute are likely to belong to the same class. For example, in Figure 2 (b), instances *Johnny Cash* and *Bob Dylan* are more likely to belong to the same class as they have a common attribute, *albums*. Because of the *smooth* labeling bias of graph-based SSL methods (see Section 2.2), such constraints are naturally captured by the methods reviewed in Section 2. All that is necessary is the introduction of bidirectional (instance, attribute) edges to the graph, as shown in Figure 2 (b).

In Figure 7, we compare class-instance acquisition performance of the three graph-based SSL methods (Section 2) on the following three graphs (also see Table 1):

TextRunner Graph: Graph constructed from the hypernym tuples extracted by TextRunner, as in Figure 5 (Section 3.3), with 175k vertices and 529k edges.

YAGO Graph: Graph constructed from the (instance, attribute) pairs obtained from the YAGO KB (Suchanek et al., 2007), with 142k nodes and 777k edges.

TextRunner + YAGO Graph: Union of the

two graphs above, with 237k nodes and 1.3m edges.

In all experimental conditions with 2 and 10 seeds per class in Figure 7, we observe that the three methods consistently achieved the best performance on the TextRunner + YAGO graph. This suggests that addition of attribute based semantic constraints from YAGO to the TextRunner graph results in a better connected graph which in turn results in better inference by the graph-based SSL algorithms, compared to using either of the sources, i.e., TextRunner output or YAGO attributes, in isolation. This further illustrates the advantage of aggregating information across sources (Talukdar et al., 2008; Pennacchiotti and Pantel, 2009). However, we are the first, to the best of our knowledge, to demonstrate the effectiveness of attributes in class-instance acquisition. We note that this work is similar in spirit to the recent work by Carlson et al. (2010) which also demonstrates the benefits of additional constraints in SSL.

Because of the label propagation behavior, graph-based SSL algorithms assign classes to all nodes reachable in the graph from at least one of the labeled instance nodes. This allows us to check the classes assigned to nodes corresponding to YAGO attributes in the TextRunner + YAGO graph, as shown in Table 2. Even though the experiments were designed for class-instance acquisition, it is encouraging to see that the graph-based SSL algorithm (MAD in Table 2) is able to learn class-attribute relationships, an important by-product that has been the focus of recent studies (Reisinger and Pasca, 2009). For example, the algorithm is able to learn that *works_at* is an attribute of the WordNet class *wordnet_scientist_110560637*, and thereby its instances (e.g. *Aage Niels Bohr, Adi Shamir*).

4 Conclusion

We have started a systematic experimental comparison of graph-based SSL algorithms for class-instance acquisition on a variety of graphs constructed from different domains. We found that MAD, a recently proposed graph-based SSL algorithm, is consistently the most effective across the various experimental conditions. We also showed that class-instance acquisition performance can be significantly improved by incorporating additional semantic constraints in the class-instance acquisition process, which for the experiments in this paper were derived from instance-attribute pairs available in an independently developed knowledge base. All the data used in these experiments was drawn from publicly available datasets and we plan to release our code[3] to foster reproducible research in this area. Topics for future work include the incorporation of other kinds of semantic constraint for improved class-instance acquisition, further investigation into per-node sparsity constraints in graph-based SSL, and moving beyond bipartite graph constructions.

Acknowledgments

We thank William Cohen for valuable discussions, and Jennifer Gillenwater, Alex Kulesza, and Gregory Malecha for detailed comments on a draft of this paper. We are also very grateful to the authors of (Banko et al., 2007), Oren Etzioni and Stephen Soderland in particular, for providing TextRunner output. This work was supported in part by NSF IIS-0447972 and DARPA HRO1107-1-0029.

References

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. *Proceedings of WWW-2008*.

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

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*.

A. Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka Jr, and T.M. Mitchell. 2010. Coupled Semi-Supervised Learning for Information Extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM)*, volume 2, page 110.

O. Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web - an experimental study. *Artificial Intelligence Journal*.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Fourteenth International*

[3] http://www.talukdar.net/datasets/class_inst/

Conference on Computational Linguistics, Nantes, France.

Metaweb Technologies. 2009. Freebase data dumps. *http://download.freebase.com/datadumps/*.

P. Pantel, E. Crestan, A. Borkovsky, A.M. Popescu, and V. Vyas. 2009. Web-scale distributional similarity and entity set expansion. *Proceedings of EMNLP-09, Singapore*.

M. Pasca and Benjamin Van Durme. 2007. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI-07. Ferbruary, 2007*.

M. Pennacchiotti and P. Pantel. 2009. Entity Extraction via Ensemble Semantics. *Proceedings of EMNLP-09, Singapore*.

K. Probst, R. Ghani, M. Krema, A. Fano, and Y. Liu. 2007. Semi-supervised learning of attribute-value pairs from product descriptions. In *IJCAI-07, Ferbruary, 2007*.

D. Rao and D. Yarowsky. 2009. Ranking and Semi-supervised Classification on Large Scale Graphs Using Map-Reduce. *TextGraphs*.

J. Reisinger and M. Pasca. 2009. Bootstrapped extraction of class attributes. In *Proceedings of the 18th international conference on World wide web*, pages 1235–1236. ACM.

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.

F.M. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, page 706. ACM.

P. P. Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *ECML-PKDD*.

P. P. Talukdar, T. Brants, F. Pereira, and M. Liberman. 2006. A context pattern induction method for named entity extraction. In *Tenth Conference on Computational Natural Language Learning*, page 141.

P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira. 2008. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 581–589.

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*.