Keepin' It Real: Semi-Supervised Learning with Realistic Tuning

Andrew B. Goldberg
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706, USA
goldberg@cs.wisc.edu

Xiaojin Zhu
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706, USA
jerryzhu@cs.wisc.edu

Abstract

We address two critical issues involved in applying semi-supervised learning (SSL) to a real-world task: parameter tuning and choosing which (if any) SSL algorithm is best suited for the task at hand. To gain a better understanding of these issues, we carry out a medium-scale empirical study comparing supervised learning (SL) to two popular SSL algorithms on eight natural language processing tasks under three performance metrics. We simulate how a practitioner would go about tackling a new problem, including parameter tuning using cross validation (CV). We show that, under such realistic conditions, each of the SSL algorithms can be worse than SL on some datasets. However, we also show that CV can select SL/SSL to achieve "agnostic SSL," whose performance is almost always no worse than SL. While CV is often dismissed as unreliable for SSL due to the small amount of labeled data, we show that it is in fact effective for accuracy even when the labeled dataset size is as small as 10.

1 Introduction

Imagine you are a real-world practitioner working on a machine learning problem in natural language processing. If you have unlabeled data, should you use semi-supervised learning (SSL)? Which SSL algorithm should you use? How should you set its parameters? Or could it actually hurt performance, in which case you might be better off with supervised learning (SL)?

A large number of SSL algorithms have been developed in recent years that allow one to improve performance with unlabeled data, in tasks such as text classification, sequence labeling, and parsing (Zhu, 2005; Chapelle et al., 2006; Brefeld and Scheffer, 2006). However, many of them are tested on "SSL-friendly" datasets, such as "two moons," USPS, and MNIST. Furthermore, the algorithms' parameters are often chosen based on test set performance or manually set based on heuristics and researcher experience. These issues create practical concerns for deploying SSL in the real world.

We note that (Chapelle et al., 2006)'s benchmark chapter explores these issues to some extent by comparing several SSL methods on several real and artificial datasets. The authors reach the conclusions that parameter tuning is difficult with little labeled data and that no method is universally superior. We reexamine these issues in the context of NLP tasks and offer a simple attempt at overcoming these roadblocks to practical application of SSL.

The contributions of this paper include:

- We present a medium-scale empirical study comparing SL to two popular SSL algorithms on eight less-familiar tasks using three performance metrics. Importantly, we *tune parameters realistically* based on cross validation (CV), as a practitioner would do in reality.
- We show that, under such realistic conditions, each of the SSL algorithms can be worse than SL on some datasets.
- However, this can be prevented. We show that CV can be used to select SL/SSL to achieve *agnostic SSL*, whose performance is almost always no worse than SL. Traditionally, CV is

often dismissed as unreliable for SSL because of the small labeled dataset size. But we show that CV is effective when using accuracy as an optimization criterion, even when the labeled dataset size is as small as 10.

- We show the power of cloud computing: we were able to complete roughly 3 months worth of experiments in less than a week.

2 SSL with Realistic Tuning

Given a particular labeled and unlabeled dataset, how should you set parameters for a particular SSL model? The most realistic approach for a practitioner is to use CV to tune parameters on a grid. We therefore argue that the model parameters obtained in this way truly determine how SSL will perform in practice. Algorithm 1 describes a particular instance[1] of CV in detail. We call it "RealSSL," as this is all a real user can hope to do when applying SSL (and SL too) in a realistic problem scenario.

3 Experimental Procedure

Given the RealSSL procedure in Algorithm 1, we designed an experimental setup to simulate different settings that a real-world practitioner might face when given a new task and a set of algorithms to choose from (some of which use unlabeled data). This will allow us to compare algorithms across datasets in a variety of situations. Algorithm 2 measures the performance of one algorithm on one dataset for several different l and u combinations. Specifically, we consider $l \in \{10, 100\}$ and $u \in \{100, 1000\}$. For each combination, we perform multiple trials ($T = 10$ here) using different random assignments of data to $D_{labeled}$ and $D_{unlabeled}$, to obtain confidence intervals around our performance measurements. All random selections of subsets of data are the same across different algorithms' runs, to permit paired t-tests for evaluation. Note that, when $l \neq \max(L)$ or $u \neq \max(U)$, a portion of D_{pool} is not used for training. Also, the RealSSL procedure ensures that all parameters are tuned by cross-validation without ever seeing the held-out test set D_{test}. Lastly, we stress that the same grid of algorithm-specific parameter values (discussed in Section 5) is considered for all datasets.

[1]The particular choice of 5-fold CV, the way to split labeled and unlabeled data, and the parameter grid, is important too. But we view them as secondary to the fact that we are tuning SSL by CV.

4 Datasets

Table 1 summarizes the datasets used for the comparisons. In this study we consider only binary classification tasks. Note that d is the number of dimensions, $P(y = 1)$ is the proportion of instances in the full dataset belonging to class $y = 1$, and $|D_{test}|$ refers to the size of the test set (the instances remaining after $\max(L) + \max(U) = 1100$ have been set aside for training trials).

[MacWin] is the Mac versus Windows text classification data from the 20-newsgroups dataset, preprocessed by the authors of (Sindhwani et al., 2005).

[Interest] is a binary version of the word sense disambiguation data from (Bruce and Wiebe, 1994). The task is to distinguish the sense of "interest" meaning "money paid for the use of money" from the other five senses (e.g., "readiness to give attention," "a share in a company or business"). The data comes from a corpus of part-of-speech (POS) tagged sentences containing the word "interest." Each instance is a bag-of-word/POS vector, excluding words containing the root "interest" and those that appeared in less than three sentences overall.

Datasets [aut-avn] and [real-sim] are the auto/aviation and real/simulated text classification datasets from the SRAA corpus of UseNet articles. The [ccat] and [gcat] datasets involve identifying corporate and government articles, respectively, in the RCV1 corpus. We use the versions of these datasets prepared by the authors of (Sindhwani et al., 2006).

Finally, the two WISH datasets come from (Goldberg et al., 2009) and involve discriminating between sentences that contain wishful expressions and those that do not. The instances in [WISH-politics] correspond to sentences taken from a political discussion board, while [WISH-products] is based on sentences from Amazon product reviews. The features are a combination of word and template features as described in (Goldberg et al., 2009).

```
**Input**: dataset $D_{labeled} = \{x_i, y_i\}_{i=1}^{l}$, $D_{unlabeled} = \{x_j\}_{j=1}^{u}$, $algorithm$, performance $metric$

Randomly partition $D_{labeled}$ into 5 equally-sized disjoint subsets $\{D_{l1}, D_{l2}, D_{l3}, D_{l4}, D_{l5}\}$.
Randomly partition $D_{unlabeled}$ into 5 equally-sized disjoint subsets $\{D_{u1}, D_{u2}, D_{u3}, D_{u4}, D_{u5}\}$.
Combine partitions: Let $D_{fold\ k} = D_{lk} \cup D_{uk}$ for all $k = 1, \ldots, 5$.
foreach parameter configuration in grid do
    foreach fold k do
        Train model using $algorithm$ on $\cup_{i \neq k} D_{fold\ i}$.
        Evaluate $metric$ on $D_{fold\ k}$.
    end
    Compute the average $metric$ value across the 5 folds.
end
Choose parameter configuration that optimizes average $metric$.
Train model using $algorithm$ and the chosen parameters on $D_{labeled}$ and $D_{unlabeled}$.

Output: Optimal model; Average $metric$ value achieved by optimal parameters during tuning.
```

Algorithm 1: RealSSL procedure for running an SSL (or SL, simply ignore the unlabeled data) algorithm on a specific labeled and unlabeled dataset using cross-validation to tune parameters.

```
Input: dataset $D = \{x_i, y_i\}_{i=1}^{n}$, $algorithm$, performance $metric$, set $L$, set $U$, trials $T$
Randomly divide $D$ into $D_{pool}$ (of size $\max(L) + \max(U)$) and $D_{test}$ (the rest).
foreach $l$ in $L$ do
    foreach $u$ in $U$ do
        foreach trial 1 up to $T$ do
            Randomly select $D_{labeled} = \{x_j, y_j\}_{j=l}^{l}$ and $D_{unlabeled} = \{x_k\}_{k=1}^{u}$ from $D_{pool}$.
            Run RealSSL($D_{labeled}$, $D_{unlabeled}$, $algorithm$, $metric$) to obtain model and tuning
                performance value (see Algorithm 1).
            Use model to classify $D_{unlabeled}$ and record transductive $metric$ value.
            Use model to classify $D_{test}$ and record test $metric$ value.
        end
    end
end
Output: Tuning, transductive, and test performance for $T$ runs of $algorithm$ using all $l$ and $u$
        combinations.
```

Algorithm 2: Experimental procedure used for all comparisons.

Name	d	$P(y=1)$	$\lvert D_{test}\rvert$
[MacWin]	7511	0.51	846
[Interest]	2687	0.53	1268
[aut-avn]	20707	0.65	70075
[real-sim]	20958	0.31	71209
[ccat]	47236	0.47	22019
[gcat]	47236	0.30	22019
[WISH-politics]	13610	0.34	4999
[WISH-products]	4823	0.12	129

Table 1: Datasets used in benchmark comparison. See text for details.

5 Algorithms

We consider only linear classifiers for this study, since they tend to work well for text problems. In future work, we plan to explore a range of kernels and other non-linear classifiers.

As a baseline supervised learning method, we use a support vector machine (SVM), as implemented by SVM^{light} (Joachims, 1999). This baseline simply ignores all the unlabeled data ($\mathbf{x}_{l+1}, \ldots, \mathbf{x}_n$). Recall this solves the following regularized risk minimization problem

$$\min_f \frac{1}{2}||f||_2^2 + C\sum_{i=1}^{l} \max(0, 1 - y_i f(\mathbf{x}_i)), \quad (1)$$

where $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, and C is a parameter controlling the trade-off between training errors and model complexity. Using the procedure outlined above, we tune C over a grid of values $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$.

We consider two popular SSL algorithms, which make different assumptions about the link between the marginal data distribution $\mathcal{P}_x$ and the conditional label distribution $\mathcal{P}_{y|x}$. If the assumption does not hold in a particular dataset, the SSL algorithm could use the unlabeled data "incorrectly," and perform worse than SL.

The first SSL algorithm we use is a semi-supervised support vector machine (S3VM), which makes the cluster assumption: the classes are well-separated clusters of data, such that the decision boundary falls into a low density region in the feature space. While many implementations exist, we chose the deterministic annealing (DA) algorithm implemented in the SVMlin package (Sindhwani et al., 2006; Sindhwani and Keerthi, 2007). This DA algorithm often achieved the best accuracy across several datasets in the empirical comparison in (Sindhwani and Keerthi, 2007), despite being slower than the multi-switch algorithm presented in the same paper. Note that the transductive SVM implemented in SVM^{light} would have been prohibitively slow to carry out the range of experiments conducted here. Recall that an S3VM seeks an optimal classifier f^* that cuts through a region of low density between clusters of data. One way to view this is that it tries to find the best possible labeling of the unlabeled data such the classifier maximizes the margin on both labeled and unlabeled data points. This is achieved by solving the following non-convex minimization problem

$$\min_{f, \mathbf{y}' \in \{-1,1\}^u} \frac{\lambda}{2}||f||_2^2 + \frac{1}{l}\sum_{i=1}^{l} V(y_i f(\mathbf{x}_i) + \frac{\lambda'}{u}\sum_{j=l+1}^{n} V(y'_j f(\mathbf{x}_j)),$$

subject to a class-balance constraint. Note that V is a loss function (typically the hinge loss as in (1)), and the parameters λ, λ' control the relative importance of model complexity versus locating a low-density region within the unlabeled data. We tune both parameters in a grid of values $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$. In past studies (Sindhwani et al., 2006), λ was set to 1, and λ' was tuned over a grid containing a subset of these values.

Finally, as an example of a graph-based SSL method, we consider manifold regularization (MR) (Belkin et al., 2006), using the implementation provided on the authors' Web site.[2] This algorithm makes the manifold assumption: the labels are "smooth" with respect to a graph connecting labeled and unlabeled instances. In other words, if two instances are connected by a strong edge (e.g., they are highly similar to one another), their labels tend to be the same. Manifold regularization represents a family of methods; we specifically use the Laplacian SVM, which extends the basic SVM optimization

[2] `http://manifold.cs.uchicago.edu/manifold_regularization/software.html`

problem with a graph-based regularization term.

$$\min_f \gamma_A ||f||_2^2 + \frac{1}{l}\sum_{i=1}^{l} \max(0, 1 - y_i f(\mathbf{x}_i)) \\ + \gamma_I \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2,$$

where γ_A and γ_I are parameters that trade off ambient and intrinsic smoothness, and w_{ij} is a graph weight between instances $\mathbf{x}_i$ and $\mathbf{x}_j$. In this paper, we consider kNN graphs with $k \in \{3, 10, 20\}$. Edge weights are formed using a heat kernel $w_{ij} = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2})$, where σ is set to be the mean distance between nearest neighbors in the graph, as in (Chapelle et al., 2006). The γ parameters are each tuned over the grid $\{10^{-6}, 10^{-4}, 10^{-2}, 1, 100\}$.

Of course, many other SSL algorithms exist, some of which combine different assumptions (Chapelle and Zien, 2005; Karlen et al., 2008), and others which exploit multiple (real or artificial) views of the data (Blum and Mitchell, 1998; Sindhwani and Rosenberg, 2008). We plan to extend our study to include many more diverse SSL algorithms in the future.

6 Choosing an Algorithm for a Real-World Task

Given the choice of several algorithms, how should one choose the best one to apply to a particular learning setting? Traditionally, CV is used for model selection in supervised learning settings. However, with only a small amount of labeled data in semi-supervised settings, model selection with CV is often viewed as unreliable. We explicitly tested this hypothesis by using CV to not only choose the parameters of the model, but also choose the type of model itself. The main goal is to automatically choose between SVM, S3VM, and MR for a particular learning setting, in an attempt to ensure that the final performance is never hurt by including unlabeled data (which might be called agnostic SSL).

Given a set of algorithms (e.g., one SL, several SSL), the procedure is the following:

1. Tune the parameters of each algorithm on the labeled and unlabeled training set using Algorithm 1.[3]

2. Compare the best tuning performance (5-fold CV average) achieved by the optimal parameters for each of the algorithms.

   - If there are no ties, select the algorithm with the highest tuning performance.
   - If there is a tie, and SL is among the best, select SL.
   - If there is a tie between SSL algorithms, select one of them at random.

3. Use the selected "Best Tuning" algorithm (and the tuned parameters) to build a model on all the training data; then apply it to the test data.

Note that the procedure is conservative in that it prefers SL in the case of ties. In this paper, we use this simple "best tuning performance" heuristic.

Finally, we stress the fact that, when applying this procedure within the context of Algorithm 2, a potentially different algorithm is chosen in each of the 10 trials for a particular setting. This simulates the real-world scenario where one only has a single fixed training set of labeled and unlabeled data and must choose a single algorithm to produce a model for future predictions.

7 Performance Metrics

We compare different algorithms' performance using three metrics often used for evaluation in NLP tasks: accuracy, maxF1, and AUROC. Accuracy is simply the fraction of instances correctly classified. MaxF1 is the maximal F1 value (harmonic mean of recall and precision) achieved over the entire precision-recall curve (Cai and Hofmann, 2003). AUROC is the area under the ROC curve (Fawcett, 2004). Throughout the paper, when we discuss a result involving a particular metric, the algorithms use this metric as the criterion for parameter tuning, and we use it for the final evaluation. We are not simply evaluating a single experiment using multiple metrics—the experiments are fundamentally different and produce different learned models.

[3]We ensure each algorithm uses the same 5 partitions during the tuning step.

8 Results

We now report the results of our empirical comparison of SL and SSL on the eight NLP datasets. We first consider each dataset separately and examine how often each type of algorithm outperforms the other. We then examine cross-dataset performance.

8.1 Detailed Results

Table 2 contains all results for SVM, S3VM, and MR for all datasets and all metrics.[4] Note that within each l,u cell for a particular dataset and evaluation metric, we show the maximum value in each row (tune, transductive, or test) in boldface. Results that are not statistically significantly different using a paired t-test are also shown in boldface.

Several things are immediately obvious from Table 2. First, no algorithm is superior in all datasets or settings. In several cases, all algorithms are statistically indistinguishable. Most importantly, though, each of the SSL algorithms can be worse than SL on some datasets using some metric. We used paired t-tests to compare transductive and test performance of each SSL algorithm with SVM for a particular l,u combination and dataset (32 settings total per evaluation metric). In terms of accuracy, MR transductive performance is significantly worse than SVM in 5 settings, while MR test performance is significantly worse in 7 settings. MR is also significantly worse in 4 settings based on transductive maxF1, in 3 settings based on transductive AUROC, and 1 setting based on test AUROC. S3VM is significantly worse than SVM in 2 settings based on transductive maxF1, 2 settings based on transductive AUROC, and in 1 setting based on test AUROC. While these numbers may seem relatively low, it is important to realize that each algorithm may be worse than SSL many times on a trial-by-trial basis, which is the more realistic scenario: a practitioner has only a single dataset to work with. Results based on individual trials are discussed below shortly.

[4]Note that the results here for a particular dataset and algorithm combination may be qualitatively and quantitatively different than in previous published work, due to differences in parameter tuning, choices of parameter grids, l and u sizes, and randomization. We are not trying to replicate or raise doubt about past results: we simply intend to compare algorithms on a wide array of datasets using the standardized procedures outlined above.

We also applied our "Best Tuning" model selection procedure to automatically choose a single algorithm for each trial in each setting. We compare average SL test performance versus the average test performance of the Best Tuning selections across the 10 trials (not shown in Table 2). Comparisons based on transductive performance are similar. When the performance metric is test accuracy, the Best Tuning algorithm performs statistically significantly better than SL in 24 settings and worse in only 6 settings. In the remaining 2 settings, Best Tuning chose SL in all 10 trials, so they are equivalent. These results suggest that accuracy-based tuning is a valid method for choosing a SSL algorithm to improve accuracy on test data. To some extent, this holds for maxF1, too: the Best Tuning selections perform better than SL (on average) in 18 settings and worse in 14 settings when tuning and test evaluation is based on maxF1. However, when using AUROC as the performance metric, cross validation seems to be unreliable: Best Tuning produces a better result in only 11 out of the 32 settings.

8.2 Results Aggregated Across Datasets

We now aggregate the detailed results to better understand the relative performance of the different methods across all datasets. We perform this summary evaluation in two ways, based on test set performance (transductive performance is similar). First, we compare the SSL algorithms across all datasets based on the numbers of times each is worse than, the same as, or better than SL. For each of the 80 trials of a particular l,u,metric combination, we compare the performance of S3VM, MR, and Best Tuning to SVM. Note that each of these comparisons is akin to a real-world scenario where a practitioner would have to choose an algorithm to use. Table 3 lists tuples of the form "(#trials worse than SVM, #trials equal to SVM, #trials better than SVM)." Note that the numbers in each tuple sum to 80. The perfect SSL algorithm would have a tuple of "(0, 0, 80)," meaning that it always outperforms SL. In terms of accuracy (Table 3, top) and maxF1 (Table 3, middle), the Best Tuning method turns out to do worse than SVM less often than either S3VM or MR does (i.e., the first number in the tuples for Best Tuning is lower than the corresponding numbers for the other algorithms). At the same time, Best Tuning

		accuracy						maxF1						AUROC						
		$u = 100$			$u = 1000$			$u = 100$			$u = 1000$			$u = 100$			$u = 1000$			
Dataset	l	SVM	S3VM	MR	SVM	S3VM	MR	SVM	S3VM	MR	SVM	S3VM	MR	SVM	S3VM	MR	SVM	S3VM	MR	
[MacWin]	10	0.60	**0.72**	**0.83**	0.60	0.72	**0.86**	**0.66**	**0.67**	**0.67**	**0.66**	**0.67**	**0.67**	**0.63**	**0.69**	**0.67**	**0.63**	**0.69**	**0.69**	Tune
		0.51	0.51	**0.70**	0.51	0.50	**0.69**	0.74	**0.77**	**0.80**	**0.74**	**0.74**	**0.75**	0.72	**0.75**	**0.82**	0.72	**0.71**	**0.80**	Trans
		0.53	0.50	**0.71**	0.53	0.50	**0.68**	0.74	0.75	**0.79**	**0.74**	**0.75**	**0.74**	0.73	**0.72**	**0.83**	**0.73**	**0.71**	**0.76**	Test
	100	0.87	0.87	**0.91**	0.87	0.87	**0.90**	**0.94**	**0.95**	**0.95**	**0.94**	**0.95**	**0.95**	0.96	**0.97**	**0.97**	**0.96**	**0.96**	**0.96**	Tune
		0.89	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**	0.91	**0.93**	**0.92**	**0.91**	**0.90**	**0.90**	**0.97**	**0.97**	**0.96**	**0.97**	**0.97**	0.96	Trans
		0.89	0.89	**0.91**	**0.89**	**0.89**	**0.90**	**0.92**	**0.92**	**0.92**	**0.92**	**0.91**	**0.91**	**0.97**	**0.97**	**0.97**	**0.97**	**0.97**	0.97	Test
[Interest]	10	**0.68**	**0.75**	**0.78**	0.68	**0.75**	**0.79**	0.73	**0.77**	**0.77**	0.73	**0.78**	**0.77**	0.52	**0.66**	**0.66**	0.52	**0.68**	**0.64**	Tune
		0.52	**0.56**	**0.56**	**0.52**	**0.56**	**0.56**	**0.72**	**0.72**	**0.72**	**0.72**	**0.71**	**0.71**	**0.55**	**0.54**	**0.54**	0.55	**0.56**	**0.61**	Trans
		0.52	**0.57**	**0.57**	0.52	**0.57**	**0.58**	**0.68**	**0.69**	**0.69**	**0.68**	**0.69**	**0.69**	**0.58**	**0.56**	**0.61**	0.58	**0.58**	**0.62**	Test
	100	0.77	**0.78**	0.76	0.77	**0.78**	**0.77**	0.84	**0.85**	**0.85**	0.84	**0.85**	0.84	0.89	**0.90**	0.89	**0.89**	**0.85**	**0.84**	Tune
		0.79	**0.79**	0.71	**0.79**	**0.79**	**0.77**	**0.84**	**0.83**	0.82	**0.84**	0.81	0.81	**0.91**	**0.91**	**0.89**	**0.91**	0.79	0.87	Trans
		0.81	**0.80**	0.78	**0.81**	**0.80**	0.79	**0.82**	**0.81**	**0.81**	**0.82**	**0.81**	**0.81**	**0.90**	**0.91**	**0.89**	**0.90**	**0.81**	0.88	Test
[aut-avn]	10	**0.72**	**0.76**	**0.82**	**0.72**	**0.76**	**0.79**	**0.89**	**0.92**	**0.91**	**0.89**	**0.92**	**0.91**	**0.58**	**0.67**	**0.65**	**0.58**	**0.67**	**0.65**	Tune
		0.65	**0.63**	**0.67**	**0.65**	**0.61**	**0.69**	**0.83**	**0.83**	**0.84**	**0.83**	**0.81**	**0.82**	**0.71**	**0.67**	**0.73**	**0.71**	**0.65**	**0.72**	Trans
		0.62	**0.61**	**0.67**	**0.62**	**0.61**	**0.67**	0.80	**0.81**	**0.82**	**0.80**	**0.81**	**0.81**	**0.71**	**0.70**	**0.73**	**0.71**	**0.65**	**0.69**	Test
	100	0.75	0.82	**0.87**	0.75	0.82	**0.86**	0.94	**0.94**	**0.95**	**0.94**	**0.94**	**0.94**	0.93	**0.94**	**0.94**	0.93	**0.94**	0.93	Tune
		0.77	0.79	**0.88**	0.77	0.83	**0.87**	**0.92**	**0.92**	**0.91**	**0.92**	**0.91**	**0.90**	**0.93**	**0.93**	0.91	**0.93**	**0.94**	**0.93**	Trans
		0.77	0.82	**0.89**	0.77	0.83	**0.87**	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**	**0.95**	**0.94**	**0.95**	**0.95**	**0.95**	**0.95**	Test
[real-sim]	10	0.53	0.63	**0.82**	0.53	0.63	**0.78**	**0.65**	**0.66**	**0.66**	**0.65**	**0.66**	**0.65**	**0.77**	**0.81**	**0.81**	**0.77**	**0.81**	**0.77**	Tune
		0.64	0.63	**0.72**	**0.64**	**0.64**	**0.70**	0.57	**0.66**	**0.70**	0.57	**0.62**	**0.56**	0.65	**0.75**	**0.79**	**0.65**	**0.74**	**0.67**	Trans
		0.65	**0.66**	**0.74**	**0.65**	**0.66**	**0.68**	0.53	0.58	**0.63**	0.53	**0.59**	**0.53**	0.64	**0.73**	**0.80**	**0.64**	**0.74**	**0.66**	Test
	100	0.74	0.73	**0.86**	0.74	0.73	**0.84**	0.88	**0.90**	**0.90**	0.88	**0.91**	0.89	0.93	**0.94**	**0.94**	0.93	**0.94**	0.93	Tune
		0.78	0.76	**0.84**	0.78	0.78	**0.85**	**0.81**	**0.83**	**0.79**	**0.81**	**0.81**	**0.81**	**0.94**	**0.93**	**0.91**	**0.94**	**0.94**	0.94	Trans
		0.79	0.78	**0.85**	0.79	0.78	**0.85**	0.78	**0.79**	**0.78**	0.78	**0.79**	0.79	**0.93**	**0.93**	**0.93**	0.93	**0.94**	0.93	Test
[ccat]	10	0.54	0.60	**0.82**	0.54	0.60	**0.81**	**0.84**	**0.85**	**0.85**	**0.84**	**0.85**	**0.84**	**0.74**	**0.78**	**0.78**	**0.74**	**0.78**	**0.74**	Tune
		0.50	0.49	**0.65**	0.50	0.51	**0.67**	**0.69**	**0.69**	**0.73**	**0.69**	0.67	**0.69**	**0.60**	**0.61**	**0.71**	0.60	**0.59**	**0.72**	Trans
		0.49	0.52	**0.64**	0.49	0.52	**0.66**	0.66	0.66	**0.69**	**0.66**	**0.67**	**0.67**	**0.61**	**0.63**	**0.72**	**0.61**	**0.59**	**0.71**	Test
	100	0.80	0.80	**0.84**	0.80	0.80	**0.84**	**0.89**	**0.89**	**0.90**	**0.89**	**0.89**	**0.89**	**0.91**	**0.92**	**0.92**	**0.91**	**0.92**	**0.91**	Tune
		0.80	**0.79**	**0.80**	**0.80**	0.81	**0.83**	**0.83**	**0.85**	**0.84**	**0.83**	**0.82**	**0.82**	**0.91**	0.91	0.89	**0.91**	**0.90**	**0.91**	Trans
		0.81	**0.80**	**0.81**	**0.81**	**0.80**	**0.82**	**0.80**	**0.81**	**0.81**	**0.80**	**0.81**	**0.81**	**0.90**	0.90	**0.90**	**0.90**	**0.90**	**0.90**	Test
[gcat]	10	0.74	**0.83**	**0.82**	**0.74**	**0.79**	**0.81**	**0.44**	**0.47**	**0.46**	**0.44**	**0.47**	**0.46**	**0.69**	**0.79**	**0.75**	**0.69**	**0.79**	**0.75**	Tune
		0.69	**0.68**	**0.75**	**0.69**	**0.72**	**0.76**	0.60	**0.62**	**0.69**	**0.60**	**0.59**	**0.62**	0.71	**0.73**	**0.82**	**0.71**	**0.69**	**0.76**	Trans
		0.66	**0.67**	**0.73**	**0.66**	**0.71**	**0.74**	0.58	**0.61**	**0.66**	**0.58**	**0.60**	**0.59**	0.69	**0.69**	**0.81**	**0.69**	**0.69**	**0.75**	Test
	100	0.77	0.77	**0.90**	0.77	0.77	**0.91**	**0.92**	**0.92**	**0.93**	**0.92**	**0.92**	**0.92**	**0.97**	0.96	**0.97**	**0.97**	**0.96**	**0.96**	Tune
		0.81	0.80	**0.89**	0.81	0.81	**0.90**	**0.88**	**0.88**	0.84	**0.88**	**0.86**	**0.85**	**0.96**	**0.97**	**0.95**	**0.96**	**0.96**	**0.96**	Trans
		0.80	0.80	**0.89**	0.80	0.80	**0.90**	**0.86**	**0.86**	**0.85**	**0.86**	**0.86**	**0.86**	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**	Test
[WISH-politics]	10	**0.70**	**0.77**	**0.79**	**0.70**	**0.77**	**0.82**	**0.61**	**0.62**	**0.61**	**0.61**	**0.62**	**0.61**	**0.74**	**0.78**	**0.74**	**0.74**	**0.78**	**0.76**	Tune
		0.50	**0.56**	**0.63**	**0.50**	**0.62**	**0.56**	**0.58**	**0.58**	**0.61**	**0.58**	**0.55**	**0.53**	0.62	**0.62**	**0.69**	**0.62**	**0.62**	**0.61**	Trans
		0.52	**0.56**	**0.60**	**0.52**	**0.62**	**0.53**	0.52	**0.53**	**0.53**	0.52	**0.54**	**0.52**	**0.57**	**0.58**	**0.61**	**0.57**	**0.62**	**0.60**	Test
	100	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.74**	0.74	**0.75**	**0.76**	**0.74**	**0.75**	**0.75**	**0.79**	**0.80**	**0.80**	**0.79**	**0.80**	**0.80**	Tune
		0.73	**0.73**	**0.71**	**0.73**	**0.73**	**0.70**	**0.65**	**0.66**	**0.67**	**0.65**	**0.64**	**0.64**	**0.76**	**0.74**	**0.75**	**0.76**	**0.75**	**0.76**	Trans
		0.75	**0.75**	0.72	**0.75**	**0.75**	0.71	**0.64**	**0.63**	**0.63**	**0.64**	**0.63**	**0.64**	**0.78**	**0.76**	**0.77**	**0.78**	**0.76**	**0.77**	Test
[WISH-products]	10	**0.89**	**0.89**	0.67	**0.89**	**0.89**	0.67	0.19	**0.22**	0.16	0.19	**0.22**	0.16	**0.76**	**0.80**	**0.74**	**0.76**	**0.80**	**0.74**	Tune
		0.87	**0.87**	0.66	**0.87**	**0.87**	0.61	**0.31**	**0.29**	**0.32**	**0.31**	0.24	0.25	**0.56**	0.52	**0.58**	**0.56**	**0.54**	**0.56**	Trans
		0.90	**0.90**	0.67	**0.90**	**0.90**	0.61	0.22	0.23	**0.30**	0.22	**0.24**	**0.27**	0.50	0.53	**0.62**	0.50	**0.54**	**0.59**	Test
	100	**0.90**	**0.90**	0.82	**0.90**	**0.90**	0.81	0.49	**0.50**	**0.54**	0.49	**0.52**	**0.52**	0.73	0.73	**0.77**	**0.73**	**0.78**	**0.75**	Tune
		0.88	**0.88**	0.81	**0.88**	**0.88**	0.80	**0.34**	0.28	**0.37**	**0.34**	**0.27**	**0.30**	**0.60**	**0.55**	**0.57**	**0.60**	**0.57**	**0.61**	Trans
		0.90	**0.90**	0.79	**0.90**	**0.91**	0.76	**0.33**	0.28	**0.33**	**0.33**	**0.32**	**0.38**	**0.59**	**0.56**	**0.60**	**0.59**	**0.56**	**0.60**	Test

Table 2: Benchmark comparison results. All numbers are averages over 10 trials. Within each cell of nine numbers, the boldface indicates the maximum value in each row, as well as others in the row that are not statistically significantly different based on a paired t-test.

		$u = 100$			$u = 1000$			
Metric	l	S3VM	MR	Best Tuning	S3VM	MR	Best Tuning	
accuracy	10	(14, 27, 39)	(27, 0, 53)	(8, 31, 41)	(14, 25, 41)	(27, 0, 53)	(8, 29, 43)	Test
	100	(27, 7, 46)	(38, 0, 42)	(20, 16, 44)	(27, 6, 47)	(37, 0, 43)	(16, 19, 45)	Test

Metric	l	S3VM	MR	Best Tuning	S3VM	MR	Best Tuning	
maxF1	10	(29, 2, 49)	(16, 1, 63)	(14, 55, 11)	(27, 0, 53)	(24, 0, 56)	(13, 53, 14)	Test
	100	(39, 0, 41)	(34, 4, 42)	(31, 15, 34)	(39, 1, 40)	(44, 4, 32)	(26, 21, 33)	Test

Metric	l	S3VM	MR	Best Tuning	S3VM	MR	Best Tuning	
AUROC	10	(26, 0, 54)	(11, 0, 69)	(12, 57, 11)	(25, 0, 55)	(25, 0, 55)	(11, 56, 13)	Test
	100	(43, 0, 37)	(37, 0, 43)	(38, 8, 34)	(38, 0, 42)	(46, 0, 34)	(28, 24, 28)	Test

Table 3: Aggregate test performance comparisons versus SVM in 80 trials per setting. Each cell contains a tuple of the form "(#trials worse than SVM, #trials equal to SVM, #trials better than SVM)."

		$u = 100$				$u = 1000$				
Metric	l	SVM	S3VM	MR	Best Tuning	SVM	S3VM	MR	Best Tuning	
accuracy	10	0.61	0.62	0.67	0.68	0.61	0.63	0.64	0.67	Test
	100	0.81	0.82	0.83	0.85	0.81	0.82	0.83	0.85	Test

Metric	l	SVM	S3VM	MR	Best Tuning	SVM	S3VM	MR	Best Tuning	
maxF1	10	0.59	0.61	0.64	0.59	0.59	0.61	0.61	0.59	Test
	100	0.76	0.75	0.76	0.75	0.76	0.76	0.76	0.76	Test

Metric	l	SVM	S3VM	MR	Best Tuning	SVM	S3VM	MR	Best Tuning	
AUROC	10	0.63	0.64	0.72	0.61	0.63	0.64	0.67	0.61	Test
	100	0.87	0.87	0.87	0.87	0.87	0.86	0.87	0.86	Test

Table 4: Aggregate test results averaged over the 80 trials (8 datasets, 10 trials each) in a particular setting.

outperforms SVM in fewer trials than the other algorithms in some settings for these two metrics. This is because Best Tuning conservatively selects SVM in many trials. The take home message is that tuning using CV based on accuracy (and to a lesser extent maxF1) appears to mitigate some risk involved in applying SSL. AUROC, on the other hand, does not appear as effective for this purpose. Table 3 (bottom) shows that, for $u = 1000$, Best Tuning is worse than SVM fewer times, but for $u = 100$, MR achieves better performance overall.

We also compare overall average test performance (across datasets) for each metric and l,u combination. Table 4 reports these results for accuracy, maxF1, and AUROC. In terms of accuracy, we see that the Best Tuning approach leads to better performance than SVM, S3VM, or MR in all settings when averaged over datasets. We appear to achieve some synergy in dynamically choosing a different algorithm in each trial. In terms of maxF1, Best Tuning, S3VM, and MR are all at least as good as SL in three of the four l,u settings, and nearly as good in the fourth. Based on AUROC, though, the results are mixed depending on the specific setting. Notably, though, Best Tuning consistently leads to worse performance than SL when using this metric.

8.3 A Note on Cloud Computing

The experiments were carried out using the Condor High-Throughput Computing platform (Thain et al., 2005). We ran many trials per algorithm (using different datasets, l, u, and metrics). Each trial involved training hundreds of models using different parameter configurations repeated across five folds, and then training once more using the selected parameters. In the end, we trained a grand total of 794,880 individual models to produce the results in Table 2. Through distributed computing on approximately 50 machines in parallel, we were able to run all these experiments in less than a week, while using roughly three months worth of CPU time.

9 Conclusions

We have explored "realistic SSL," where all parameters are tuned via 5-fold cross validation, to simulate a real-world experience of trying to use unlabeled data in a novel NLP task. Our medium-scale empirical study of SVM, S3VM, and MR revealed that no algorithm is always superior, and furthermore that there are cases in which each SSL algorithm we examined can perform worse than SVM (in some cases significantly worse across 10 trials). To mitigate such risks, we proposed a simple meta-level procedure that selects one of the three models based on tuning performance. While cross validation is often dismissed for model selection in SSL due to a lack of labeled data, this Best Tuning approach proves effective in helping to ensure that incorporating unlabeled data does not hurt performance. Interestingly, this works well only when optimizing accuracy during tuning. For future work, we plan to extend this study to include additional datasets, algorithms, and tuning criteria. We also plan to develop more sophisticated techniques for choosing which SL/SSL algorithm to use in practice.

Acknowledgments

A. Goldberg is supported in part by a Yahoo! Key Technical Challenges Grant.

References

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, November.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*.

Ulf Brefeld and Tobias Scheffer. 2006. Semi-supervised learning for structured output variables. In *ICML06, 23rd International Conference on Machine Learning*, Pittsburgh, USA.

R. Bruce and J. Wiebe. 1994. Word-sense disambiguation using decomposable models. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 139–146.

Lijuan Cai and Thomas Hofmann. 2003. Text categorization by boosting automatically extracted concepts. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*.

Olivier Chapelle and Alexander Zien. 2005. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*.

Olivier Chapelle, Alexander Zien, and Bernhard Schölkopf, editors. 2006. *Semi-supervised learning*. MIT Press.

Tom Fawcett. 2004. ROC graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4.

Andrew B. Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 2009. May all your wishes come true: A study of wishes and how to recognize them. In *Proceedings of NAACL HLT*.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

M. Karlen, J. Weston, A. Erkan, and R. Collobert. 2008. Large scale manifold transduction. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 448–455. Omnipress.

Vikas Sindhwani and S. Sathiya Keerthi. 2007. Newton methods for fast solution of semi-supervised linear SVMs. In Leon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large-Scale Kernel Machines*. MIT Press.

V. Sindhwani and D. Rosenberg. 2008. An rkhs for multi-view learning and manifold co-regularization. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 976–983. Omnipress.

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. Beyond the point cloud: from transductive to semi-supervised learning. In *ICML05, 22nd International Conference on Machine Learning*.

Vikas Sindhwani, Sathiya Keerthi, and Olivier Chapelle. 2006. Deterministic annealing for semi-supervised kernel machines. In *ICML06, 23rd International Conference on Machine Learning*, Pittsburgh, USA.

Douglas Thain, Todd Tannenbaum, and Miron Livny. 2005. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356.

Xiaojin Zhu. 2005. Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison.